UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

MODELOS ELIPSOIDALES DE LA CABEZA HUMANA

T E S I S

QUE PARA OBTENER EL TÍTULO DE INGENIERO EN COMPUTACIÓN PRESENTA:

WALDO RAMÍREZ MONTAÑO

DIRECTOR DE TESIS:

DR. JORGE ALBERTO MÁRQUEZ FLORES

México, D.F. Noviembre 2005

PÁGINA INTENCIONALMENTE EN BLANCO

Agradecimientos...

A todas y cada una de las personas con las cuales he podido convivir y disfrutar un regalo inexplicable: la vida, en especial...

a mi madre,
a mi padre,
a mi hermana,
a mi hermano,

por su ser y amor, que será imposible terminar de agradecer ya que me enseñaron algo más importante: el significado de la palabra "familia".

Waldo Ramírez Montaño

PÁGINA INTENCIONALMENTE EN BLANCO

Los extremos se tocan

Índice

Toda palabra dicha despierta una idea contraria
Johann Wolfgang von Goethe

Introducción

Diversos estudios sobre la antropometría del cuerpo humano han propuesto una relación morfológica con la geometría, al usar modelos morfológicos basados en primitivas geométricas. Ante los avances tecnológicos en el ámbito de ingeniería, se expande la posibilidad de crear herramientas de análisis para atacar y clarificar dicha propuesta. En esta tesis se plasma un ejemplo, con el estudio sobre la relación morfológica entre la *cabeza humana* y ciertas *superficies geométricas* (*esferas y elipsoides*), con el diseño e implementación de un marco de referencia homogéneo, basado inicialmente en *PCA* (Análisis de Componentes Principales) y refinado mediante métodos de evaluación de errores morfológicos que hacen uso de *campos de distancia.* Para ello, después del análisis de requerimientos se hizo uso del control de versiones (*VC*) basado en *revisiones,* junto con una arquitectura de software de tres capas: *Modelo-Vista-Controlador* (*MVC*).

Con el fin de aprovechar los avances tecnológicos computacionales, respetando la planificación de recursos para el proyecto, se hizo uso de una infraestructura con sistema operativo *Linux* y desarrollo de software bajo *lenguaje C estándar,* incluyendo soporte en graficación tridimensional por computadora mediante una *API* versión de dominio público basada en *OpenGL*: *Mesa3D,* con la implementación de interfaces de acceso para el usuario mediante *GLUT*.

En el capítulo 1 se analizan los antecedentes del proyecto, se explica la preferencia de uso de elipsoides y se establecen los objetivos y metas específicos. El capítulo 2 describe el marco de referencia homogéneo que sostiene la relación entre cabezas y elipsoides, así como su comparación. El capítulo 3 reporta el papel de la ingeniería de software con el análisis de requerimientos y especificaciones, así como el detalle de la administración de la configuración y arquitectura de software para los programas resultantes; en el capítulo 4 su diseño, y abre la posibilidad de considerar superficies paramétricas especiales (parches spline, parches Bézier). Finalmente, el capítulo 5 ilustra los resultados a las expectativas con las discusiones y conclusiones correspondientes.

CAPÍTULO I

Antecedentes sobre modelos simplificados y ajuste de datos

1.1 Proyecto de construcción de Banco de Cabezas.

1.2 Necesidad de contar con modelos elipsoidales.

1.3 Proyectos relacionados.

1.4 Objetivos.

1.5 Metas y alcances de los programas y metodologías a desarrollar.

Prefiero un vicio tolerante que una virtud obstinada
Moliére

1.1 Proyecto de Construcción de Banco de Cabezas

Gracias a la evolución de los recursos computacionales gráficos, sobre todo en las últimas dos décadas [Wal94], es posible diseñar protocolos computacionales útiles en el desarrollo de herramientas de software con interacción en tiempo real para el usuario. Un ejemplo reciente es VRML (desarrollado por *Silicon Graphics* [FG95]), el cual es ampliamente usado en entornos de investigación (ver [Han01, WMRCS01, Ier04]), ya que busca ofrecer el marco de trabajo (framework) para la creación de sistemas libres de plataforma de graficación 3D interactiva mediante el esquema Web más popular: *Internet.*

El proyecto a partir del cual surgió la propuesta presente está relacionado con la dosimetría de los teléfonos celulares, así como con la morfometría de la cabeza humana (en especial en la región auricular) [MBBSG00]. Los estudios del proyecto anterior han evolucionado, dando fruto a metas adicionales, como el desarrollo de métodos para la creación de un atlas computacional y asistir en problemas de antropometría craneofacial [MBBSG03].

La antropometría ha tomado mayor importancia debido al nuevo surgimiento de dispositivos físicos de uso humano y a la consideración de diferencias entre individuos y grupos (poblaciones). Por ejemplo, en México existen necesidades o situaciones que involucran estudios antropométricos, como los citados a continuación [Cau04]:

A) Gran cantidad de equipo y maquinaria laboral se importa de otros países altamente industrializados, los cuales se diseñaron para población no mexicana.
B) Fabricantes nacionales no suelen diseñar sus productos para la población mexicana, ya que comúnmente se basan en diseños importados.
C) No se conocen las características físicas (Cartas Antropométricas) de la población mexicana, y en casos donde si existen, se mezcla la población masculina y femenina, regiones del país e incluso intervalos de edades.

Afortunadamente en México ya existe la iniciativa de promover el análisis antropométrico grupal nacional, con proyectos como: *Determinación del perfil antropométrico en una empresa metalmecánica* [MM02], donde la población en estudio consistió en setenta trabajadores mexicanos masculinos que cumplen con los requisitos del perfil antropométrico que se establecieron en dicho proyecto; la cédula antropométrica de cada trabajador contiene 51 variables, de las cuales 6 se asocian a la cabeza y cara, cuyos resultados se muestran en la *tabla 1.1*, usando la nomenclatura siguiente:

Variable antropométrica	*Descripción*
22	*Circunferencia de la cabeza*. Se toma la máxima circunferencia de la cabeza, medida por encima de las cejas.
23	*Distancia de oído a oído sobre la cabeza*. Se toma la distancia desde el centro de un oído hacia el centro del otro pasando sobre la cabeza.
24	*Ancho de la cara a la altura de las patillas*. Se toma el ancho de la cara medida a través de las proyecciones más laterales de los huesos temporales (arco cigomático).

25	*Ancho de la cabeza*. Se toma la máxima medida horizontal de la cabeza sobre las orejas (parte superior del pabellón auricular).
26	*Altura de la barbilla a la parte superior de la cabeza*. Se toma la distancia del límite inferior del maxilar inferior (gntion) al nivel superior de la cabeza (vertex).
27	*Longitud de la cabeza*. Se mide la máxima longitud de la cabeza medida de la frente (glabela) a la parte posterior más sobresaliente de la cabeza (occipital).

Variables antropométricas	*Media*	*Mediana*	*Moda*	*Desviación estándar (±)*	*Percentil 25*	*Percentil 75*
22	57.13	57.00	57.00	1.54	56.38	58.03
23	37.60	38.00	38.00	1.55	36.50	38.78
24	14.85	14.95	15.00	0.64	14.48	15.30
25	15.87	15.90	15.60	0.55	15.60	16.22
26	25.18	25.40	25.50	0.94	24.40	25.73
27	19.39	19.40	19.60	0.80	18.90	19.80

Tabla 1.1 Resultados "*Variables Antropométricas de cabeza y cara*", del proyecto "*Determinación del perfil antropométrico en una empresa metalmecánica*". Métrica de medición: *centímetros*.

La *tabla 1.1* refleja que la máxima relación geométrica, desde el punto de vista morfológico, que se le asocia a la cabeza humana es la circunferencia (variable 22), considerando únicamente la porción superior a las cejas. La asociación geométrica de otras variables (23, 24), sugieren la posibilidad de expresar la cabeza humana como una superficie geométrica: el análisis que se presenta a continuación, se basa en metodología y justificación matemática que proponen superficies geométricas (esfera, elipsoide) asociadas a la cabeza humana.

1.2 Necesidad de contar con modelos elipsoidales

El análisis de una población de modelos de cabezas humanas requiere un marco de referencia homogéneo para establecer un mismo sistema de coordenadas para distintos modelos y conjuntos de datos, y por otra parte un espacio de trabajo formal con criterios de comparación válidos; existen diversas estrategias de análisis estadístico de datos [AK05], por ejemplo: *Clustering (jerárquico, K-medias,* SOM) o *PCA*.

El perfil de los modelos de cabezas humanas, al ser considerados como sólidos rígidos, establece un esquema matemático cuyo criterio permite poder representarlos con una expresión (ecuación) matemática correspondiente a una superficie geométrica; es decir, mencionando brevemente conceptos presentes en el capítulo II, a partir de un modelo tridimensional alineado puede ser generada una superficie de n-orden que es matemáticamente equivalente de acuerdo al criterio de comparación establecido por los pa-

rámetros que se muestran en la *tabla 1.2*, en el caso más simple posible: el de un elipsoide en forma canónica (es decir, sin considerar posición y orientación).

n	***Parámetros***	***Ecuación***	***Ejemplo***
0 (esfera)	r	$x^2+y^2+z^2=r^2$	$x^2+y^2+z^2=75^2$
1 (elipsoide)	a, b, c	$\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}=1$	$\left(\frac{x}{100}\right)^2+\left(\frac{y}{65}\right)^2+\left(\frac{z}{80}\right)^2=1$

Tabla 1.2. En esencia, una esfera es una elipsoide: *r=a=b=c*

Ejemplos:

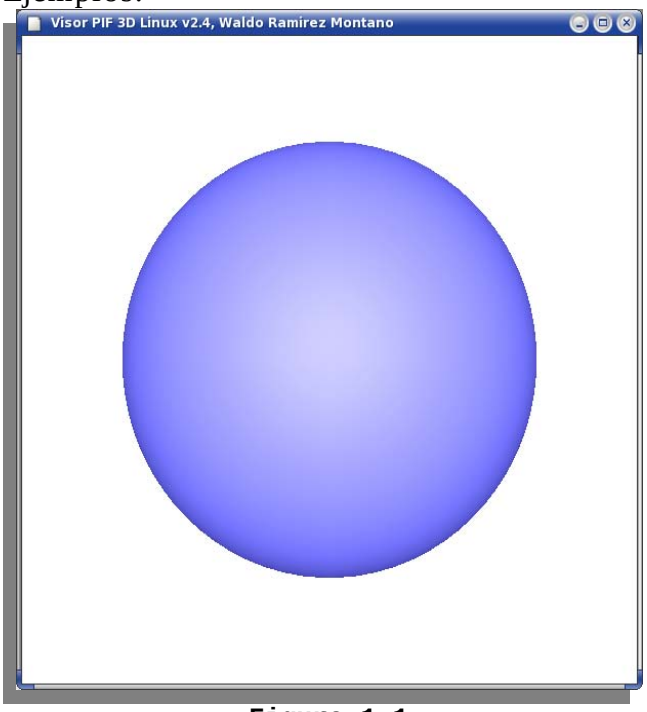


Figura 1.1

Esfera de *r=75*, obtenida con:
./creaElipsoide ./F01_01.pi 20 75 75 75

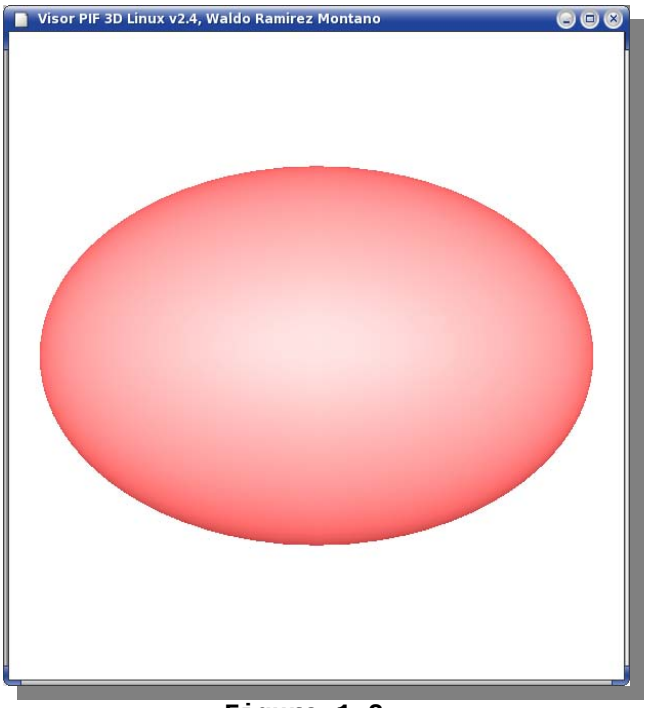


Figura 1.2

Elipsoide de *a=100, b=65, c=80*, obtenida con:
./creaElipsoide ./F01_02.pi 20 100 65 80

Ambos ejemplos cuentan con 20 como valor para el *nivel de resolución de puntos*[1] y se emplea al programa *visor* para su despliegue; se hace uso de proyección *Frustum* y cálculo de normales *Gouraud*.

A pesar de que tradicionalmente los modelos orden cero, las esferas, han sido la representación morfológica de la cabeza, cráneo, e incluso cerebro humano [DSL02, Bie03, SDBGSHF04], la propuesta de aproximación geométrica presente requiere el emplear modelos orden uno, las elipsoides, ya que a diferencia de las esferas en forma canónica (que solo tienen un parámetro de variación geométrica, el radio -r-), las elipsoides en forma canónica tienen tres parámetros (los ejes principales, -a, b, c-), que proporcionan mejor aproximación geométrica y morfológica al modelo; cabe señalar que el considerar la forma no canónica de la elipsoide implica la presencia de parámetros adicionales (posición y orientación), con lo cual se presentan seis grados de libertad adicionales.

1 Resolución discreta de la superficie para los puntos que la componen. Se explica a detalle en el capítulo IV.

Con dicho marco de referencia homogéneo es viable el análisis de resultados entre el ajuste de dos o más elipsoides y modelos de cabezas, ya que en estas últimas también es posible obtener los tres parámetros mencionados, como se aprecia en el siguiente capítulo.

1.3 Proyectos relacionados

Existe una gran diversidad de proyectos de investigación relacionados con conceptos de antropometría, dosimetría y morfología, del cuerpo humano. Como caso particular, el vínculo principal del presente trabajo es con los proyectos siguientes:

(A) Construcción de modelos de cabezas humanas para estudios de antropometría y dosimetría de antenas RF y de teléfonos celulares.

En el proyecto se hizo un estudio de la absorción de ondas de radio-frecuencia que tiene la cabeza del cuerpo humano ante el uso de antenas y dispositivos de comunicación móvil (en particular, el teléfono celular). Se estableció que la absorción mencionada depende de características anatómicas complejas: en particular en las regiones del oído y mentón. Para poder realizar los estudios de dosimetría con teléfonos celulares, a través de un *escáner láser 3D*[2] se adquirieron imágenes cilíndricas en escala de gris que fueron procesadas para crear una base de datos de maniquíes ("phantoms") antropométricos, generando así los perfiles de construcción de los modelos 3D. A partir de los perfiles, se crearon las superficies trianguladas en formato VRML versión 1.0, con las cuales se realizaron las simulaciones discretas para los estudios de problemas de dosimetría y antropometría, destacando el análisis a la región del oído humano, como se muestra en las *figuras 1.3 a la 1.7*.

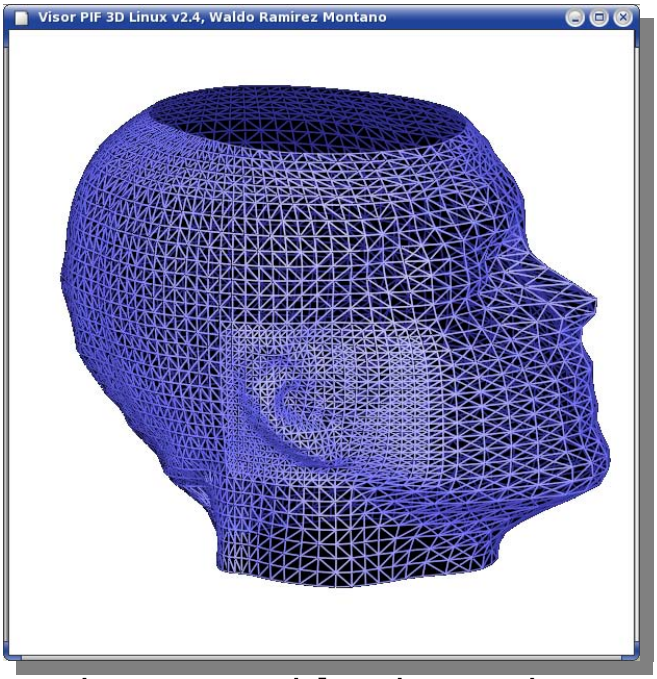


Figura 1.3. Modelo cabeza38.pi en modo gráfico Wireframe.

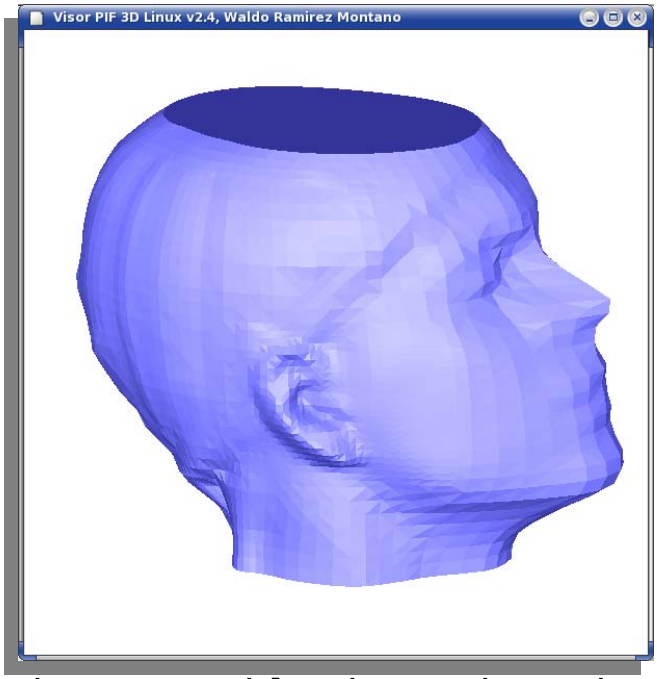


Figura 1.4. Modelo cabeza38.pi en modo gráfico sólido y cálculo de normales planas.

El modelo original, en formato *VRML vers.1.0*, se tradujo al formato *PIF* y se desplegó con nuestro programa *visor*, usando proyección *Ortho*.

2 Inc. Cyberware Laboratory, "4020/rgb 3d scanner with color digitizer". http://www.cyberware.com .

En la *fig.1.3* se observa mayor resolución de puntos en la región del oído, y por ende, la presencia de mayor cantidad de polígonos (triángulos) que se aprecia en el despliegue en modo gráfico sólido de la *fig.1.4*. Como punto adicional, la presencia del casquete incompleto en el modelo refleja la relación crucial que se tiene, con el proyecto presente, pues una aplicación de la elipsoide se encarga de completar la región faltante.

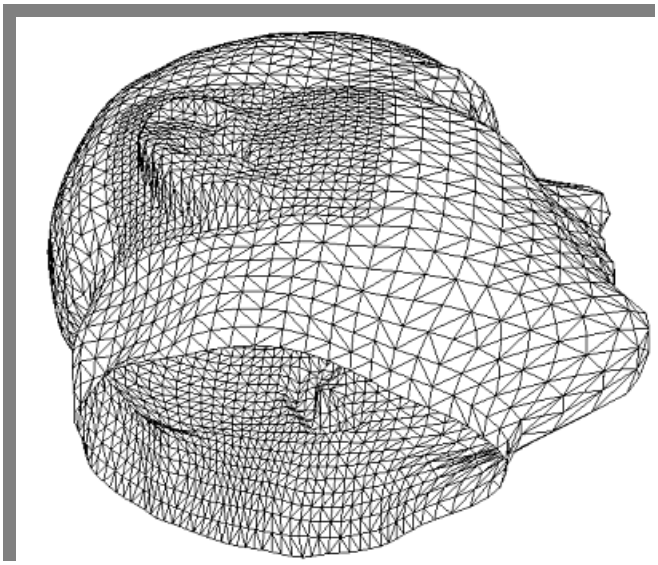


Figura 1.5. Modelo cabeza39.pi con oreja.

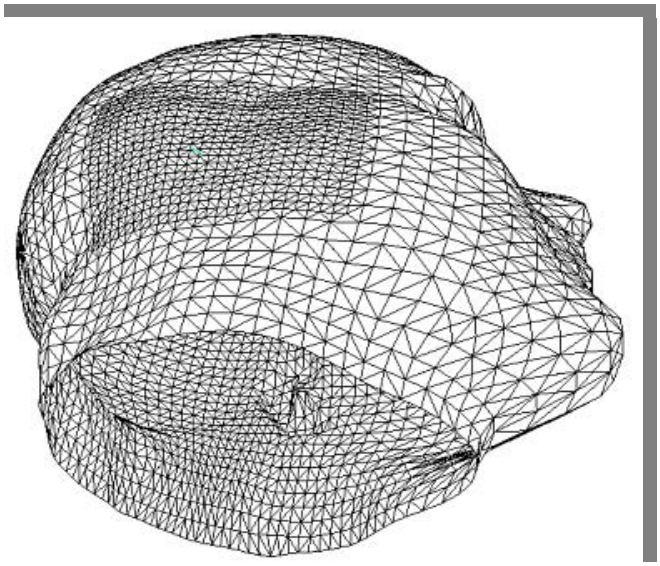


Figura 1.6. Modelo cabeza39.pi sin oreja.

Ante una simplificación de una población de modelos de cabezas, se plasma la oportunidad de generar un modelo geométrico -una elipsoide- morfológicamente representativo que puede ser complementado al agregarle rasgos y orejas. Con lo anterior se apoyan a estudios relacionados, como el análisis de los efectos biológicos de la proximidad e intensidad de los campos electromagnéticos generados por el teléfono celular [SF02].

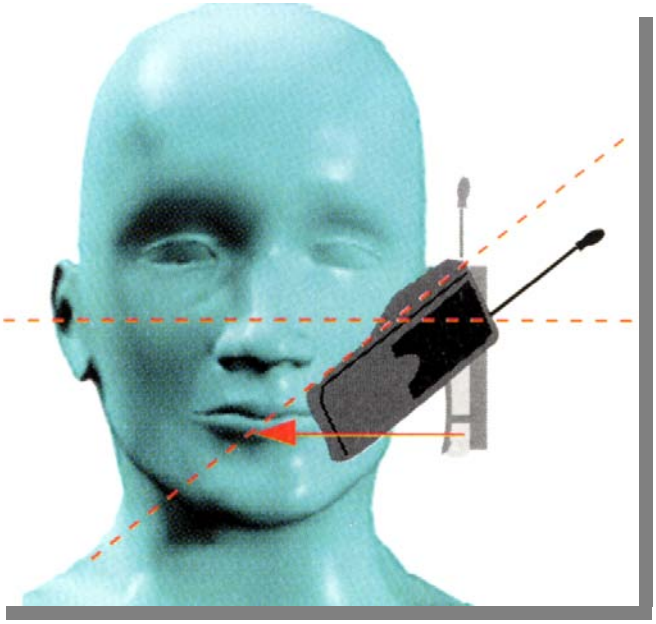


Figura 1.7. La posición del teléfono celular influye en las medidas resultantes del SAR local.

Cabe señalar que en la planificación y desarrollo del proyecto, se contó inicialmente con la colaboración del *Departamento de Tratamiento de Señales e Imágenes* de la *Ecole Nationale Supérieure des Télécommuncations*[3], así como de los laboratorios de dosimetría de antenas

3 ENST, Escuela de Ingeniería especializada en Telecomunicaciones, Informática y Tecnología de la Información, ubicada en París, Francia. http://www.enst.fr/

del sitio Marcoussis, de la empresa *Alcatel*[4]. Dichos laboratorios tienen interés en explotar la base de datos que se creó; un primer paso es una alineación robusta y mejorada a la que se tiene actualmente.

(B) Promedio morfológico de anatomías para la construcción de atlases y para antropometría craneofacial.

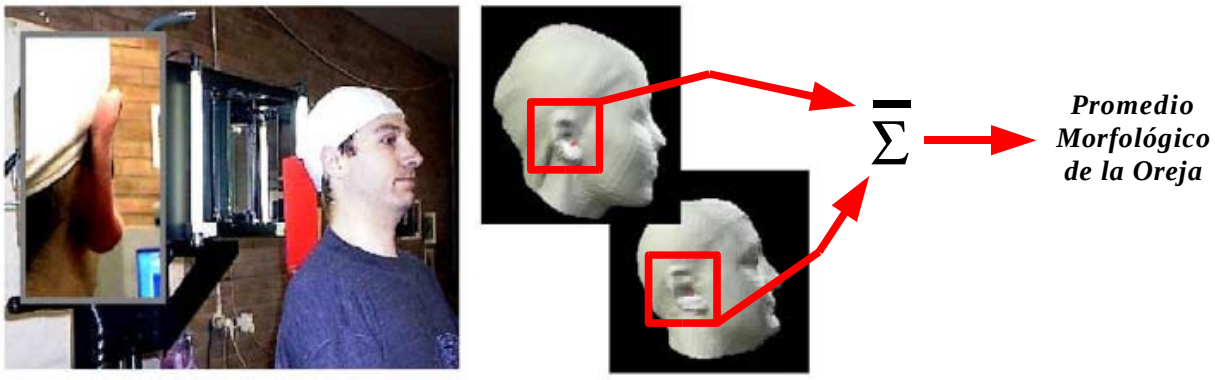


Figura 1.8. Ejemplo de un caso de estudio. ***A través de los modelos de cabeza humana (obtenidos mediante el escáner mencionado en el proyecto anterior), se analiza a detalle a la oreja mediante la metodología propia del proyecto; con ello se obtiene al modelo representativo de la oreja (es decir, su promedio morfológico).***

En ese proyecto (el cual nace del proyecto descrito en el inciso anterior) se requiere tener una mejor alineación de los modelos de cabezas humanas y un sistema de referencia de coordenadas homogéneas para dichos modelos; lo anterior implica el discernir entre defectos de correspondencia de rasgos debidos a la alineación y defectos de correspondencia debidos a variaciones intrínsecas entre individuos.

1.4 Objetivos

El objetivo central consiste en la mejora de las técnicas de construcción del banco de modelos tridimensionales de cabezas humanas y su aplicación en extracción de información antropométrica, para el grupo de 40 modelos de cabezas distintas que tiene disponible el *Laboratorio de Análisis de Imágenes y Visualización* del *Centro de Ciencias Aplicadas y Desarrollo Tecnológico*[5], *Universidad Nacional Autónoma de México.* Para ello planteamos los siguientes objetivos particulares:

a) Escoger y explotar los recursos computacionales.
- Sistema Operativo: Linux.
- Entorno gráfico: API basado en OpenGL [SGI92] sobre compilador *gcc*.

4 Empresa dedicada al desarrollo de tecnologías de comunicaciones, que opera a nivel internacional. http://www.alcatel.com

5 CCADET, UNAM. Centro de investigación original científica aplicada, desarrollo tecnológico y formación de recursos humanos de alta calidad en diversos ámbitos de ingeniería, para desarrollar procesos y productos innovadores que ayuden a resolver problemas prioritarios del país de interés nacional. http://www.cinstrum.unam.mx/

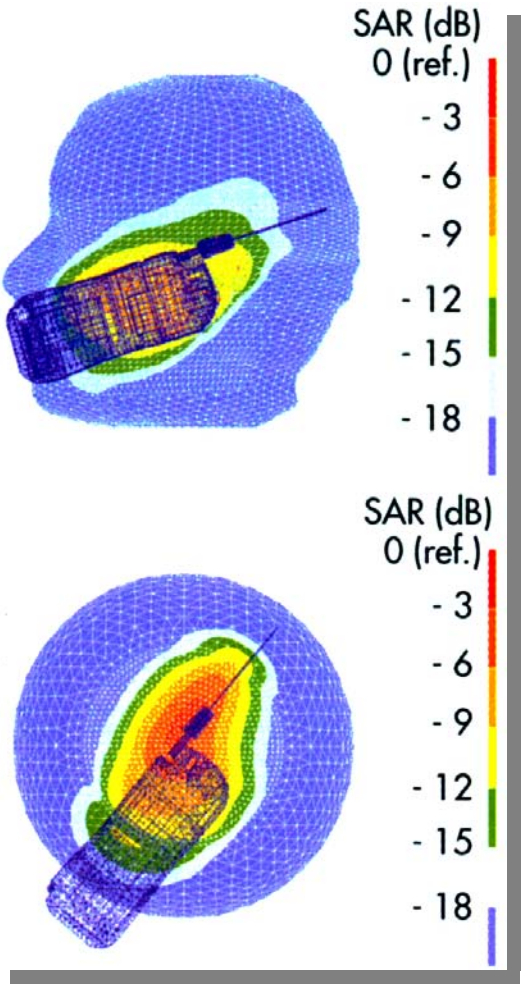
b) Implementar el marco de referencia homogéneo para estandarizar el análisis a todo modelo.
   - Discriminar puntos (regiones) alejados "de la cabeza humana", es decir, del modelo preliminar; esto hará más robusto al modelo. A dichos puntos los denominamos como *puntos externos (outliers)*.
   - Alinear los modelos.

c) Crear la elipsoide representativa de cada modelo.
   - Hacer la comparación cabeza-elipsoide con campos de distancia, obteniendo la representación numérica del error.

d) Demostrar la influencia de puntos alejados (orejas, nariz, mentón) en la obtención de la elipsoide más representativa del modelo, desde el punto de vista geométrico-morfológico.

e) Obtener una elipsoide representativa de una población de cabezas humanas.

f) Completar el casquete craneal de los modelos tridimensionales con las elipsoides resultantes.



Uno de los resultados fue el generar a la elipsoide representativa de la población de modelos de cabezas del proyecto, a partir de la elipsoide representativa de cada modelo. Lo anterior apoya a estudios como los mostrados en las figuras 1.9, 1.10.

Figura 1.9. Una opción al análisis del SAR en la cabeza humana, hace uso de un *modelo maniquí físico o por computadora,* en este ejemplo: un esferoide.

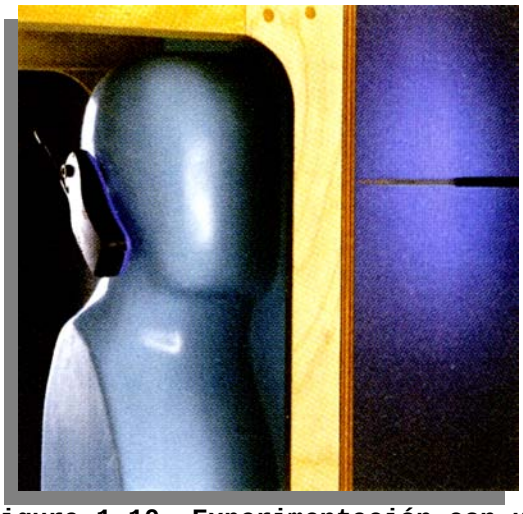


Figura 1.10. Experimentación con un modelo maniquí físicamente construido en materiales sintéticos.

1.5 Metas y alcances de los programas y metodologías a desarrollar

Como punto de partida es esencial enfatizar que, ante los avances tecnológicos presentes en la electrónica y computación de nuestros días, existe una relación crucial entre la evolución de las ciencias de investigación y la ingeniería en computación. Al aprovechar la oportunidad de planear, diseñar e implementar sistemas de cómputo que explotan a dichas tecnologías, se obtiene un apoyo significativo en la obtención de resultados científicos, sobre todo el agilizar al transcurso de la investigación sobre la dimensión del tiempo: mayor cantidad de resultados con base teórica en menor tiempo.

Ante las ventajas tecnológicas mencionadas, las metas, alcances y metodologías establecidas para los programas del proyecto presente fueron: el desarrollo de un formato interno (de modelos tridimensionales) compatible para toda operación, así como una interfaz visual (interactiva en tiempo real) que permitiera llevar a cabo las transformaciones geométricas necesarias para la alineación de los modelos. A su vez, se buscó un ajuste geométrico iterativo, requerido para que la comparación morfológica entre dos modelos generara una estimación aceptable de su error morfológico.

CAPÍTULO II

Alineación de datos en 3D a primer orden (elipsoides).

2.1 Modelos simplificados por análisis de componentes principales (PCA).

2.2 Aplicación en alineación geométrica.

2.3 Otros métodos de alineación.

2.4 Ajuste robusto de modelos elipsoidales.

Nunca pienso en el futuro porque llega muy pronto
Albert Einstein

2.1 Modelos simplificados por análisis de componentes principales PCA - Principal Components Analysis.

Considérese a $n > 2$ como la cantidad de dimensiones propias del conjunto de m datos que corresponden a un modelo n-dimensional. El *análisis de componentes principales* (PCA) aplicado a dicho modelo se interpreta como un procedimiento estadístico multivariable que manipula a los m datos n-dimensionales buscando que sus máximas variantes sean proyectadas a los ejes (también conocidos como *eigenvectores*), con lo que usualmente se reduce la dimensionalidad del modelo -disminuir a n- [HSEBCKB05], eliminando la menor cantidad de información posible e incluso brindando la posibilidad de compresión de datos [Ver01]. La justificación principal aplicada a la reducción de n, es el uso del método de *descomposición de valores singulares* (*SVD*) [DH03], que se detalla a continuación. El aspecto de "hallar la máxima variación" es lo que dará pie a ejes principales asociados a los datos del modelo bajo análisis.

En el espacio vectorial real n-dimensional, considérese una *matriz ortogonal A*, de dimensión $n \times n$ y a su matriz transpuesta A^T, en donde se cumple la definición del *eigenvector (***x***)* y *eigenvalor* (λ):

$$A^T A = \mathbf{1}_{n \times n}$$

$\mathbf{x} \in \mathbb{R}^n$, es un eigenvector de $A_{n \times n}$, si con el eigenvalor $\lambda \in \mathbb{R}$, se cumple:

$$A\,\mathbf{x} = \lambda\,\mathbf{x}$$

Y para cada eigenvalor λ_i, i =1, 2,..., n su *valor singular* es: $\sigma_i = \sqrt{\lambda_i}$ para $i = 1,2,\ldots,n$.

Con lo cual el teorema de la *Descomposición de Valores Singulares* [Yu02] establece que para la matriz ortogonal *A*, cuyo rango es *r*, existen dos matrices ortogonales, *U* y *V* tales que:

$$A = U\,\Sigma\,V^T = \sum_{i=1}^{r} \mathbf{u_i} \cdot \sigma_i \cdot \mathbf{v_i^T}$$

$$\text{donde } \Sigma = diag\,(\sigma_{1,}\sigma_{2,}\ldots,\sigma_n) \in \mathbb{R}^{n \times n}$$

Con esta descripción, se aprecia que la naturaleza de PCA es ser un método lineal de representación de información, que a su vez refleja los limites matemáticos que presenta.

Sinopsis del método PCA.
La interpretación estadística formal consiste en eliminar la correlación en los datos de forma lineal, obteniendo la mayor varianza. Sea una matriz $\boldsymbol{X}_{m \times n}$ de m datos con n dimensiones, $x_1, x_2, \ldots, x_n$, y sea $\boldsymbol{S}_{n \times n}$ la matriz de varianza-covarianza de los m datos. Existe un vector a_1 de longitud n tal que:

$a_1^T \boldsymbol{S}\, a_1$ es maximizado sujeto a $a_1^T a_1 = \mathbf{1}$, a_1 siendo un eigenvector de la matriz $\boldsymbol{S}$. A partir de la matriz $\boldsymbol{S}$, se hace el cálculo del primer *componente principal* (PC), definido como:

$z_1 = \sum_{i=1}^{n} a_{1i} x_i$ brindando la combinación lineal de los datos que da la *máxima variación* y por lo tanto, una orientación dominante del conjunto de datos.

De forma similar, un segundo componente principal se define de la manera siguiente:

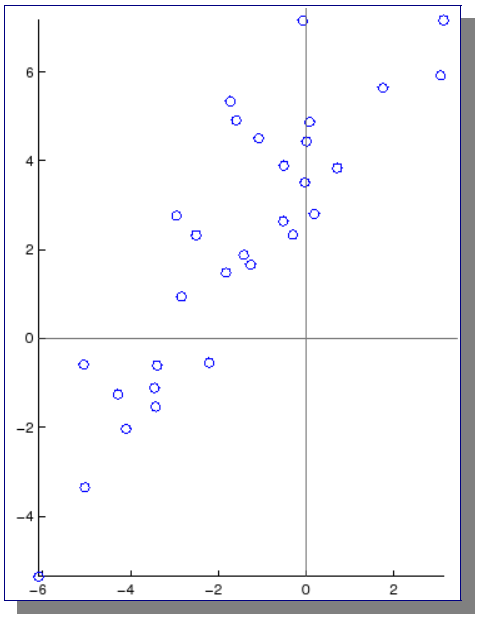
$z_2 = \sum_{i=1}^{n} a_{2i} x_i$ sujeto a que $a_2^T \boldsymbol{S}\, a_2$ es maximizado y ademas $a_2^T a_2 = \mathbf{1}$ y $a_2^T a_1 = \mathbf{0}$,

garantizando la ortogonalidad entre los PC, de acuerdo a SVD.

El resto de los componentes principales se obtiene de manera semejante, resaltando la ortogonalidad entre ellos. La reducción de la dimensión se obtiene ante la elección de los k componentes principales, considerando a $k < n$.

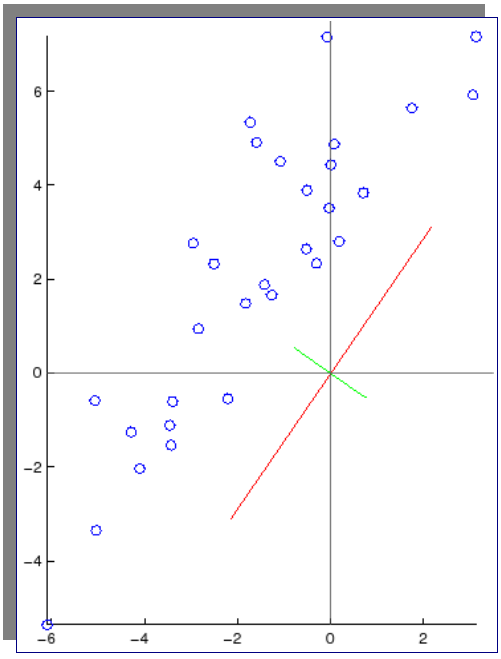
Ejemplo 2.A. Aplicación de PCA mediante Matlab 6.5[5]

Considérese a **X** como el conjunto de treinta datos aleatorios generados en un espacio bidimensional, los cuales representan al modelo a analizar, y se muestran en la *tabla 2.1* junto con la *fig.2.1*:



1	-2,84	0,95	***11***	-5,03	-3,35	***21***	-3,43	-1,53
2	-1,73	5,34	***12***	-0,52	2,64	***22***	-1,59	4,91
3	0,07	4,88	***13***	-4,29	-1,25	***23***	-0,03	3,52
4	0,01	4,44	***14***	3,06	5,93	***24***	3,13	7,17
5	-0,08	7,16	***15***	-3,39	-0,6	***25***	-1,08	4,51
6	0,7	3,84	***16***	-0,51	3,89	***26***	-2,21	-0,54
7	1,75	5,65	***17***	-0,3	2,34	***27***	0,18	2,81
8	-4,1	-2,03	***18***	-3,46	-1,11	***28***	-5,06	-0,58
9	-1,42	1,88	***19***	-6,09	-5,36	***29***	-2,95	2,77
10	-1,83	1,49	***20***	-2,5	2,33	***30***	-1,27	1,66

Tabla 2.1, Figura 2.1. Datos aleatorios generados.



Con fines ilustrativos, a continuación se observa un conflicto para el cumplimiento de la reducción dimensional del modelo: no hay *alineación* entre componentes principales y los ejes coordenados; además existe una *traslación* entre los componentes principales y los datos del modelo (*fig.2.2*).

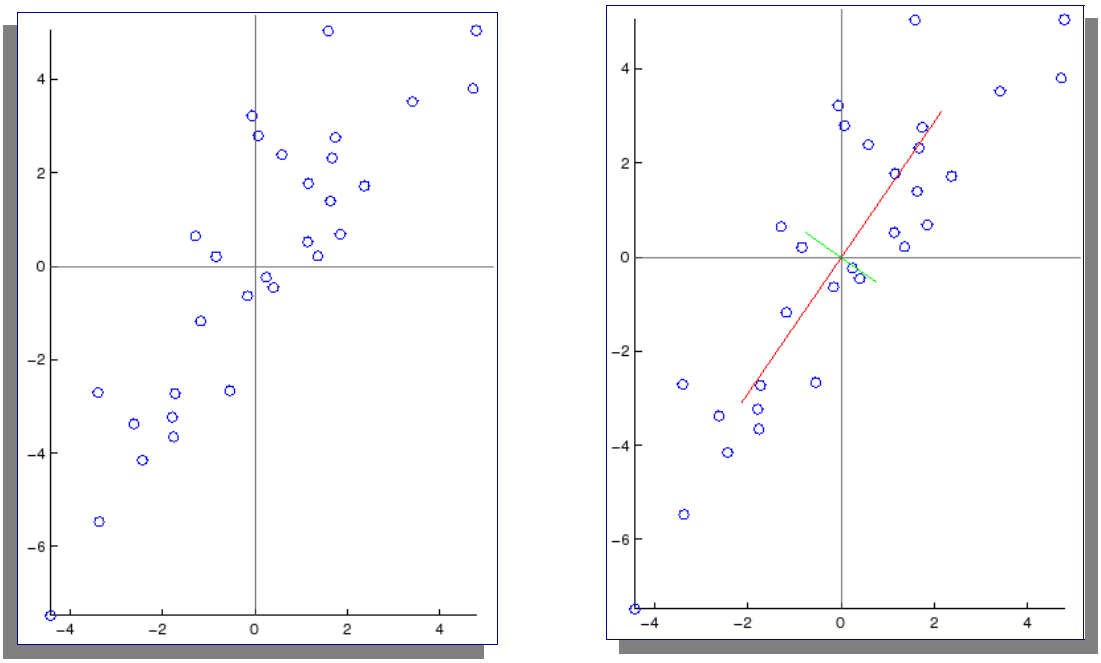
La matriz de covarianza también refleja que el modelo no está alineado, o mejor dicho: no está *geométrica-mente alineado*, ya que no es una matriz diagonal. Por lo anterior, el cálculo de los *componentes principales* no resultan en una matriz diagonal unitaria.

```
Cov = 5.1840   6.2437      PC = -0.5698   -0.8218
      6.2437   9.8591           -0.8218    0.5698
```

Figura 2.2. Previo a alineación geométrica

5 En el apéndice B puede consultar al *script* Matlab que fue empleado.

La traslación necesaria se consigue mediante la obtención de valor cero en la media estadística del modelo. Otra interpretación de ésta traslación es obtener el *centroide geométrico* (o bien, el *centro de masa*) en el origen del sistema de referencia. Para ello, se debe aplicar una transformación lineal simple a los datos del modelo (*figs. 2.3 y 2.4*), lo cual permite que para la alineación geométrica sólo falte la rotación del modelo.



Figuras 2.3 y 2.4. Aplicación de traslación. Falta rotar al modelo para alinearlo.

Cabe resaltar que Matlab permite el ajuste automático de la media para el cálculo de la matriz de covarianza y los componentes principales (PC). En el proyecto presente se implementaron los procedimientos, *excepto* la reducción de dimensiones (ya que no fue necesaria), por lo cual es sumamente importante indicar la traslación y rotación para alinear geométricamente los datos de cada modelo, ya que la comparación de los modelos de cabezas humanas siguió un protocolo que garantizara su validez.
Matemáticamente, la rotación se efectúa mediante la multiplicación matricial de los datos del modelo y los componentes principales:

$$X_r = (X^T \Omega)^T$$

donde: Ω es la matriz de los Componentes Principales (PC).

En la *fig.2.5* se aprecia la alineación geométrica (modelo posterior a la rotación). Otra prueba es que los componentes principales ya se representan como una matriz diagonal unitaria. En el análisis de las cabezas humanas no se redujo la dimensión de los modelos, aunque sí se desarrolló un método alternativo, ya que parte del problema es que algunos de sus puntos pueden no ser válidos o "*externos*" (outliers) a la población que se intenta modelar (*fig2.6*). Para complementar el ejemplo citado, a continuación se muestra la aplicación principal del método de PCA: *seleccionar las dimensiones más significativas con el fin de obtener la mejor representación homogénea del modelo con la mínima pérdida de información*. En este caso, se selecciona la dimensión con el eigenvalor más significativo: **14.1885**, es decir, la dimensión representada por el eje de mayor longitud (*fig.2.7*).

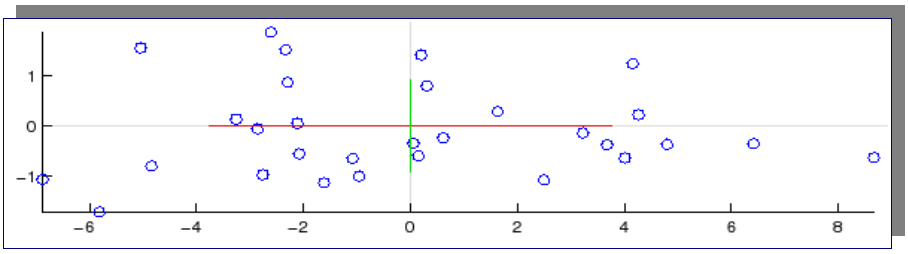


Figura 2.5. Modelo alineado geométricamente.
cov = 14.1885 0.0000 PC = -1.0 0.0
0.0000 0.8586 0.0 1.0

Como alternativa para los *puntos externos (outliers)* que predominan en las regiones de nariz, mentón y orejas de los modelos de cabezas humanas, fue necesario diseñar un método distinto a la reducción dimensional para modelar a la elipsoide representativa de mejor ajuste al casquete correspondiente.

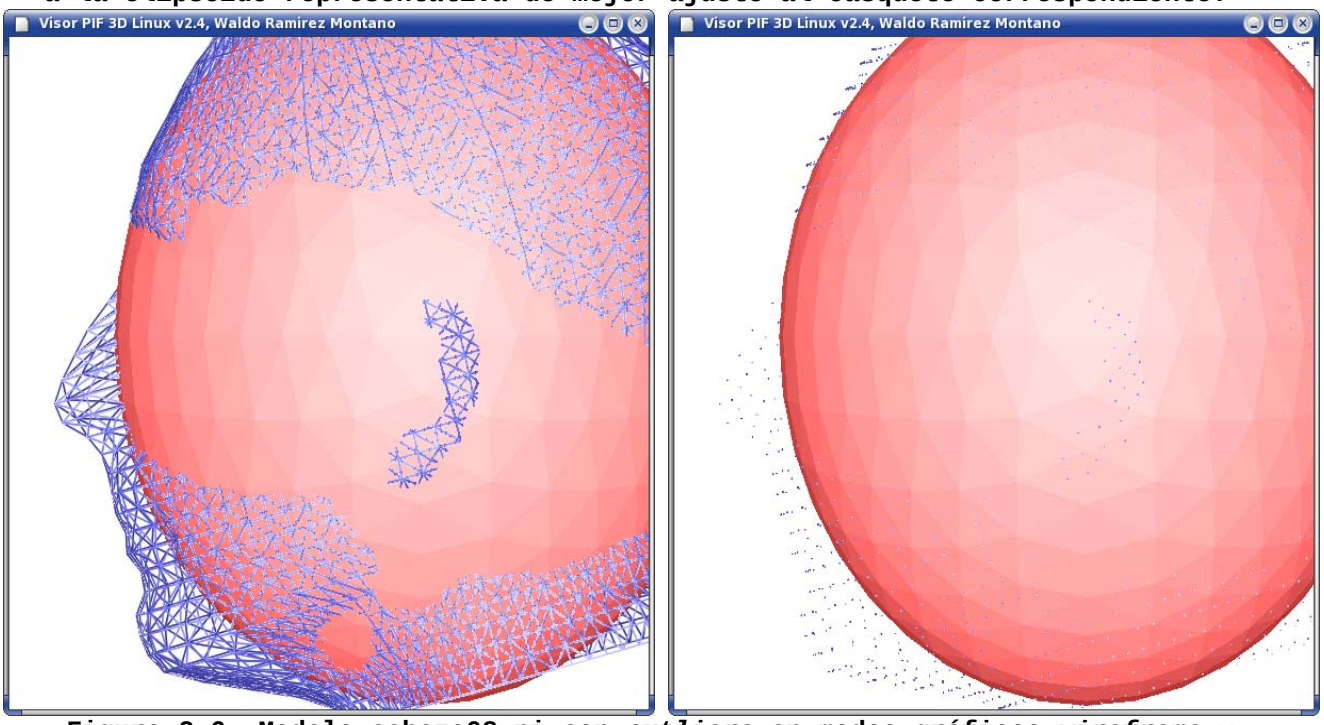


Figura 2.6. Modelo cabeza08.pi con *outliers* en modos gráficos *wireframe* y *puntos*, junto con su primer versión de elipsoide representativa (con *nivel de resolución* 10) en modo gráfico *solido*; ambos con proyección *Ortho*.

En la *fig.2.7* se aprecia la reducción dimensional, y en este ejemplo particular, resulta un modelo unidimensional. Con ello, se implementa la discriminación de datos, basada en el criterio de tolerancia ante la pertenencia de los datos al componente principal elegido.

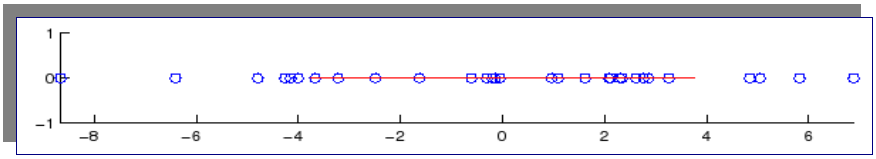


Figura 2.7. Reducción dimensional del modelo.

El método alterno para los modelos de cabezas humanas penaliza a los *outliers* sin decrementarles dimensión: así como en la reducción dimensional, se les aplica transformación

de traslación a los puntos externos, pero usando como límite un criterio basado en su elipsoide representativa, con el fin de alterar a los eigenvalores y eigenvectores del modelo de cabeza para regenerar a su elipsoide y mejorar su ajuste al cráneo (*fig.2.8)*.

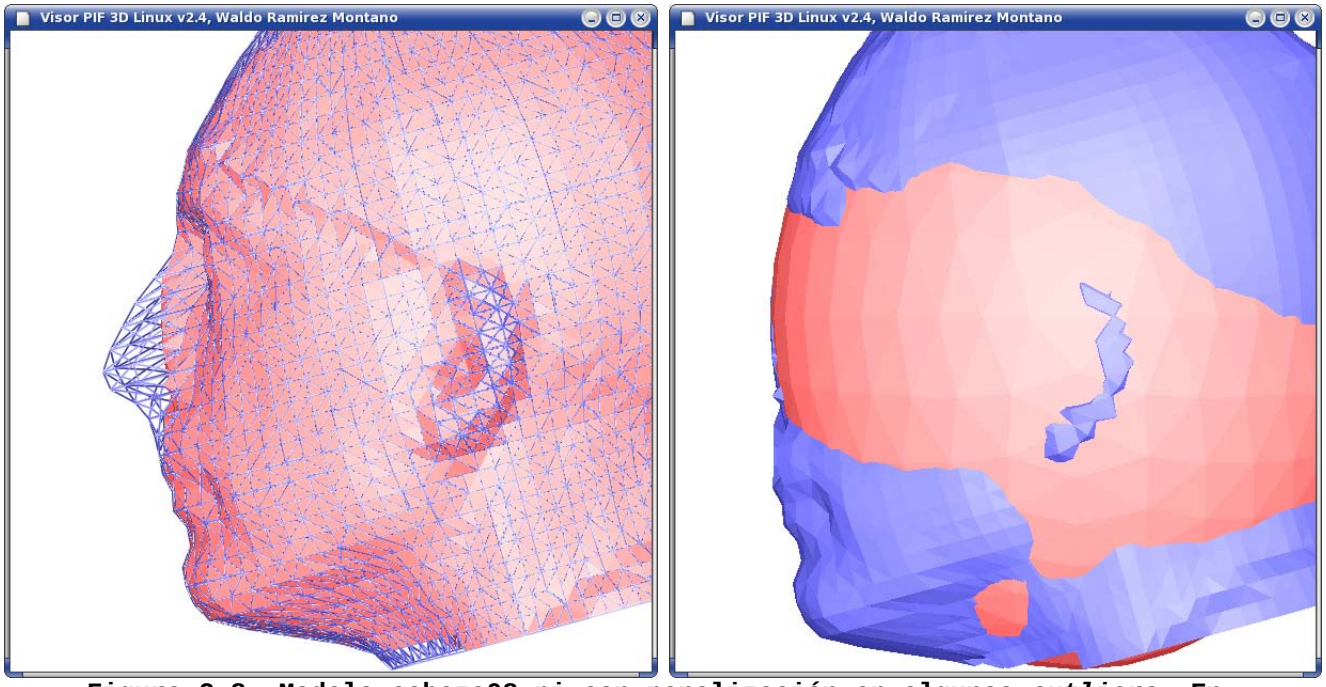


Figura 2.8. Modelo cabeza08.pi con penalización en algunos *outliers*. En esta figura se aplicó una tolerancia que trasladó a los puntos externos de la nariz y orejas, sin reducción dimensional. Diseñamos distintas tolerancias de penalización, como se describe en el capítulo IV.

Finalmente, en PCA se hace la rotación inversa (*fig.2.9*) y se elije la información basada en la discriminación mencionada; con esto se garantiza linealmente la menor pérdida de información debido a la ortogonalidad entre los componentes principales.

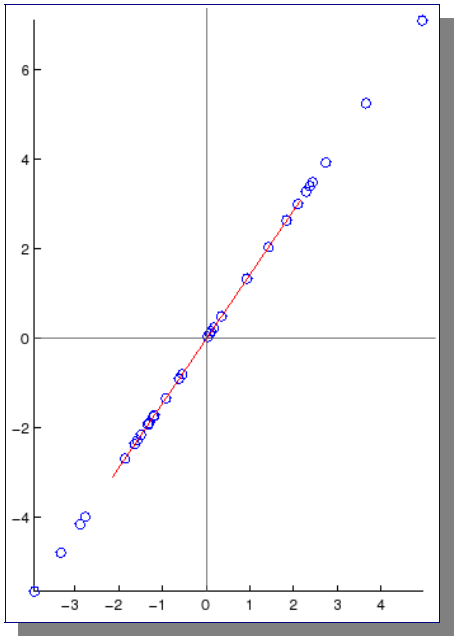


Figura 2.9. Modelo con la reducción dimensional a 1 eje.

De igual forma, desde el punto de vista matemático, la rotación inversa equivale a la multiplicación matricial inversa:

$$X = (X_r^T \Omega^{-1})^T$$

donde: Ω^{-1} es la matriz inversa de los Componentes Principales (PC).

Con el ejemplo anterior se aprecia la utilidad del PCA, aunque no se observa su limitación principal: *dependencia lineal del método* (no captura correlaciones no lineales entre dimensiones). Otra limitación es el que PCA no percibe más de *n* métodos de variación en *n* dimensiones: no distingue más de un grupo o conglomerado (*cluster*) de datos. En consecuencia, regiones de naturaleza distinta deben segmentarse y analizarse por separado.

2.2 Aplicación en alineación geométrica

Como se mencionó, en el presente proyecto se aplica una metodología basada en PCA: *alineación geométrica* [Már04]. En lugar de buscar la reducción dimensional en el modelo de datos para un análisis estadístico, el fin de la alineación es obtener una *orientación canónica* [LCB01] de los modelos tridimensionales, haciendo uso de la métrica Euclidiana de distancia. No es necesario implementar el cálculo independiente de la medida de similitud en el algoritmo de alineación geométrica, ya que lo incluye el uso del parámetro fundamental para obtener dicha orientación: el **tensor de inercia**, por lo que es importante señalar los conceptos de mecánica en la *geometría de masas* [DS01].

Inercia de un sólido
El punto de partida, es considerar a los modelos de cabezas humanas analizados como objetos sólidos rígidos, ya que no se considera que sufran deformaciones. Con lo anterior, la información necesaria para relacionar a los sistemas de fuerzas que actúan sobre un sólido a través del tiempo, son los momentos de orden cero, de primer orden y de segundo orden.

Distribución finita de masas
Todo modelo analizado en el proyecto, fue apreciado desde su perfil discreto en malla de alambre (wireframe) triangular, y por ello los modelos están compuestos de *n* puntos P_i, de masa finita m_i (vértices de los triángulos), obteniendo la masa total [Ell02] M, también conocida como *momento de orden cero*:

$$M = \sum_{i=1}^{n} m_i$$

Momentos de primer orden: momentos estáticos
La distinción de "primer orden" se refiere a la manipulación de distancias *con exponente uno*, por lo cual, distancias de los puntos másicos P_i con algún otro punto o línea o plano, generan un momento estático. Para la alineación geométrica interesa el momento estático con respecto a un punto: Se define a un momento estático como un vector $\boldsymbol{M_O}$ que, desde el punto de vista discreto, es la suma de cada producto de m_i con $\boldsymbol{OP_i}$ (distancia correspondiente entre P_i y un punto $\boldsymbol{O}$ cualquiera en el espacio) entre M,

$$\boldsymbol{M}_O = \sum_{i=1}^{n} \frac{\boldsymbol{OP_i}\ m_i}{M}$$

Debido a que $\boldsymbol{O}$ puede ser cualquier punto en el espacio, se obtiene un campo de vectores: un conjunto de momentos estáticos. Para el análisis de todo modelo de cabeza humana, el punto $\boldsymbol{O}$ considerado ha sido el origen. Por lo tanto, $\boldsymbol{OP}_i = \vec{\boldsymbol{x}}_i$, son las coordenadas de m_i.

Por ejemplo, en la *fig.2.10*, el modelo cabeza00.pi original tiene $\boldsymbol{M_O}$ = (13.33, 26.73, -74.64); cuando $\boldsymbol{O}$ es elegido de tal forma que el momento estático $\boldsymbol{M_O}$ sea nulo, se le denomina *centro de masa en el origen*. En la alineación geométrica es necesario *trasladar* los objetos para obtener su centro de masa en el origen, ya que junto con los ejes principales describen completamente la orientación del objeto. Cabe señalar que en el análisis geométrico, se le llama *centroide* ($\boldsymbol{C}$) al centro de masa. En la *fig.2.11* se aprecia el resultado de aplicar la traslación necesaria al modelo **cabeza00.pi**, con lo que se obtiene $\boldsymbol{M_O} = \boldsymbol{C}$ = (0.0, 0.0, 0.0).

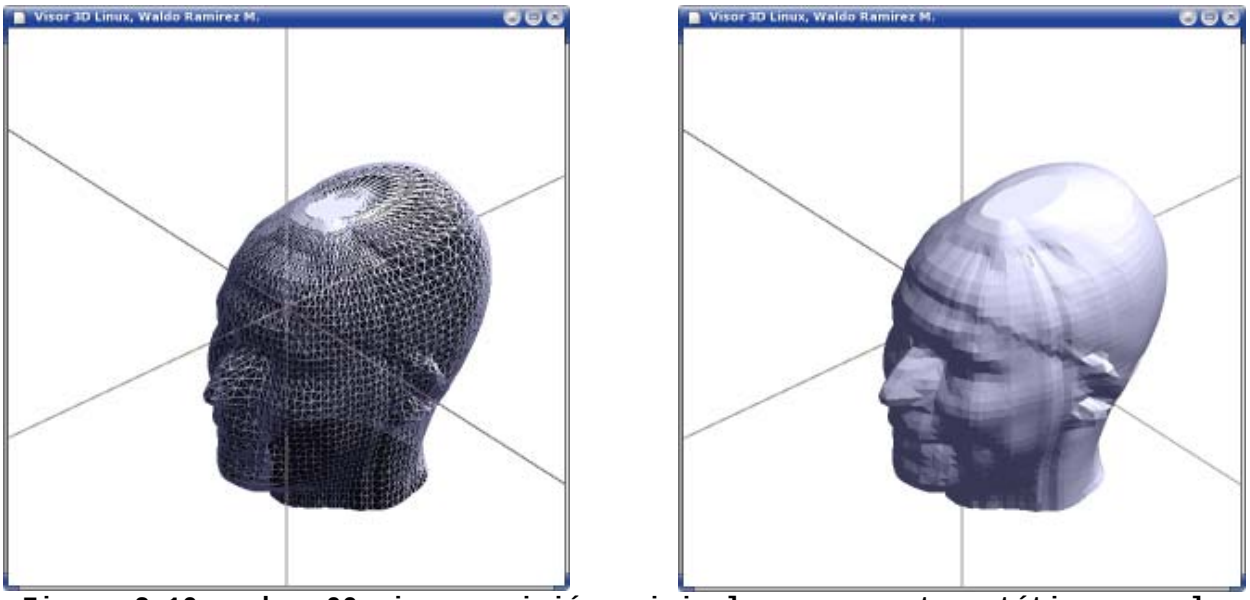


Figura 2.10. cabeza00.pi en posición original, con momento estático no nulo.

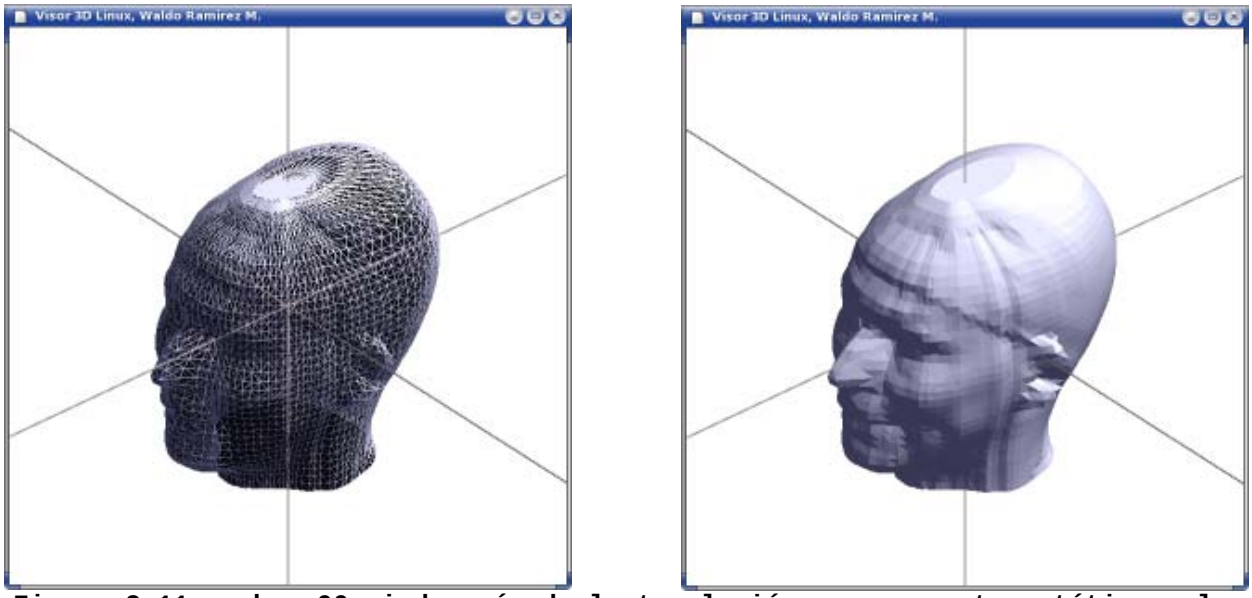


Figura 2.11. cabeza00.pi después de la traslación, con momento estático nulo.

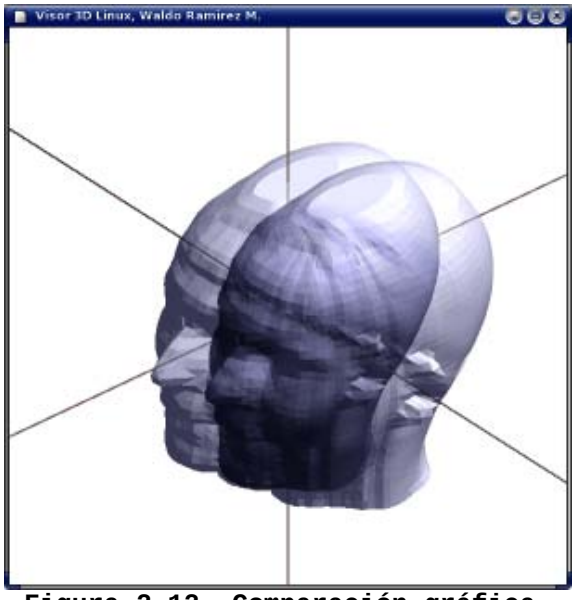


Figura 2.12. Comparación gráfica.

La *fig.2.12* ilustra que la traslación es una transformación geométrica lineal pura, como se detalla en el capítulo IV.

Momentos de segundo orden: momentos de inercia

Las características físicas de un cuerpo rígido pueden ser descritas por sus propiedades de inercia: masa (momento de orden cero) y momento de inercia [Kwo98]; la medida de inercia es, en el movimiento lineal, la masa del sistema y en su contraparte angular, el momento de inercia. Por ende, además de considerar la masa total del sistema, se analiza su distribución: *dos cuerpos con la misma masa total pero con distribución de masa distinta, tienen momentos de inercia distintos*. Como se puede intuir, el término *segundo orden* se refiere a manipulación de distancias con exponente dos ante los puntos másicos,

$$I_O = \sum_{i=1}^{n} \boldsymbol{R_i^2}\, m_i$$

Cabe señalar que cada $\boldsymbol{R_i}$, se refiere a la distancia mínima del punto másico m_i al eje de rotación unitario **h**. Considerando al análisis detallado presente en el apéndice A, un cuerpo rígido cuenta con momentos de inercia I_{xx}, I_{yy}, I_{zz} y con productos de inercia $I_{xy}, I_{yx}, I_{yz}, I_{zy}, I_{xz}, I_{zx}$:

$$I_{xx} \equiv \sum_{i=1}^{n} m_i (y_i^2 + z_i^2) \qquad I_{yy} \equiv \sum_{i=1}^{n} m_i (x_i^2 + z_i^2) \qquad I_{zz} \equiv \sum_{i=1}^{n} m_i (x_i^2 + y_i^2)$$

$$I_{xy} = I_{yz} \equiv -\sum_{i=1}^{n} m_i y_i x_i \qquad I_{xz} = I_{zx} \equiv -\sum_{i=1}^{n} m_i x_i z_i \qquad I_{yz} = I_{zy} \equiv -\sum_{i=1}^{n} m_i y_i z_i$$

Con lo cual, ante el eje de rotación, $\mathbf{h} = \cos\alpha\,\mathbf{i} + \cos\beta\,\mathbf{j} + \cos\gamma\,\mathbf{k}$, se obtiene la ecuación:

$$I_O = I_{xx}\cos^2\alpha + I_{yy}\cos^2\beta + I_{zz}\cos^2\gamma + 2I_{xy}\cos\alpha\cos\beta + 2I_{yz}\cos\beta\cos\gamma + 2I_{zx}\cos\gamma\cos\alpha$$

Con el fin de aclarar el objetivo primordial ante el uso de aplicación de alineación geométrica, es posible expresar la presencia del eje de rotación **h** mediante el uso de variables auxiliares,

$$u \equiv \frac{\cos\alpha}{\sqrt{I_O}}, \qquad v \equiv \frac{\cos\beta}{\sqrt{I_O}}, \qquad w \equiv \frac{\cos\gamma}{\sqrt{I_O}},$$

y al sustituirlos en la ecuación, se obtiene:

$$1 = I_{xx}u^2 + I_{yy}v^2 + I_{zz}w^2 + 2I_{xy}uv + 2I_{yz}vw + 2I_{zx}wu$$

La ecuación satisface la forma general de una elipsoide con centro en el origen del marco de referencia, y es conocida como *elipsoide de inercia*. Al aplicar la alineación geométrica se busca que los productos de inercia sean nulos, es decir,

$$\text{Si } I_{xy} = I_{yz} = I_{zx} = 0, \text{ entonces, } I_{xx}u^2 + I_{yy}v^2 + I_{zz}w^2 = 1$$

De esa manera, es posible calcular a los parámetros necesarios para obtener el modelo geométrico tridimensional que represente a dicha elipsoide en coordenadas canónicas: los momentos de inercia.

2.3 Otros métodos de alineación

Para mencionar otros métodos de alineación, a algunos se les incluye como parte de un proceso conocido como *Fusión de Datos* (*Data Fusion*) [HSEBCKB05]. El proceso combina datos y conocimiento de diferentes fuentes, cuyo objetivo es obtener la máxima información útil haciendo uso de una discriminación confiable, y así minimizando a los datos representativos necesarios: las modalidades a "fusionar" no se encuentran usualmente alineadas o en "registro geométrico". Los métodos de alineación en Fusión de Datos, se ilustran en la *tabla 2.2*:

Alineación de datos
Interpolación y correlación espacial o temporal
Métricas de distancia: Euclidiana, Minkowsky, Manhattan, Mahalanobis
Métricas de correlación
Figuras de Mérito
Clustering, Votación (Voting), Entropía, Álgebra de Imágenes (Image Álgebra)
Mínimos cuadrados, error cuadrático medio, máxima verosimilitud

Tabla 2.2. Data Fusion (también conocido como *Sensor Fusion*). Su aplicación práctica ha sido en áreas donde los resultados necesarios del análisis no tienen una métrica directa, por ejemplo: procesamiento de imágenes médicas.

Como se aprecia, existe una diversidad de estrategias en Fusión de Datos; incluso también se emplea a PCA, pero únicamente para obtener datos que son argumentos para otros métodos de alineación, ya que en ocasiones no es adecuado considerar al origen de datos en el centroide. En Fusión de Datos, se tiene el fin de atacar problemas específicos donde la alineación de datos se ejecuta a través de ajustes referentes a espacio y tiempo, elección del sistema de coordenadas y transformaciones que establezcan un marco de trabajo común para el procesamiento de datos. El considerar *alineados* a dos conjuntos de datos requiere de medidas de similitud para optimizar la alineación.

Métricas de distancia espacial y métricas estadísticas.
Una estrategia de alineación de datos consiste en establecer la *métrica de distancia* que se emplea en el sistema coordenado común. Por ejemplo, como se especifica en la sección 2.2, la métrica que se estableció ante el diseño del proyecto es una implementación particular de la distancia Minkowsky: la *distancia euclidiana.*

$D_p(X_j, X_k)=\sqrt[p]{\sum_{i=1}^{n} \lvert x_{ij}-x_{ik}\rvert^p}$	$D_1(X_j, X_k)=\sum_{i=1}^{n} \lvert x_{ij}-x_{ik}\rvert$	$D_2(X_j, X_k)=\sqrt{\sum_{i=1}^{n} \lvert x_{ij}-x_{ik}\rvert^2}$
Distancia Minkowsky	Distancia Manhattan (p=1)	Distancia Euclidiana (p=2)

Donde: $X_j=(x_{1j}, x_{2j}, ..., x_{nj}), X_k=(x_{1k}, x_{2k}, ..., x_{nk})$, en el espacio $\Re^n$.

La distancia Mahalanobis se basa en conceptos estadísticos: correlaciones entre variables mediante las cuales diferentes patrones pueden ser identificados y analizados,

$$D_M(x)=\sqrt{(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

donde: $x=(x_1, x_2, ..., x_n)$, son los valores de los atributos en la observación.
$\mu=(\mu_1, \mu_2, ..., \mu_n)$, son los valores de comparación. Usualmente, son las medias estadísticas de los atributos.
Σ^{-1}, es la matriz de covarianza inversa.

Si se considera a la matriz de covarianza como la matriz identidad, nuevamente se presenta una implementación de la distancia Euclidiana.

Métricas de correlación.
También conocidas como *métricas de asociación,* asignan un valor numérico que representa al *grado de similitud* entre los datos. La métrica puede ser basada en distancia espacial o mediciones estadísticas de correlación entre observaciones y predicciones (por ej., la distancia Mahalanobis), funciones heurísticas como figuras de mérito, y medidas que cuantifican el realismo de una observación basándose en suposiciones previas [Kle04].

Existen *medidas de similitud globales* [Dia01] entre dos estados (E_A, E_B); como en el caso de las métricas de distancia, las medidas de similitud Euclidiana y Manhattan son un caso particular de la medida de similitud Minkowsky [MMCL04],

$$\mathrm{sim}(E_A, E_B)=\sqrt[r]{\frac{1}{p}\sum_{i=1}^{i=p} [\, w_i \cdot \mathrm{sim}_i(x_i^A, x_i^B)]^r}$$

donde: $\mathrm{sim}_i(x_i^A, x_i^B)$, son las medidas de similitud local,
x_i^A, x_i^B, son los atributos de cada medida de similitud local,
$W=(w_1, w_2, ..., w_i, ..., w_n)$, es el *peso* que se le da a cada medida,
p es la cantidad de medidas de similitud local.

La métrica de similitud Manhattan se obtiene con r=1, mientras que la métrica de similitud Euclidiana se obtiene con r=2. Se aprecia que en la medida de similitud Minkowsky la variación de r implementa casos específicos, ya que no llega a ser conveniente establecer una medida de similitud general para objetivos particulares, porque cada uno de ellos requiere su aproximación eficaz. Así lo indica la medida de similitud de [BS03], donde la medición del grado de parecido de formas entre dos objetos, se basa en el *trabajo (o energía)* necesario para la transformación de uno en otro. Para ello, su alineación conlleva la previa normalización de escala entre los objetos a comparar -deben tener el mismo número de *spels* (elementos espaciales: en 2D *pixels*, en 3D *voxels*)- para posteriormente aplicar el criterio de ejes principales con el fin de establecer al apto ambiente de partida. El trabajo necesario (W) para la transformación de cada *spel* (el movimiento de una distancia s_i) esta dado, en unidades de trabajo, por:

$$W = \sum_{i=1}^{n} s_i$$

Al establecer la normalización (de 0 a 100%) de la medida de similitud (el trabajo) en un grupo de N objetos a ser comparados, dicha medida entre dos objetos (O_1, O_2), se puede escribir como:

$$S_{1,2} = \left(1 - \frac{W_{1,2}}{W_{máx}}\right) * 100$$

donde: $S_{1,2}$, es la medida de similitud entre los objetos 1 y 2,
$W_{1,2}$, es el trabajo realizado al comparar a los objetos 1 y 2.
$W_{máx}$, es el máximo trabajo realizado al comparar N objetos entre sí.

Las *figuras de mérito* son funciones que miden la concordancia entre los datos resultantes y el ajuste al modelo, para la elección particular de parámetros. Por convención, la figura de mérito (también conocida como *función de mérito*) devuelve un valor numérico pequeño cuando la concordancia es buena (el ejemplo más común es la obtención de un error total pequeño). En el proceso conocido como *regresión,* el ajuste (alineación) a los parámetros se basa en el valor de la función de mérito hasta que un umbral de valor pequeño se obtenga, y por ende, se produce un "mejor ajuste" (best-fit). A dichos parámetros se les conoce como *parámetros de mejor ajuste*.

Clustering, cuya alineación consiste en la separación de un grupo de datos en subgrupos (clusters), con el fin de que los datos en cada subgrupo sean "*similares*" entre ellos y "*no similares*" con el resto de subgrupos. Usualmente la similitud o proximidad es definida por una medida de distancia.

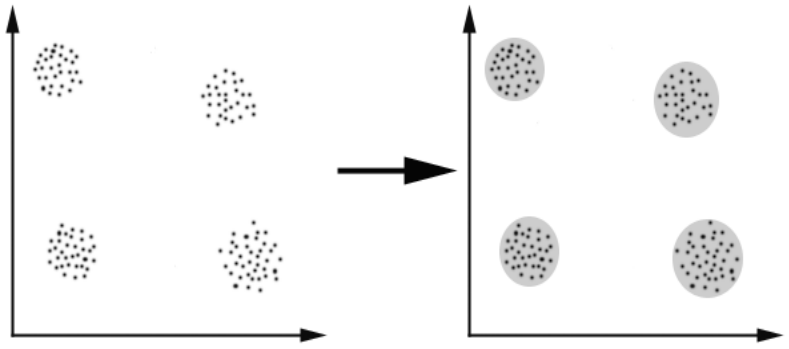


Figura 2.13. Ejemplo gráfico de *Clustering*.

La *votación,* es usada implícita o explícitamente, en la fusión de datos originada de múltiples fuentes, para obtener sistemas altamente confiables en el paradigma de computación multicanal (de varias modalidades, que pueden tener alineaciones diferentes). Su proceso de alineación, suele estar subdividido en alineación para el dominio del tiempo y alineación para el dominio espacial. Por ejemplo, en [TMR04] ante el análisis de múltiples entradas de imágenes de cámaras de video distintas, su alineación espacio-temporal consiste en: (1) analizar el mismo instante en toda imagen de entrada y (2) determinar por votación al área en movimiento (a la cual llaman *blob*); con lo anterior definen el centro de masa de la *blob* para así poder determinar a la trayectoria que siguió, e ilustrarlo en un mapa 2D.

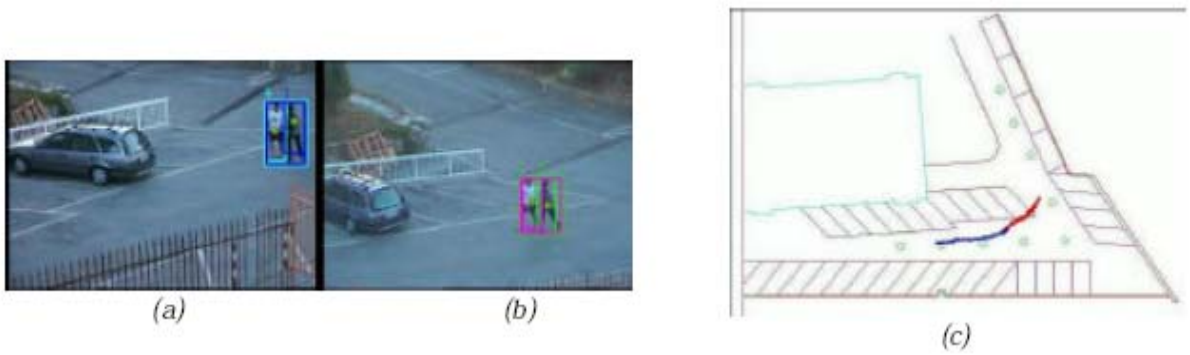


Figura 2.14. Estimación correcta de trayectorias en el mapa(c) usando múltiples vistas(a-b) en condición de oclusión.

Entropía, fue introducida en el contexto de cuantificación de información por C. Shannon (1948), con la noción de modelar la información como un proceso estadístico. Un método de alineación consiste en maximizar la información mutua o minimizar la entropía conjunta. La alineación con entropía es usual en el procesamiento digital de imágenes, ya que se le interpreta como el contenido promedio de información, que no es afectado por el tamaño de la imagen ni el patrón de grises que emplee, pero sí por la *frecuencia* de cada nivel de gris. Una limitación en la alineación con entropía, es que interpreta al ruido como parte de la información real, ya que también es energía [LVK01].

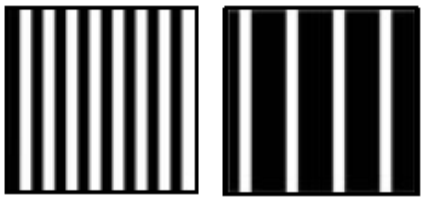


Figura 2.15. Ejemplo de entropía en procesamiento digital de dos imágenes binarias de 16X16 pixels.
(a) Al ser misma frecuencia, presenta entropía de 1 bit de información/pixel.
(b) Al ser frecuencia distinta, presenta entropía de 0.81 bit de información/pixel.

Álgebra de imágenes, consiste en imágenes (usualmente en dominio de tres-dimensiones, en lugar de un solo plano), *templates* (imágenes cuyos pixels son imágenes), operaciones matemáticas (suma, resta, multiplicación, etc.), el grupo de valores posibles para imágenes (simbolizado con **F**) en el dominio espacial de imágenes (simbolizado con **X**) y tipos de coordenadas (cartesianas, polares, etc.). Tiene la finalidad de establecer un marco de trabajo formal que implique operaciones inducidas avaladas para establecer los criterios de alineación formales, como antesala para la toma de decisiones [Rit99].

Máxima verosimilitud, error cuadrático medio: inferencia estadística - estimación.
Considerando la alineación como inferencia estadística conlleva establecer que, con el método de *máxima verosimilitud*, los datos bajo análisis son muestras aleatorias de cierta función de probabilidad de población que presenta un parámetro desconocido y sólo es posible estimar al parámetro de la distribución mediante la obtención de su valor máximo en la función de probabilidad de las muestras aleatorias. Un ejemplo típico es la estimación de la media poblacional en una distribución normal: la media estadística de la muestra es la estimación de la media poblacional, ya que proporciona la máxima verosimilitud.
El *error cuadrático medio* establece un criterio de comparación entre estimadores, consiste en la suma de la varianza de la estimación con el sesgo al cuadrado de la estimación.

Mínimos cuadrados: regresión.
Además de ser una opción para la alineación de datos, la regresión es un método ampliamente usado para estimar analíticamente a la relación entre los datos esperados de la variable dependiente *y* (también llamada *variable aleatoria*), ante la variación de la variable independiente *x* (también llamada *variable controlada*).
En el análisis de regresión, la dependencia de los valores de *y* ante los valores de *x* es una dependencia de la media μ de *y* en *x*, tal que $\mu=\mu(x)$ sea una función en el sentido ordinario, la cual también es conocida como *curva de regresión*. El caso más simple es la regresión lineal, ya que la relación entre la variable dependiente y la variable independiente es una función lineal, donde la curva de regresión es una *línea recta*: $\mu(x)=\kappa_0+\kappa_1 x$.
El método de *mínimos cuadrados,* también conocido como mínimos cuadrados de Gauss, es ampliamente usado (no sólo en alineación) y establece el siguiente principio: *La línea recta debería ser ajustada a través de los puntos dados tal que la suma de los cuadrados de las distancias de dichos puntos a la línea recta sea mínima, donde la distancia es medida en la dirección vertical (la dirección del eje-y).*

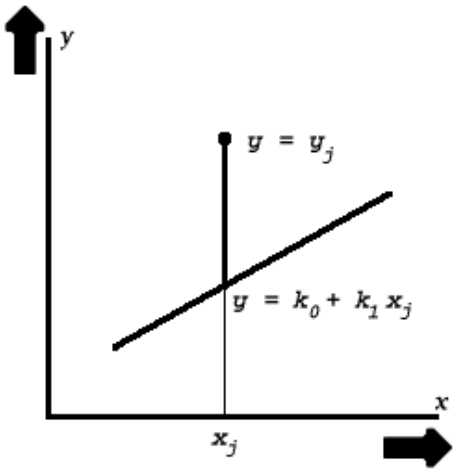


Figura 2.16. Representación gráfica de la distancia mínima.

Con el enunciado anterior, a través de una recta obtenida de la muestra analizada, se busca minimizar la distancia *y*-vertical:

$$\left|y_j-(k_0+k_1 x_j)\right|$$

La suma de los cuadrados de sus distancias es:

$$q=\sum_{j=1}^{n}(y_j-k_0-k_1 x_j)^2$$

Con lo cual, mediante cálculo diferencial, se busca minimizar a *q*,

$$\frac{\partial q}{\partial k_0}=0,\frac{\partial q}{\partial k_1}=0$$

Con estas condiciones, para la muestra de datos se obtiene la fórmula siguiente [Kre99]:

$$y-\bar{y}=k_1(x-\bar{x})$$

donde: $\bar{y}=\frac{1}{n}(y_1+\ldots+y_n), \bar{x}=\frac{1}{n}(x_1+\ldots+x_n)$ ...(medias de *x*, *y*).

$$k_1 = \frac{n\sum x_j y_j - \sum x_i \sum y_j}{n(n-1)s_x{}^2} \quad \text{...(coeficiente de regresión).}$$

$$s_x{}^2 = \frac{1}{n-1}\left[\sum_{j=1}^{n} x_j{}^2 - \frac{1}{n}\left(\sum_{j=1}^{n} x_j\right)^2\right] \quad \text{...(varianza de los valores } x\text{).}$$

A continuación el *ejemplo 2.B*, muestra una implementación para el método mencionado:

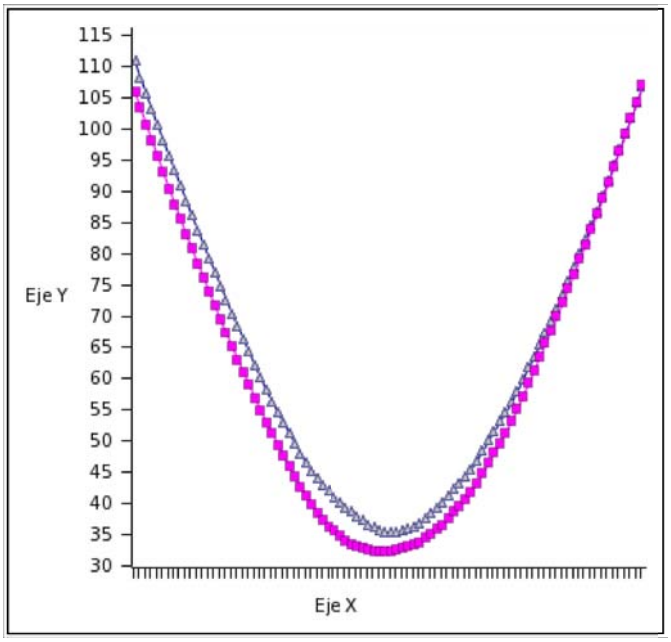


Figura 2.17. Ante una correcta definición de la variable independiente, es posible obtener a la variable dependiente de acuerdo a la naturaleza de la regresión lineal. En este ejemplo, se observan dos muestras de datos, que cuentan con la misma variable independiente (eje X); se busca la regresión lineal de la diferencia entre las dos muestras, por lo cual dicha diferencia toma el lugar de la variable dependiente, para así posteriormente generar a su recta de regresión lineal, como se aprecia en esta figura.

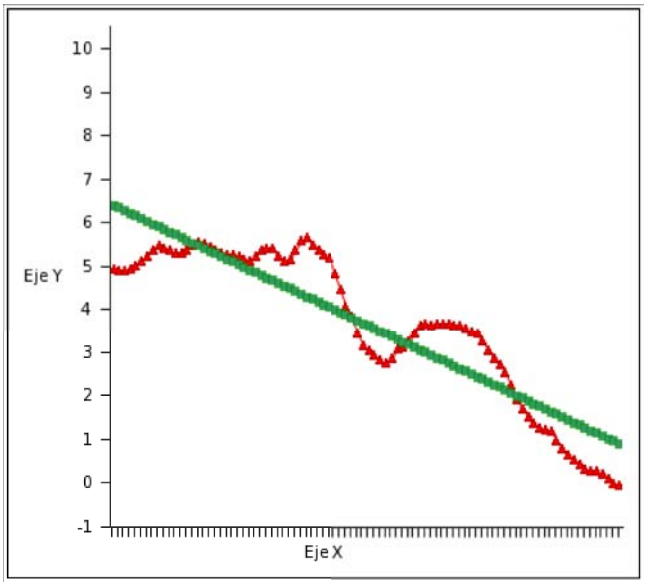


Figura 2.18. Mediante la diferencia de las dos muestras observadas en la *fig.2.17*, se genera la variable dependiente apta para la regresión lineal. De acuerdo al método analítico descrito en la *fig.2.16*, la recta genera a la mínima suma de los cuadrados de distancias verticales entre dicha recta y la curva generada por la diferencia entre las muestras originales.

Como se puede observar, se profundizó al estudio matemático en la regresión por mínimos cuadrados; se debe a que una implementación de dicho método también intervino en nuestro caso, aunque no en la alineación geométrica, pero sí en la *medición del error* en el ajuste robusto de modelos elipsoidales, siendo la variable de salida las diferencias entre los *campos de distancia*[6] correspondientes.

6 Consulte la sección 4.5 para obtener detalle sobre los campos de distancia.

2.4 Ajuste robusto de modelos elipsoidales.

La obtención de la elipsoide representativa para cada modelo de cabeza disponible, tiene como justificación un ajuste robusto e iterativo con bases analíticas, que hace uso de un esquema homogéneo para validar la comparación y las modificaciones necesarias entre modelos.
Como se mencionó en la sección 2.1, uno de los detalles principales en el método, es el trato especial a los puntos alejados (*outliers*) de los modelos de cabeza, donde un outlier se refiere a un punto que sobrepasa al criterio de tolerancia establecido por la elipsoide representativa. Outliers se hacen presentes primordialmente en nariz, mentón y orejas, como se aprecia en la *fig.2.19*.

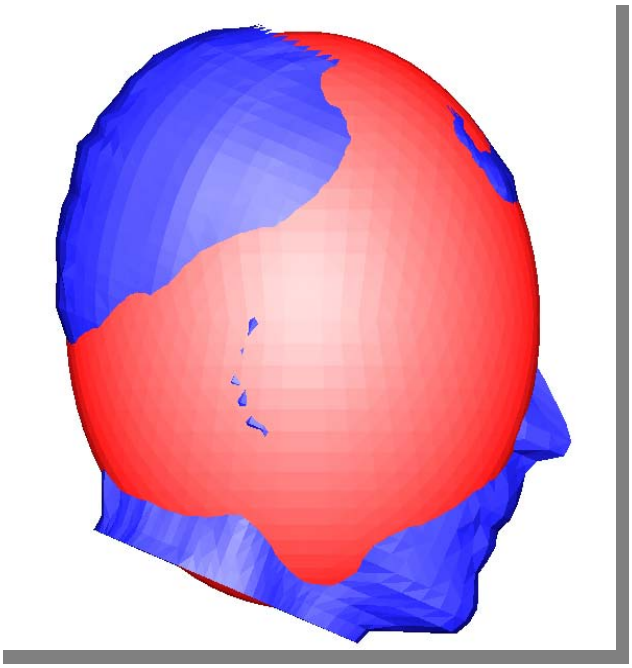


Figura 2.19. Ejemplo de outliers en una iteración del ajuste robusto al modelo cabeza03.pi.

Como punto inicial, el ajuste robusto de modelos elipsoidales toma en cuenta al factor crucial para la alineación geométrica de los modelos: el tensor de inercia. Cabe señalar que, el tensor de inercia es extremadamente sensible a la modificación de puntos en los modelos; es decir, refleja cambios considerables en la alineación del modelo con la simple modificación de posición de un punto del modelo, y más aún con descartarlo. En este caso dichos puntos son los puntos alejados (outliers). Mediante PCA la alineación geométrica permite simplificar al tensor de inercia, al hacer uso de dos transformaciones en todo el modelo: traslación y rotación. La simplificación reduce la cantidad de variables resultantes, siendo en total tres parámetros: los valores principales, que fueron empleados para generar a los modelos elipsoidales. Es factible recordar que una de las metas fue la comparación morfológica de los modelos, por lo cual en lugar de emplear a la *elipsoide de inercia* para los ajustes morfológicos, se emplearon *elipsoides homogéneas* que presentaran tensores de inercia tendientes al correspondiente tensor en los modelos de cabezas; además, la masa total considerada en todo modelo fue unitaria, con la finalidad de poder generar a la elipsoide representativa de la población de los modelos de cabezas.

Con lo anterior, el nivel conceptual del ajuste robusto consiste en:

1. Alinear geométricamente al modelo cabeza, mediante PCA.
   Las transformaciones geométricas usadas en la alineación de cada modelo no le provocan alteración, ya que se aplican uniformemente a todo el modelo.
2. Obtener los eigenvalores del tensor de inercia para poder generar a la primer versión de la elipsoide homogénea representativa.
   Al tener alineado al modelo de cabeza, los eigenvectores son afines al sistema de referencia, lo cual causa el no requerir mayores parámetros para la obtención de los eigenvalores que permitan crear a la elipsoide homogénea.

3. Generar la elipsoide homogénea con nivel discreto 1, considerando a los valores a, b, c de la elipsoide como los eigenvalores del punto 2.
   Con el fin de respetar la relación analítica entre el tensor de inercia y la elipsoide homogénea correspondiente, el nivel discreto inicial es 1, evitando la intervención de parámetros adicionales.
4. Aplicar transformación de escala a la elipsoide homogénea para obtener una elipsoide homogénea equivalente con eigenvalores tendientes a los del modelo cabeza en su tensor de inercia, de acuerdo a un criterio de tolerancia. Con ello, es posible generar a la elipsoide equivalente con mayor nivel discreto (se empleó al nivel 25).
5. Realizar la medición del error mediante el análisis a los campos de distancia de ambos modelos. La métrica del error consiste en la diferencia cuadrada entre dichos campos de distancia. Cabe señalar que es posible el uso de otros criterios de error, como el cálculo de patrones estadísticos (media, varianza, mediana, etc.).
   Cabe señalar que los campos de distancia permiten expresar a cada modelo como un patrón unidimensional, sin necesidad de modificar a los modelos, mejorando así el rendimiento de la medición del error cuadrático.
6. Aplicar la transformación necesaria a los puntos externos (es decir, trasladar a *outliers*) del modelo de cabeza con respecto a la elipsoide homogénea equivalente, con el fin de mejorar al modelo de elipsoide.
7. Repetir al proceso iterativo hasta que la medición de error deje de disminuir.

CAPÍTULO III

Ingeniería de Software

3.1 Requerimientos y especificaciones: Información, errores y formatos existentes.

3.2 Arquitectura de software Modelo-Vista-Controlador (MVC).

3.3 Control de Versiones (VC).

3.4 Herramientas de software a utilizar.

Todo es muy difícil antes de ser sencillo
Thomas Fuller

3.1 Requerimientos y especificaciones: Información, errores y formatos existentes.

Dada la evolución e incremento de importancia del *software* en actividades como la investigación científica moderna, se le ha asociado el tener un doble papel: ser un *producto* y de forma simultánea ser el vehículo de entrega de *otro producto,* que muchos estiman será el más importante del siglo XXI, la *información* [Pre98]. El cómo lograr que el software cumpla el papel citado, llega a ser resumido con la definición del *proceso* del software: el esquema de trabajo para la definición del producto, o en otras palabras, un conjunto de etapas parcialmente ordenadas con la intención de lograr la obtención de un producto de software de calidad. Para establecer al modelo que represente al proceso de la ingeniería de software, es conveniente establecer la definición y especificación de requerimientos [Som92], que para esta tesis, se muestra a continuación:

Definición de requerimientos funcionales.
Todo modelo tridimensional resultante debe mantener la misma estructura poligonal de los modelos de cabezas originales.

- Únicamente se permite emplear a triángulos como polígonos.
- No deben presentarse triángulos o vértices concurrentes.
  - Es necesario eliminar redundancia de vértices porque el tensor de inercia es muy sensible a la densidad de puntos, lo cual causaría momentos de inercia que reflejarían cambios morfológicos significativos con preferencias en las regiones de vértices redundantes.

Es necesario contar con un programa visor de modelos tridimensionales, que debe:

- Contar con interfaces de entrada estándar para el usuario (ratón y teclado), así como información de ayuda para dichas interfaces.
- Desplegar a uno o dos modelos 3D, de forma individual o simultánea.
- Desplegar la información geométrica y mecánica relacionada, del modelo 3D.
- Proporcionar la capacidad de ejecutar la alineación geométrica de los modelos 3D.
- Crear la *elipsoide homogénea* correspondiente a un modelo 3D bajo análisis, de acuerdo a sus valores de *ejes principales*.
- Brindar la opción de almacenar los cambios hechos al modelo 3D.
  - Guardar la nueva instancia del modelo 3D, usando el formato original.
  - Exportar la información geométrica y mecánica en un formato estándar.

Es necesaria la presencia de un programa que realice la comparación morfológica, con el cual se permita obtener la *estimación del error* entre cada modelo de cabeza 3D y su elipsoide homogénea representativa, por lo cual debe:

- Operar con el mismo formato de modelo 3D que emplee el programa visor.
- Emplear el criterio de comparación provisto por los *campos de distancia.*
- Usar a la métrica de distancia euclidiana.
- Almacenar, en un archivo de texto, a las distancias resultantes de la comparación entre los dos modelos.
- Crear una imagen que ejemplifique gráficamente a las distancias obtenidas en el punto anterior.

Es necesaria la presencia de herramientas que permitan realizar el ajuste de los modelos de cabezas 3D, en cada iteración, para mejorar a su elipsoide representativa, por lo cual deben:

- Implementar los criterios de ajuste para los *puntos externos o alejados (outliers)* en los modelos de cabeza 3D.
- Los criterios deben preservar al mallado del modelo (su topología: conexiones y puntos vecinos), lo cual implica que "eliminar outliers" se interpreta como *proyectar (trasladar)* dichos vértices al modelo de tolerancia (la elipsoide homogénea equivalente).

Definición de requerimientos no funcionales.

- Todo avance del proyecto (desarrollo de software o documentación) debe ser hecho sobre el *sistema operativo Linux*, con compatibilidad en distintas de sus distribuciones.

El formato del modelo tridimensional debe ser texto plano (ASCII) con sintaxis y semántica simple; su contenido debe describir a la información tridimensional del modelo que almacena (sus puntos e índices) y comentarios relacionados con el modelo.

- Todo polígono en el modelo esta formado por tres vértices, los cuales enumerados en sentido antihorario, determinan la dirección de su vector normal unitario.
- El formato original de los modelos (VRML ver.1.0) sufre cambios externos (por ejemplo, ya existe VRML ver.2.0), por lo cual se decidió crear un formato interno, de fácil exportación a VRML o similares, que cumpla con los requerimientos.
- El formato interno resultante se denomina **PIF** (*Point~Index~Format*), el cual almacena al modelo tridimensional en un archivo con dos regiones: Puntos e Índices.
- PIF permite almacenar a los comentarios relacionados antes de la información tridimensional, mediante un símbolo que indique el ignorar la línea de comentario.
- Se almacena en un archivo de texto plano, con código de caracteres ASCII, que debe tener la extensión "**.pi**" en el nombre del archivo, por ejemplo: *cabeza10.pi*.

Dentro de los requerimientos funcionales, se percibe la necesidad de desarrollar un programa visor en lugar de adquirir alguno disponible, ya que debe cumplir con requerimientos particulares que implementen la alineación geométrica diseñada en el proyecto[7]. Con ello, ante el análisis de recursos computacionales disponibles, se propuso el uso de OpenGL o equivalente, ya que cuenta con los siguientes beneficios [Hal00]:

- El origen de OpenGL fue para fines científicos e investigación.
- Proporciona un API robusta, con potencial para desarrollo de software de graficación tridimensional en tiempo real.
- Aprovecha ampliamente la funcionalidad disponible en el hardware 3D (tarjetas de aceleración gráfica).
- Es multiplataforma, y además ofrece la posibilidad de implementación de *clones OpenGL* compatibles, sin problemas legales.
- Es un estándar abierto y maduro, lo cual le brinda constante evolución y soporte.
- Cuenta con una librería auxiliar que permite acelerar el desarrollo de las interfaces con el usuario: *GLUT*.

El resto de requerimientos funcionales fue asimilado por el análisis, diseño y desarrollo del núcleo de los programas: abarca el diseño formal de la funcionalidad requerida, la creación del *modelo* del proyecto. El enunciado anterior consistió en diseñar los métodos matemáticos (detallados en el capítulo 4) implementados con el patrón de diseño MVC.

7 Al inicio, se utilizaron visores de dominio público: VRWave (http://www.iicm.edu/vrwave), VRWeb (http://www2.iicm.edu/vrweb), OpenVRML (http://openvrml.org/), entre otros.

3.2 Arquitectura de software Modelo-Vista-Controlador (MVC).

Dentro de la evolución del desarrollo de software se ha establecido una abstracción del software que lo divide en módulos (capas): una estructura o estructuras que comprenden los elementos del software, las propiedades externas visibles de estos elementos, y las relaciones entre ellos, es decir, la *arquitectura de software* [Bas04]. Dentro del análisis de requerimientos, se decidió usar MVC, que es una arquitectura de software que separa a las *capas* (layers) del software desarrollado en tres módulos, lo cual facilita el mantenimiento de las posibles variaciones de interfaces con las cuales interactúe dicho software [Dea05]:

- *Modelo* (*Model*). Representación de la información diseñada para la aplicación: idealmente considerada como la esencia de la aplicación, sin cambios, que no tiene contacto directo con el usuario. En este caso, los datos (vértices y triángulos) se representan como *PIF*, siendo toda la funcionalidad interna y formato de datos definidos para los modelos tridimensionales.
- *Vista* (*View*). Presentación del modelo en formas aptas para el usuario, es decir, las interfaces de despliegue necesarias. En este caso, se establecen tres vistas: API de OpenGL o equivalente, visor de modelos tridimensionales como GUI, y la consola de comandos del sistema operativo (CLI) como interfaz de comandos para obtención de campos de distancia.
- *Controlador* (*Controller*). Interfaz de entrada que le permite al usuario reflejar a los eventos de entrada en el modelo y así obtener los resultados esperados en las vistas correspondientes, por ejemplo, la alineación geométrica de un modelo tridimensional. En este caso, el controlador interpreta la interacción del usuario a través de dispositivos de entrada estándar.

Como se indica en las definiciones previas, la arquitectura MVC causó la separación de (1) la lógica funcional y los datos, de (2) la forma en cómo se le despliega al usuario, de (3) la forma en cómo la controla el usuario,

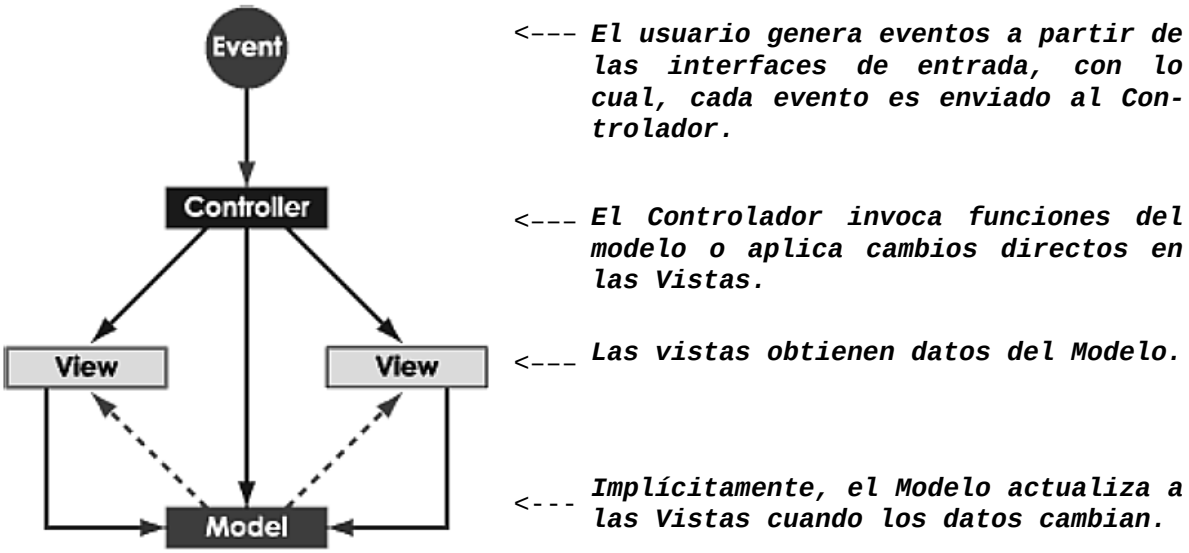


Figura 3.1. Diagrama de la arquitectura de software MVC
http://en.wikipedia.org/wiki/Image:Mvc.gif
(Imagen de dominio público)

PIF (Modelo)

- Point~Index~Format – Estructura, formato y lógica del modelo tridimensional que permita crear operaciones de transformación y modificación de acuerdo a los objetivos establecidos, haciendo uso de datos con una sintaxis y semántica simple.
  - Transformaciones 3D para la alineación geométrica.
  - Transformaciones 3D para la aproximación geométrica a elipsoide.
  - Cálculo de campos de distancia.
  - Operación de traducción a VRML1, para permitir la visualización de los modelos resultantes en plataformas que soporten VRML1.

API-GUI-CLI (Vistas)

- API OpenGL equivalente – Distribución compatible en Linux: *The Mesa 3D Graphics Library (http://www.mesa3d.org/)*.
  - Representa a los métodos disponibles para invocar a los resultados visuales de las interacciones con el usuario.
- Visor PIF 3D – GUI que permite desplegar a los resultados de operaciones hechas a modelos tridimensionales, aprovechando la tecnología de cómputo: tanto de software (API de OpenGL o equivalente) como de hardware (Acelerador 3D GPU o equivalente).
- Consola (CLI) – Despliega información resultante en modo texto (por ejemplo, la ubicación de imágenes resultantes en un campo de distancia), así como el avance que paulatinamente reporta el modelo.

Intérprete de eventos (Controlador)

- Emplea *GLUT callbacks,* con la finalidad de indicar a la vista GUI el despliegue de información solicitada.
  - Cuenta con una innata comunicación de eventos con la vista API y por ende, con la vista GUI.
- Traduce a los eventos que ingresa el usuario (con dos dispositivos físicos comunes: teclado y ratón) mediante funciones particulares en cada vista.

3.3 Control de Versiones (VC).

A pesar de la presencia de un análisis de requerimientos y especificaciones previo al desarrollo e implementación del software resultante (el conjunto de programas establecido en los requerimientos funcionales) es esperado que, ante la evolución del proyecto, se presenten *cambios:* ya sea durante su desarrollo, mantenimiento ó reemplazo. Con la finalidad de controlar a dicha evolución, existe la disciplina denominada *Administración de la Configuración de Software* (SCM), la cual cuenta con una definición estándar de la IEEE [FD90]: *SCM es el proceso de la identificación y definición de los componentes en un sistema, controlando su liberación (release) y cambios a través de su ciclo de vida, almacenando y reportando el estado de sus componentes y peticiones de cambio, y verificando que los componentes del sistema estén completos y correctos.* En el presente proyecto se dio énfasis únicamente al *control de versiones (VC)* de la SCM mediante el uso de una herramienta de Ingeniería de Software Asistida por Computadora (CASE): *Subversion (SVN),* con una interfaz gráfica para el usuario: *JSVN*.

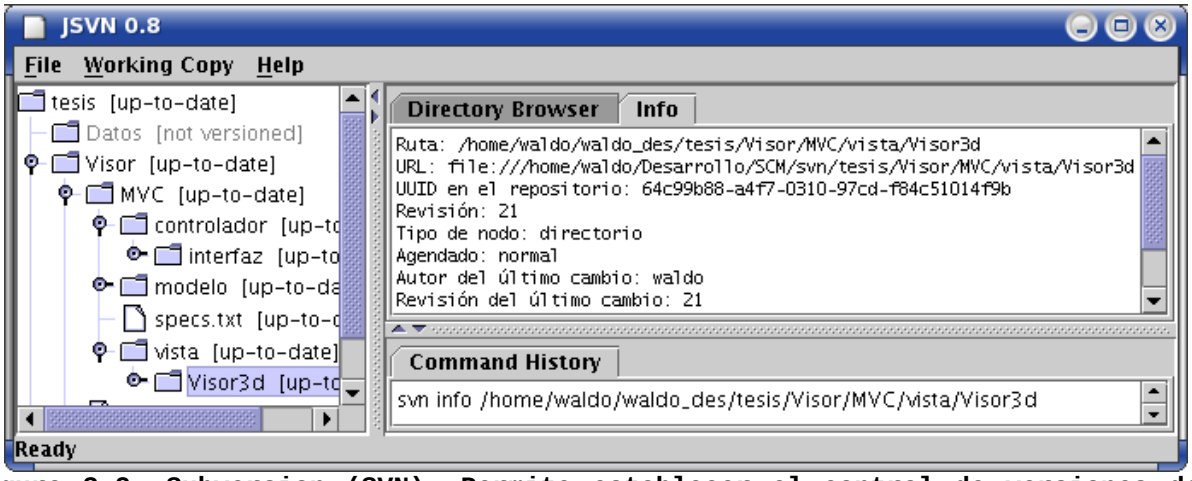


Figura 3.2. Subversion (SVN). Permite establecer el control de versiones del código fuente mediante *revisiones*; la interfaz gráfica JSVN facilita su uso. SVN (http://subversion.tigris.org) y JSVN (http://jsvn.alternatecomputing.com) se distribuyen bajo *licencias de código-abierto (open-source)*.

SVN es un sistema de control de versiones que crea un *repositorio* con el cual administra a archivos y carpetas mediante el control de las *publicaciones (commits)* de cambios a lo largo del tiempo: a diferencia del usual sistema o servidor de archivos, el repositorio "recuerda" los cambios hechos a sus archivos y carpetas, lo que permite recobrar versiones previas o consultar su historia de avance. Antes de publicar los cambios en el repositorio, los usuarios responsables hacen sus modificaciones correspondientes en su *área privada de trabajo (working copy),* con lo que, cuando SVN acepta su publicación, crea automáticamente un nuevo estado del repositorio: una *revisión* (identificada con un número natural único, *fig3.3*); posteriormente, cada usuario puede actualizar su área privada de trabajo para recibir todo cambio publicado en el repositorio [CFP05].

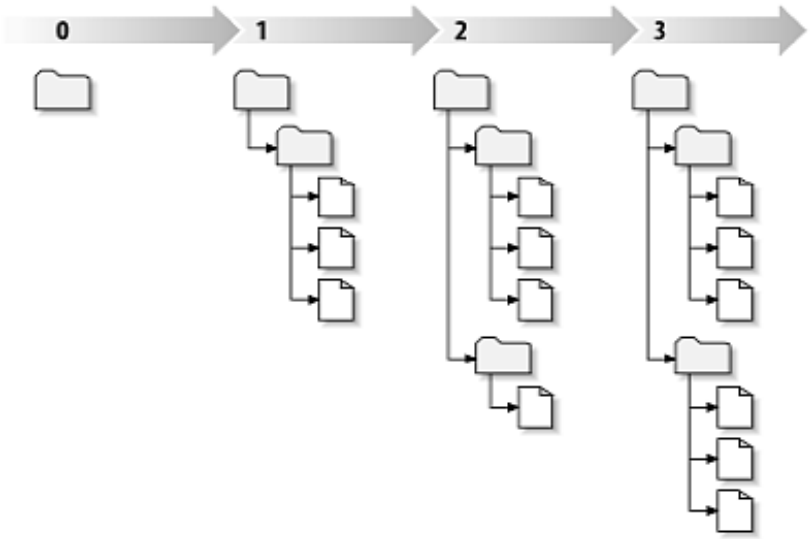


Figura 3.3. El repositorio. SVN aplica los números de revisión a todo el árbol del repositorio y no sólo a archivos individuales, con lo cual el decir *"revisión 5 de archivo.txt"* significa *"archivo.txt, como aparece en la revisión 5"*.

3.4 Herramientas de software a utilizar.

De acuerdo al flujo de eventos mostrado en el diagrama de la arquitectura de software MVC (*fig3.1*), a continuación se detalla información sobre las herramientas de software empleadas en el desarrollo de software.

GLUT (OpenGL Graphics Library Utility Toolkit, versión 3.7)[Kil97]
Además de contar con una innata relación con la GUI (ante la creación de ventanas de despliegue), para la interpretación de eventos de entrada por parte del usuario en el programa visor, se implementaron las siguientes características que GLUT proporciona:

- Implementación multiplataforma para sistemas de ventanas. Evita la necesidad de reprogramar el código correspondiente al controlador (la interpretación de los eventos de entrada), ya que es transparente para sistemas de ventanas compatibles (X-windows, MacOS, Win32).
- Compatible con lenguaje ANSI C.
- El procesamiento de eventos de basa en funciones de respuesta (*callback*).
- Permite desplegar en la ventana de vista un menú auxiliar de apoyo para el usuario.
- Permite implementar una función de respuesta a ausencia de eventos: rutina *idle*.
- Toda etiqueta de sus funciones cuenta con el prefijo "**glut**", por ejemplo:
  *void **glut**KeyboardFunc(void (*func)(unsigned char key, int x, int y));*

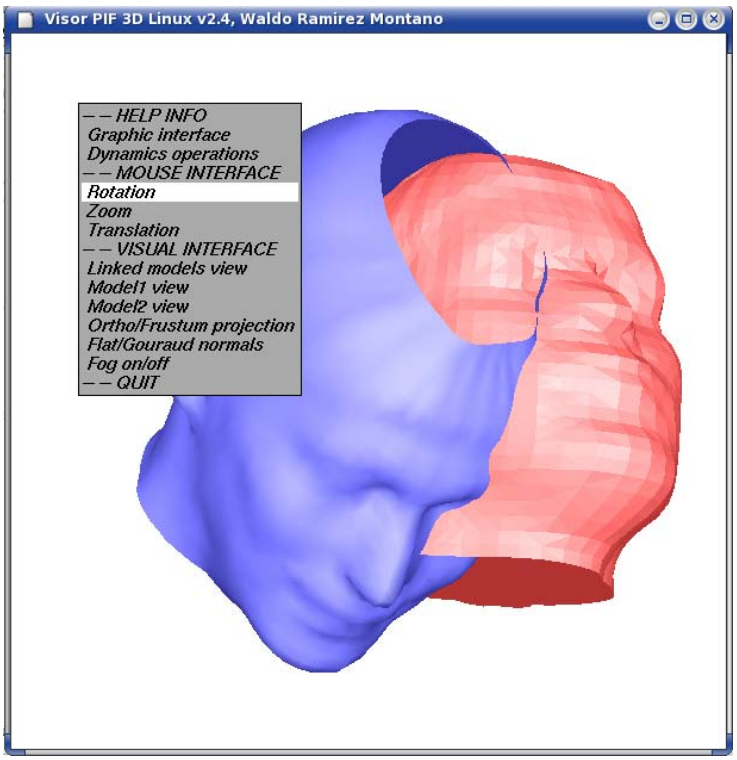


Figura 3.4. GLUT. Proporcionó el control de eventos de ratón y teclado. En este ejemplo, la respuesta (*callback*) al evento provocado por el tercer botón del ratón (usualmente el *botón central*), causa una actualización de la vista: despliega el menú de atajo para poder invocar a otros eventos.

OpenGL (Open Graphics Library, versión 1.5) [SA03]
OpenGL es un API para el hardware gráfico; consiste en una gran cantidad de procedimientos y funciones que permiten al programador especificar a los objetos y operaciones involucrados en la presentación (*rendering*) de imágenes de objetos 3D. Al ser Linux el sistema operativo requerido para la implementación del proyecto, se analizó la posibilidad de emplear un clon OpenGL que ofrezca beneficios funcionales y facilidades financieras. Como resultado del análisis, se presentó una excelente opción, *The Mesa 3D Graphics Library* [Pau93], *versión 6.0.1* (equivale a OpenGL versión 1.5).

- Es una implementación *open-source* de las especificaciones de OpenGL.
- Aprovecha la mayoría de controladores de hardware 3D disponibles, especialmente los ofrecidos por el proyecto DRI (también open-source), de Xfree86 [DRI99].
- El costo de usar Mesa es nulo, siempre y cuando se respete la licencia de uso ofrecida por su autor, Brian Paul.
- La única razón por la cual no puede ser llamado "OpenGL" es porque su autor no ha adquirido una licencia de *OpenGL ARB* (Architectural Review Board).
- En caso de no contar con la librería de GLUT, ofrece una distribución equivalente. Con ello, se completa la generación de las ventanas gráficas de despliegue de objetos 3D.

La obtención de la vista para el usuario se logra con la implementación de OpenGL para establecer el *Pipeline de Gráficos* (*Graphics Pipeline*[8]), lo cual depende del estado del modelo, y de los eventos de entrada: determinan las transformaciones 3D a ser aplicadas, tanto de modelado (rotación, traslación, escala), proyección (ortogonal o piramidal) y de espacio visible, como se detalla en el capítulo IV.

Herramienta CASE: SVN (versión 1.2.0)

- Diseñado para sustituir a *CVS (Concurrent Versions System),* el sistema de control de versiones más popular en software open-source.
- Controla versiones de archivos *y* carpetas, junto con sus metadatos.
- La publicación de cambios aplica a todo el repositorio.
- Permite su implementación *stand-alone*[9].
- El repositorio no exige un formato exclusivo, puede ser creado sobre el *sistema de archivos* del sistema operativo (en este caso *ext3,* el correspondiente a Linux).

Herramienta CASE: JSVN (versión 0.8)

- Cliente Java Swing que funciona como GUI para manipular al repositorio de SVN.
- Abstracción gráfica simple de la mayoría de comandos SVN.
- Su ejecución requiere Java SDK o RE de *Sun Microsystems* (se empleó *J2SDK 1.4.2_08*).

Compilador: GCC (versión 3.3.2)

- Usado para generar el código ejecutable de los programas desarrollados.
- Desarrollo en código de lenguaje C (ANSI C).

Herramientas de documentación

- *OpenOffice 1.1.0.* Suite de software de oficina creada originalmente por *Sun Microsystems* (http://www.openoffice.org). Es open-source, multi-plataforma y multi-lenguaje, que

8 Secuencia de procesos aplicados a los datos 3D originales con el fin de producir su imagen de despliegue (*rendered image*), véase apéndice C.

9 Característica de sistemas cuya ejecución es hecha 100% en una sola computadora, es decir, de forma local (sin hacer uso de red).

permite la creación de documentos de información con herramientas acordes a sus objetivos (documentos de texto, hojas de cálculo, presentaciones, etc.). Soporta formatos externos (por ej. Microsoft Office XP) y la estandarización XML.

- *Gnumeric 1.2.6.* Hoja de cálculo de alto rendimiento para gran cantidad de datos, mediante una interfaz simple y sencilla para el usuario. Permite la importación de datos presentes en archivos de texto ASCII y almacenar su análisis en formato Gnumeric XML (http://www.gnome.org/projects/gnumeric).
- *GIMP 1.2.* Programa GNU de distribución libre, empleado para la manipulación de imágenes (http://www.gimp.org). Permitió realizar la interpretación de formatos PNM de las imágenes correspondientes a los campos de distancia, así como su combinación.
- *ImageMagick 5.5.7.* Suite de software libre que permite crear, editar, y transformar imágenes digitales (http://www.imagemagick.org). Fue empleado para la aplicación de escala homogénea a imágenes PNM y posteriormente convertirlas a JPEG; con ello permitió crear la base de GIFs animados de los campos de distancia.
- *Multivalent 8.3.* Herramientas diseñadas para el proceso de documentación digital (http://multivalent.sourceforge.net/) en formato PDF, entre otros.
- *LiVES 0.9.5-pre5.* Sistema de edición de video de distribución gratuita (Linux Video Editing System, http://www.xs4all.nl/~salsaman/lives/). De entre todas las características multimedia del sistema LiVES, permite la captura de video, su edición y su almacenaje en formatos estándar.
- *MJPEG Tools 1.8.0.* Herramientas de video digital (http://mjpeg.sourceforge.net/) que permiten almacenar y procesar a flujos de video en formato MPEG estándar. Tiene compatibilidad de uso con en el sistema LiVES.

Cabe recordar que el sistema operativo sobre el cual se desarrolló, implementó y probó a todo el software resultante, fue una distribución del sistema operativo Linux (*Mandrake 10.0, kernel 2.6.3*).

Con la plataforma elegida se presentaron resultados satisfactorios y 100% transparentes, ya que la ejecución de los programas compilados en distribuciones de Linux distintas, no presentaron conflictos:

- Red Hat 7.0.
- Fedora 3.0.

Lo anterior incluye a distribuciones *Linux LiveCD*[10]:

- CDMedic 6 (http://cdmedicpacsweb.sourceforge.net/cdmedic_en.html).
- Knoppix 3.9 (http://www.knopper.net/knoppix/index-en.html).
- SLAX5.0.5 (http://slax.linux-live.org/).

10 Distribución especial de Linux, orientada para operar *directamente* desde algún dispositivo extraible (usualmente un CD); es decir, no requiere la instalación del sistema operativo en el disco duro de la computadora.

CAPÍTULO IV

Diseño de los programas

4.1 Diseño de estructuras de datos.

4.2 Transformaciones geométricas en 3D.

4.3 Modelos simplificados.

4.4 Alineación geométrica iterada y robusta.

4.5 Campos de distancia euclidianos.

4.6 Diseño del proceso iterativo de ajuste de modelos.

4.7 Estimación de errores morfológicos.

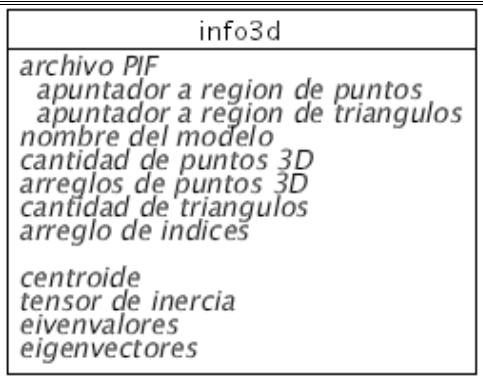
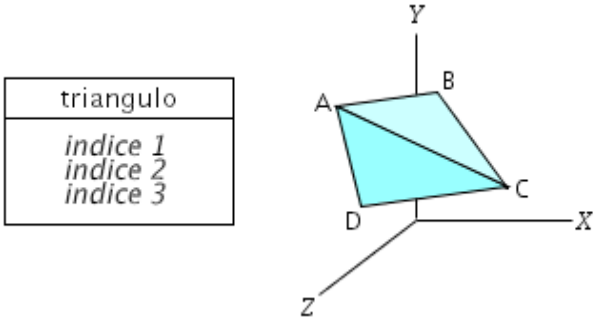
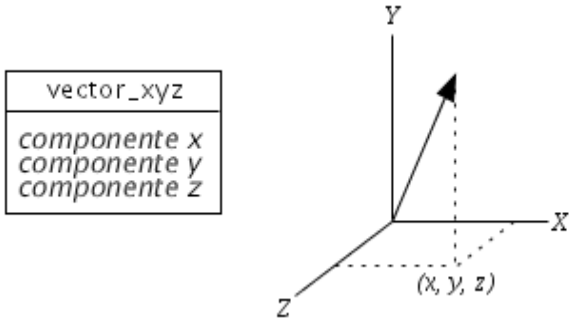
Aprendiendo a morir se aprende a vivir mejor
Platón

4.1 Diseño de estructuras de datos.

A pesar de que la arquitectura de software tiende a ser considerada como parte del diseño de software, también se mencionó a MVC en el proceso de análisis con el fin de establecer el contexto conceptual y funcional para satisfacer a los requerimientos especificados (véase sección 3.1). A continuación, el diseño de datos encabeza al diseño formal: las estructuras de datos establecidas para ser parte de la capa modelo.

- *Estructuras de datos para definir y manipular a un modelo tridimensional.*

Contiene a toda instancia de datos, tanto de tipo estático (por ejemplo, variables de tipo entero) como dinámico (por ejemplo, cadenas de caracteres dinámicas), necesaria para establecer la información que identifica a un modelo 3D. Para ello, se establecieron cuatro estructuras de datos: *información 3d, triángulo, vector xyz y ecuación 3D*.

Descripción	*Modelo de datos y algunos ejemplos*
Información 3D. Contiene a todo el conjunto de datos cruciales para definir a un modelo 3D: archivo de almacenaje, datos de los puntos y polígonos que lo componen, así como sus datos mecánicos (tensor de inercia, eigenvalores, eigenvectores y centroide).	
Triángulo. Establece al único polígono permitido para construir la superficie de un objeto tridimensional; esta compuesto por índices que señalan qué puntos componen al polígono. Como se ilustra en la *fig.4.1,* no es válida la repetición de índices en un mismo triángulo, lo cual requirió de una función de reducción de redundancia de índices.	
	Figura 4.1: Ejemplo con dos triángulos (Ia, Ic, Ib), (Id, Ic, Ia).
Vector xyz. Para el espacio tridimensional real, almacena a los valores numéricos que definen al vector 3D correspondiente; útil en la creación y manipulación de vectores usados en el despliegue gráfico, así como en la funcionalidad analítica junto con la estructura *ecuación 3D*.	
	Figura 4.2: Ejemplo de un vector y sus componentes.

Ecuación 3D. Define analítica y canónicamente a los modelos 3D simplificados (elipsoides) y auxiliares (planos) usados en este proyecto. Los atributos "grado" indican qué exponente acompaña a la expresión canónica de cada variable (para elipsoides: 2.0, para planos: 1.0), en donde también interviene el coeficiente correspondiente a cada variable.
En la *fig.4.3* se aprecia un ejemplo de elipsoide con vector origen distinto de *(0, 0, 0)*,

$$\frac{(x-x_0)^2}{a^2}+\frac{(y-y_0)^2}{b^2}+\frac{(z-z_0)^2}{c^2}-1=0\,.$$

En la *fig.4.4* se observa como ejemplo a un fragmento de plano, en donde se indica su vector normal (en sentido contrario al reloj),

$$ax+by+cz+d=0\,.$$

ecuacion3d
grado en x *grado en y* *grado en z* *coeficientes* *vector normal* *vector origen*

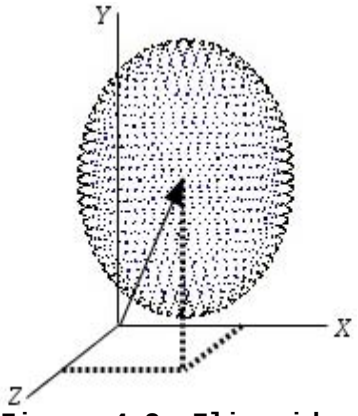


Figura 4.3. Elipsoide.

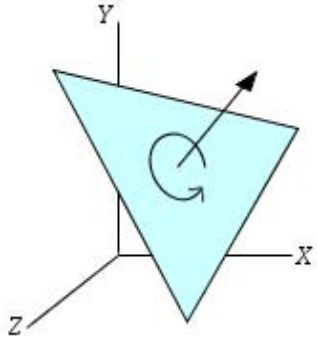


Figura 4.4. Plano.

Tabla 4.1. Estructuras de datos para definir y manipular a un modelo 3D.

Cada instancia de modelo 3D cuenta con una estructura física almacenada en un archivo de texto en código ASCII, al cual denominamos *archivo en formato PI ó PIF*,

- *Estructura de un archivo PIF (Point~Index~Format):*

```
#PIF model file    ---> Primer línea del archivo PIF.
# COMENTARIO 1           El símbolo '#' (número) indica que la línea de
#     ...                texto es un comentario.
# COMENTARIO M
X01   Y01   Z01    ---> Primera región: puntos (X, Y, Z). Indican los p
X02   Y02   Z02         puntos (vértices) que pertenecen al modelo.
X03   Y03   Z03
      ...
Xp    Yp    Zp
~                  ---> El símbolo '~' (tilde) indica fin de región.
Ia    Ib    Ic     ---> Segunda región: índices (I). Indican los n
Id    Ie    If          triángulos del modelo a través de la indexación
Ig    Ih    Ij          de los p puntos.
      ...
Ir    Is    It
~                  ---> El símbolo '~' (tilde) indica fin de región.
```

X_p, Y_p, Z_p son números reales con 10 números decimales como máximo.
$I_\alpha, I_\beta, I_\gamma$ son números enteros que indexan a los 3 puntos que forman a un triángulo, por lo cual, su valor debe ser menor que la cantidad de puntos, y además: $\alpha \neq \beta \neq \gamma$.

```
#PIF model file
# Ellipsoid level-1: 6 points, 8 triangles.
0.0000000000 99.5680000000 0.0000000000
50.5000000000 0.0000000000 0.0000000000
0.0000000000 0.0000000000 44.6666900000
-50.5000000000 0.0000000000 0.0000000000
-0.0000000000 0.0000000000 -44.6666900000
0.0000000000 -99.5680000000 0.0000000000
~
0 2 1
3 2 0
0 4 3
1 4 0
1 2 5
5 2 3
3 4 5
5 4 1
~
```

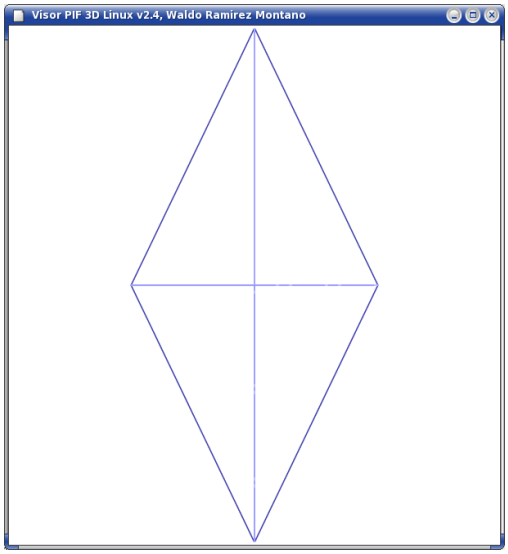


Figura 4.5. Ejemplo de elipsoide nivel-1 en su archivo PIF.
Izquierda: archivo PIF. Derecha: despliegue en Visor3D, modo wireframe.

A su vez, diseñamos un esquema *XML* que permite almacenar la información geométrica y mecánica de una instancia de modelo 3D PIF, lo cual permite el usar una plantilla *XSL* para desplegarle como *HTML*[11] en navegadores estándar de Internet.

- *HTML resultante del esquema XML interpretado con la plantilla XSL* ***information.xsl,***

3D Model Information.

File: **"./Ejemplo_.pi"**
Number of points: **6**, number of triangles: **8**.

Geometric centroid (x, y, z):
(0.0000000000, 0.0000000000, 0.0000000000)

Eigenvalues.	*Eigenvectors.*
(1) 6099.9505341142.	**(1)** (1.0000000000, 0.0000000000, 0.0000000000).
(2) 7921.5762640260.	**(2)** (0.0000000000, 1.0000000000, 0.0000000000).
(3) 6925.4255187601.	**(3)** (0.0000000000, 0.0000000000, 1.0000000000).

Inertia Tensor:
6099.9505341143 -0.0000000000 -0.0000000000
-0.0000000000 7921.5762640261 -0.0000000000
-0.0000000000 -0.0000000000 6925.4255187599
PCA: "abc" values:

Inertia Ellipsoid.	*Homogeneous Ellipsoid.*
(1) 0.012803739907193.	**(1)** 147.877070980188705.
(2) 0.011235546362617.	**(2)** 112.957954443769836.
(3) 0.012016466181782.	**(3)** 133.192541827423923.

Figura 4.6. Ejemplo XML→HTML de información de un modelo 3D PIF. Consulte al apéndice D.1 para detalle en los formatos XML, XSL diseñados.

11 HTML (HyperText Markup Language), comúnmente usado para la creación de paginas web en Internet, es un estándar multiplataforma que aseguró la compatibilidad de despliegue requerida.

• *Estructuras de datos para el ajuste y medición de error entre modelos tridimensionales.*

De igual manera, fue necesario diseñar estructuras de datos involucradas en el ajuste de modelos, para implementar la obtención de los campos de distancia euclidianos (véase sección 4.5, elementos fundamentales para el criterio de comparación entre modelos): *triángulos 2D y filtro de puntos*.

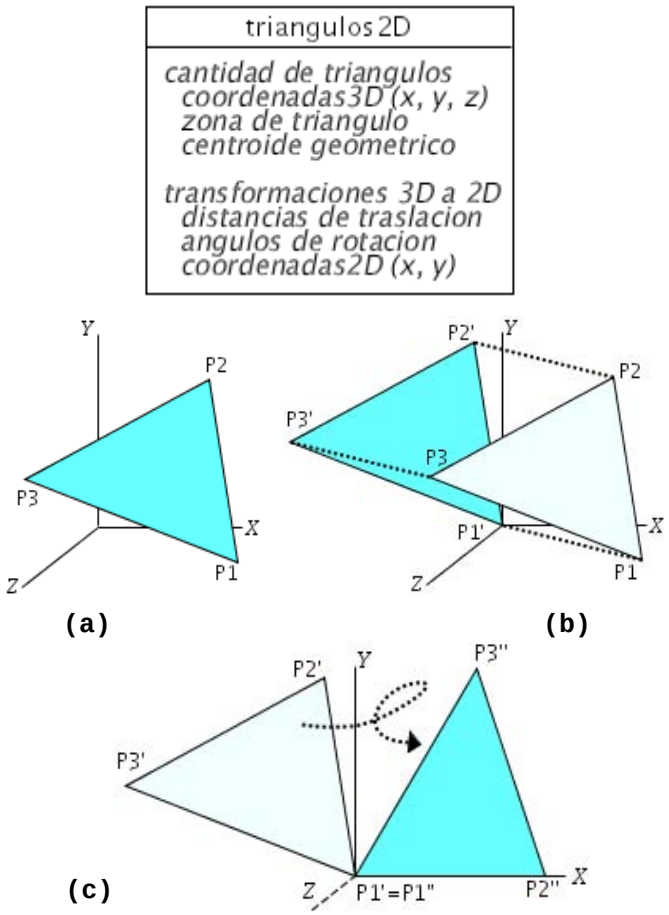
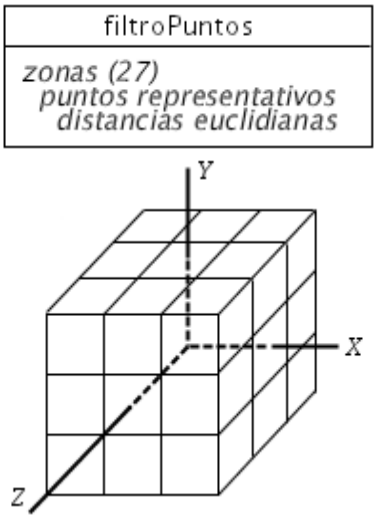
Descripción	*Modelo de datos y algunos ejemplos*
Triángulos 2D. Cada instancia contiene la información referente a todo triángulo del modelo bajo análisis, con el fin de interpretar a sus triángulos en 2D (considerando a $z=0$), que es fundamental en el algoritmo que diseñamos para la obtención de campos de distancia. Con el fin de poder obtener a la menor distancia euclidiana de todo triángulo con respecto a otro punto en el espacio 3D, a este último también se le deben aplicar las transformaciones pertinentes, y por ende, esta estructura de datos incluye la bitácora de transformaciones que causan la reducción de dimensión en triángulos: $(x, y, z) \rightarrow (x, y)$.	
Figura 4.7. Ejemplo de triángulo 3D a 2D. ***(a) Triángulo 3D original,*** ***(b) Traslación P1 a origen,*** ***(c) Triángulo 2D (posterior a rotaciones en X, Y, Z).***	
Filtro de puntos. En la obtención de los campos de distancia, suele destacar la considerable cantidad de tiempo que requiere [SOM04], por lo cual ésta estructura organiza a los puntos del espacio 3D bajo análisis en 27 zonas; lo anterior causa reducción en la cantidad de operaciones y comparaciones necesarias para obtener las distancias buscadas, y por ende, se consume menor cantidad de tiempo.	
Figura 4.8. Agrupación por zonas, del espacio 3D bajo análisis.	

Tabla 4.2. Estructuras de datos para el ajuste y medición de error.

Ante el cálculo de un campo de distancia, para cada porción (plano) resultante se obtienen datos en tres archivos de texto en código ASCII: *distancias euclidianas, imagen del plano en escala de grises (formato PGM) e historial de transformaciones de triángulos 3D a 2D.*

- *Estructura de un archivo con distancias euclidianas.*

```
#dfWxH                    ---> Primer línea del archivo DF (distance field)
# COMENTARIO 1                 El símbolo '#' (número) indica que la línea
# COMENTARIO 2                 de texto es un comentario. "WxH" es la reso-
#     ...                      lución (pixeles) del plano: ancho x altura.
# COMENTARIO M
d01                       ---> Distancias euclidianas d. Indican las p dis-
d02                            tancias mínimas del plano al modelo PIF.
d03
...
dp
#Time taken: n seconds.   ---> La última línea es un comentario que indica
                               el tiempo (n segundos) invertido.
```

La resolución del plano debe provenir de un cuadrado, es decir, W=H.
Toda distancia "d" es un número real con un máximo de 10 números decimales.
La métrica del tiempo es únicamente en segundos.

Ejemplo de un fragmento de archivo de distancias euclidianas.

```
#df8100
139.2757337606
136.5536603258
...
131.9044242350
134.6379694631
#Time taken: 81 seconds.
```

Porción de un archivo DF (de distancias euclidianas): *La resolución del plano elegido es de 90X90 (W=H=90). Sólo se muestran algunos resultados de distancias, ya que el archivo de texto cuenta con un total de 8100 valores de distancias euclidianas. El tiempo invertido resultó ser de 81 segundos, para una elipsoide nivel 20 (1602 puntos con 3200 triángulos) en un equipo Intel PIV HT 3.2GHz, 1GB RAM.*

- *Estructura de una imagen del plano en escala de grises: PGM, subformato PNM.*

```
P2              ---> "Número mágico".
# COMENTARIO 1  ---> El símbolo '#' (número) indica que la línea de texto
# COMENTARIO 2       es un comentario.
#     ...
# COMENTARIO M
90 90           ---> Ancho y alto de la imagen (pixeles).
255             ---> Valor máximo de gris, lo cual define a: [0,...,255].
G0              ---> Nivel de gris G (número entero) para cada pixel.
G1
G2
...
Gn              ---> En total deben ser 90X90=8100 valores de grises.
```

Como se indica en el glosario, PNM es un grupo de formatos de imágenes [Pos91], del cual empleamos PGM en código ASCII (número mágico P2) para apreciar a los planos de campos de distancia como imágenes en escala de gris [0,...,255].

Ejemplo de una imagen PGM y un fragmento de su contenido en código ASCII.

```
P2
#Euclidean Distance Fields
90 90
255
0
5
...
14
9
```

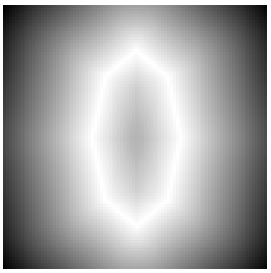


Figura 4.9. Plano XY con Z=0 del campo de distancias de una elipsoide nivel-2 (18 puntos con 32 triángulos).

- *Estructura de un historial de transformaciones de triángulos 3D a 2D.*

```
# Triangle to 2D. Applied to MODEL'S NAME, resolution: 3.333330
# COMENTARIO 1
# COMENTARIO 2
#     ...
# COMENTARIO M
CXc Yc Zc,ξ
P1P2P3 X1  Y1  Z1 ,X2  Y2  Z2 ,X3  Y3  Z3 ,T-Tx  Ty  Tz ,Yα ,Zβ ,Xγ ,Pδ
P2P3P1 X1A Y1A Z1A,X2A Y2A Z2A,X3A Y3A Z3A,T-Txa Tya Tza,Yαa,Zβa,Xγa,Pδa
P3P1P2 X1B Y1B Z1B,X2B Y2B Z2B,X3B Y3B Z3B,T-Txb Tyb Tzb,Yαb,Zβb,Xγb,Pδb
...
_R1  R2  R3  R4  R5  R6  R7  R8  R9
_R10 R11 R12 R13 R14 R15 R16 R17 R18
_R19 R20 R21 R22 R23 R24 R25 R26 R27
R28
```

Al grupo de renglones iniciales se le considera comentarios, al empezar con el símbolo '#'.
El primer renglón es el título que indica el nombre del modelo y la resolución distancia-pixel.
*Para **cada triángulo** del modelo, se cuenta con cuatro renglones de datos que reflejan al criterio empleado para justificar su interpretación 2D y así validar a la mínima distancia euclidiana que le corresponde al triángulo. Como parte final, se indica la cantidad de triángulos que pertenecen a cada una de las zonas posibles del espacio 3D.*

A continuación se detalla a la nomenclatura del historial, seguida de un ejemplo.
Para cada triángulo se almacena:

*(1) El centro geométrico (***Xc Yc Zc***) del triángulo y la zona (ξ) en la cual se encuentra.*

*(2) Primer opción para interpretar al triángulo en 2D (***P1P2P3***): P1 en el origen.*

(a) Ubicación 2D de los tres puntos (**X1 Y1 Z1 ,X2 Y2 Z2 ,X3 Y3 Z3**).

*(b) Traslación de P1 al origen (***Tx Ty Tz***).*

*(c) Ángulos de rotación para colocar a P2 en el eje X (***Yα ,Zβ***) y a P3 en el plano XY (***Xγ***).*

*(d) Ángulo de inclinación P1P3 en el plano XY (***Pδ***).*

*(3) Segunda opción para interpretar al triángulo en 2D (***P2P3P1***): P2 en el origen.*

(a) Ubicación 2D de los tres puntos (**X1A Y1A Z1A ,X2A Y2A Z2A ,X3A Y3A Z3A**).

*(b) Traslación de P2 al origen (***Txa Tya Tza***).*

*(c) Ángulos de rotación para colocar a P3 en el eje X (***Yαa ,Zβa***) y a P1 en el plano XY (***Xγa***).*

*(d) Ángulo de inclinación P2P1 en el plano XY (***Pδa***).*

*(4) Tercer opción para interpretar al triángulo en 2D (***P3P1P2***): P3 en el origen.*

*(a) Ubicación 2D de los tres puntos (***X1B Y1B Z1B ,X2B Y2B Z2B ,X3B Y3B Z3B***).*

*(b) Traslación P3 a origen (***Txb Tyb Tzb***).*

*(c) Ángulos de rotación para colocar a P1 en el eje X (***Y**α**b ,Z**β**b***) y a P2 en el plano XY (***Xy*b***).*

*(d) Ángulo de inclinación P3P2 en el plano XY (***P**δ**b***).*

Para las 27+1 zonas posibles de ubicación de triángulos (mediante su centro geométrico):

*(1) Se indica la cantidad de triángulos en cada una de las zonas válidas (***R1,...,R27***).*

(2) En caso de que algún triángulo del modelo no se encuentre en alguna de las 27 zonas válidas, se le clasifica en la zona "27+1" *(***R28***).*

Ejemplo de historial en un triángulo de un modelo elipsoide nivel-1.

```
      Posición del triángulo analizado:
P1: 20.1000000000 88.7700000000 0.5000000000
P2: 20.1000000000 -11.1000000000 116.1600000000
P3: 121.2100000000 -11.1000000000 0.5000000000
```

La fig.4.10 muestra el plano XY, Z=3.33333 del campo de distancia en la elipsoide nivel-1.

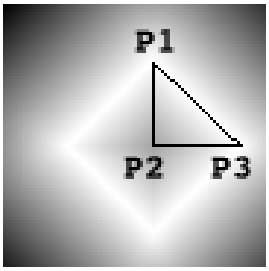


Figura 4.10. Se destaca al triángulo analizado.

```
# Triangle to 2D. Applied to ./fig10R_, resolution: 3.333330
C53.8033332825 22.1900005341 39.0533332825,15          → Centroide en zona 15.
P1P2P3 0.0000000000 0.0000000000 0.0000000000,         → 1er opción: P1 al origen
152.8111661496 -0.0000000000 -0.0000000000,              P2 al eje X
65.2702099677 126.2420242660 -0.0000000000,              P3 al plano XY
T-20.1000000000 -88.7700000000 -0.5000000000,          → Traslación 1er opción
Y4.7123889804,Z5.5709177576,X4.0704259237,P1.0936221217 → Rotaciones 1er opción
P2P3P1 87.0775238968 125.5737127427 -0.0000000000,     → 2da opción: P1 al plano XY
0.0000000000 0.0000000000 0.0000000000,                  P2 al origen
153.6244371837 0.0000000000 -0.0000000000,               P3 al eje X
T-20.1000000000 11.1000000000 -116.1600000000,         → Traslación 2da opción
Y0.8524194670,Z0.0000000000,X5.6319073811,P0.9644883037 → Rotaciones 2da opción
P3P1P2 142.1170257218 0.0000000000 -0.0000000000,      → 3er opción: P1 al eje X
71.9353085816 135.7415893500 0.0000000000,               P2 al plano XY
0.0000000000 0.0000000000 0.0000000000,                  P3 al origen
T-121.2100000000 11.1000000000 -0.5000000000,          → Traslación 3er opción
Y3.1415926536,Z0.7792284735,X4.1615012054,P1.0834822282 → Rotaciones 3er opción
          ...
 _0 0 0 0 0 0 0 0 0 0
 _0 0 0 0 4 4 0 0 0 0
 _0 0 0 0 0 0 0 0 0 0
 0
```

Para todo triángulo del modelo se presenta la misma información con sus datos correspondientes.

→ ***Finalmente, el historial señala la distribución de triángulos en las 27+1 zonas posibles. En este caso, los 8 triángulos se encuentran en zonas conocidas (R14 y R15).***

Revisando a detalle el historial, se observa que es posible validar la interpretación 2D de los triángulos con una inspección rápida: *todo triángulo debe tener z=0.0 en sus tres puntos, y en cualquiera de las tres opciones de ajuste.*

4.2 Transformaciones geométricas en 3D.

Como se mencionó en capítulos previos, una razón potencial del uso de OpenGL es su capacidad de proporcionar un ambiente gráfico tridimensional con alto rendimiento en tiempo real, gracias a tomar en cuenta la aceleración 3D a nivel hardware. Al ser *Mesa 3D Graphics Library* la implementación símil de OpenGL indicada para satisfacer las necesidades del proyecto presente, se analizó la funcionalidad de las transformaciones geométricas para llevar a cabo la implementación del Visor gráfico en tiempo real, así como el diseño de transformaciones propias de nuestra aplicación, basado en sus principios matemáticos, donde la representación estándar de las transformaciones geométricas es una matriz, ya que [Lan05]:

- Es eficiente en transformaciones lineales (en este caso, escala y rotación).
- Permite el uso de *coordenadas homogéneas* para representar transformaciones afines (y poder incluir a la traslación en el esquema lineal); lo anterior implica el uso de una cuarta coordenada (w) en el espacio tridimensional (la cual en la práctica tiene el valor w=1, por simplicidad):

  (x, y, z, w), representa a la posición $\left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}\right)$.
  $(x, y, z, 0)$, representa a un punto en el infinito.
  $(0,0,0,0)$, es indeterminado.

- La multiplicación matricial, *al ser asociativa*, permite combinar a las transformaciones y representarlas en una sola matriz; sin embargo el orden de la multiplicación *si* altera al resultado, es decir, *esta multiplicación no es conmutativa.*

$$p' = \left(T_0 * \left(T_1 * \left(T_2 p\right)\right)\right) = \left(\left(\left(T_0 * T_1\right) * T_2\right) p\right) = \left(T_0 * T_1 * T_2\right)(p)$$
$$\left(T_1 * T_2\right)(p) \neq \left(T_2 * T_1\right)(p)$$

donde: p es la posición original,
p' es la posición resultante,
T_0, T_1, T_2 son las matrices de transformaciones.

La nomenclatura estándar de las matrices comunes de transformaciones geométricas 3D, usando coordenadas homogéneas, es la siguiente:

Matrices comunes de transformaciones 3D.

Transformaciones lineales (escalas y rotaciones) y afines (traslaciones).

Matriz de Escala.

$$T_S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz de Traslación.

$$T_T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz de Rotación en *X*.

$$T_{Rx} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz de Rotación en *Y*.

$$T_{Ry} = \begin{bmatrix} \cos(\alpha) & 0 & -\sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

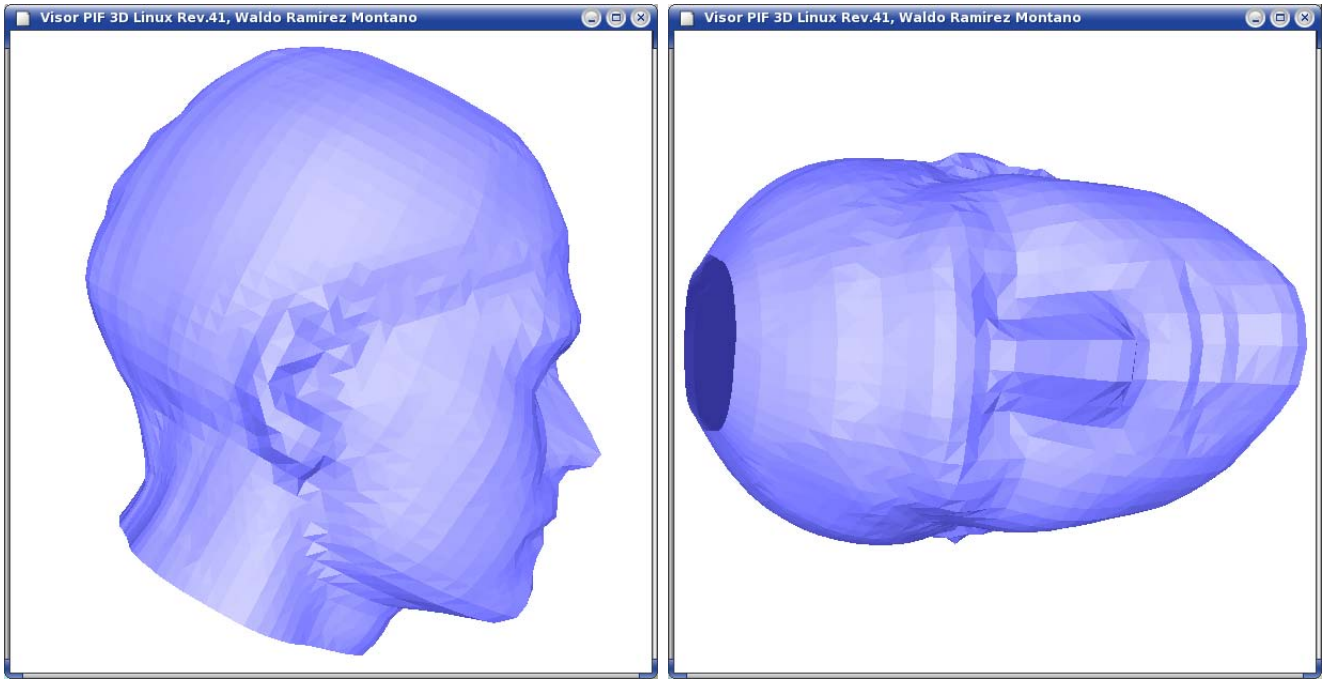
Matriz de Rotación en *Z*.

$$T_{Rz} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Mediante multiplicaciones matriciales, el diseño de las transformaciones necesarias para la alineación geométrica se abstrae en un grupo de tres transformaciones para el objeto 3D:

- *Transformaciones rígidas que modifican al modelo para alinearlo geométricamente.*

Este grupo incluye a traslaciones y rotaciones; alteran a los datos del modelo con el fin de diagonalizar a su matriz de tensor de inercia. Debido a que el orden de las transformaciones si altera al resultado, se debe (1) trasladar el centroide geométrico al origen, para después (2) rotar al modelo buscando que sus eigenvectores se alineen. Además, existe un sub-conjunto de rotaciones que mantiene la alineación del modelo, a pesar de que sí altera a sus datos: rotaciones de $90 \times n$ *grados,* donde $n=1, 2, 3, ..., m$; permiten presentar al tensor de inercia alineado de forma estándar, como se ejemplifica en la *fig.4.11*,



Tensor de inercia de cabeza04.pi alineado en forma no-estándar.

$$I = \begin{bmatrix} I_{zz} & 0 & 0 \\ 0 & I_{xx} & 0 \\ 0 & 0 & I_{yy} \end{bmatrix}$$

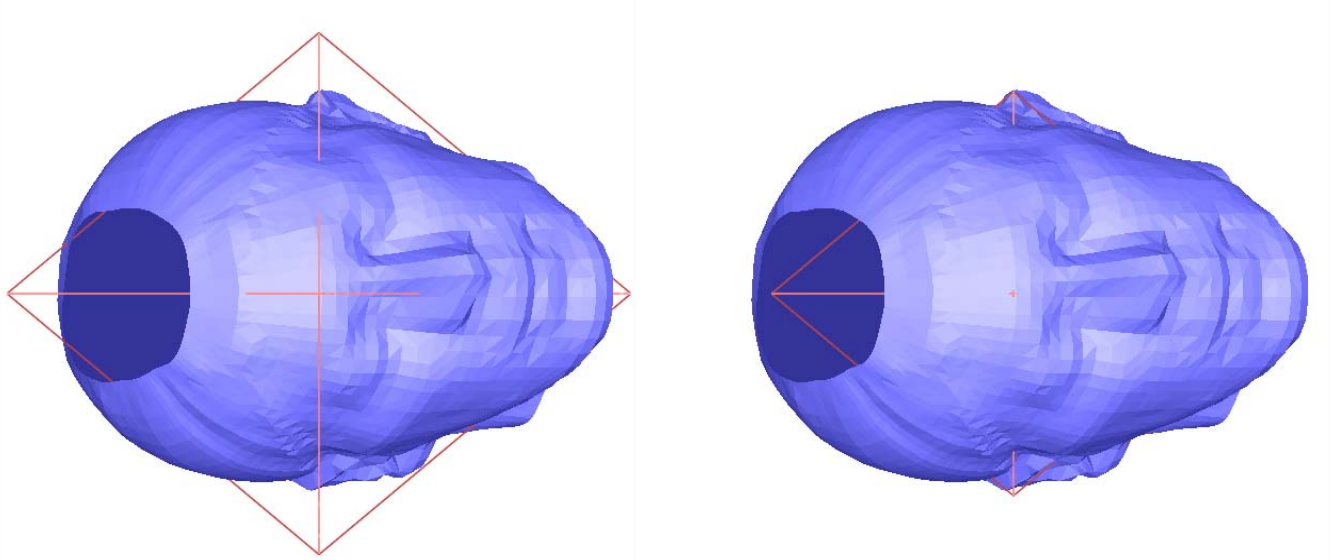
Tensor de inercia de cabeza04.pi alineado en forma estándar, después de rotaciones de 90 grados en ejes X, Y, Z.

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

Figura 4.11. Modelo cabeza04.pi con tensor de inercia alineado en forma no-estándar (izquierda) y en forma estándar (derecha). *Proyección frustum con normales planas*.

Cabe señalar que después de la primer iteración, es necesario aplicar transformaciones de traslación particulares a los *puntos externos ó intrusos (outliers)* con el fin de mejorar el a-juste robusto de la elipsoide al modelo de cabeza correspondiente.

• *Transformaciones que modifican al modelo para ser homogéneo respecto a otro modelo.*
La expresión "ser homogéneo" se refiere a transformar al modelo elipsoide para que cuente con eigenvalores y eigenvectores equivalentes a los propios del modelo con el cual se esté comparando. En nuestro caso la transformación de escala se encarga de cumplir el criterio, al ser aplicado de forma uniforme a la elipsoide, para así hacerla homogénea a su modelo de cabeza. Es imprescindible que ambos modelos estén alineados geométricamente y que la elipsoide haya sido creada a partir del Visor.

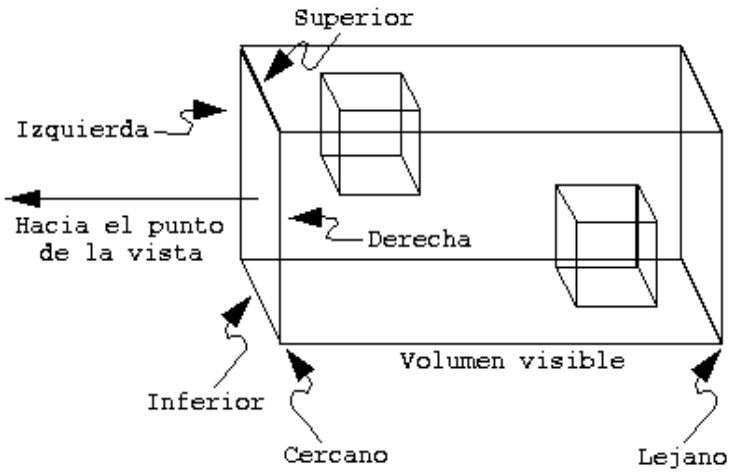


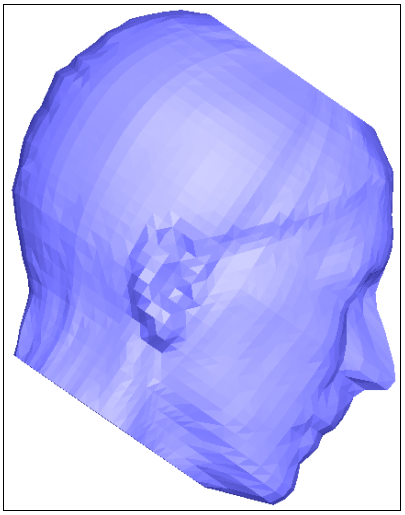
Proyecciones ortogonales (ortho) con normales planas.

Tensores de Inercia:		***Tensores de Inercia:***	
6014.907 0.0 0.0	***10024.845*** 0.0 0.0	***6014.907*** 0.0 0.0	***6014.907*** 0.0 0.0
0.0 ***7578.523*** 0.0	0.0 ***12630.871*** 0.0	0.0 ***7578.523*** 0.0	0.0 ***7578.523*** 0.0
0.0 0.0 ***6882.647***	0.0 0.0 ***11471.078***	0.0 0.0 ***6882.647***	0.0 0.0 ***6882.647***
cabeza03	**elipsoide**	**cabeza03**	**elipsoide**

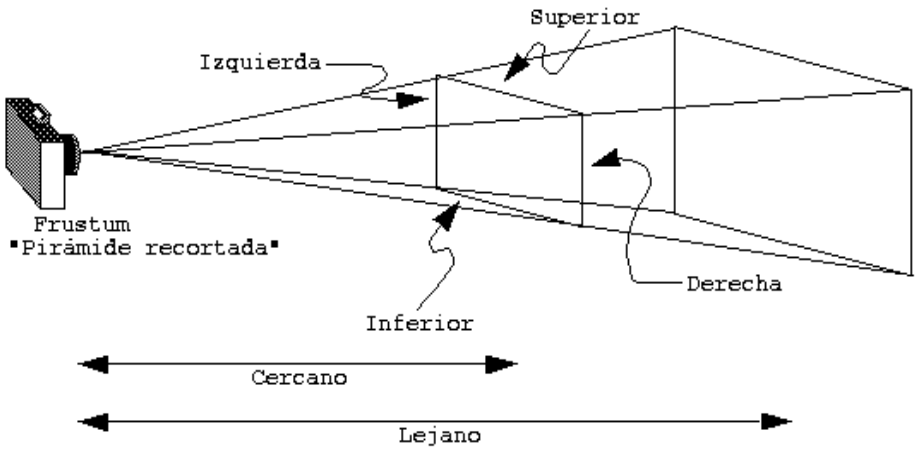
Figura 4.12. Modelo cabeza03.pi y su elipsoide nivel-1: No homogénea (izquierda) y homogénea (derecha, después de aplicarle las transformaciones de escala necesarias). Se incluye a los tensores de inercia con tres dígitos decimales, antes y después de las transformaciones (en la aplicación, se consideran a 10 dígitos decimales para los cálculos).

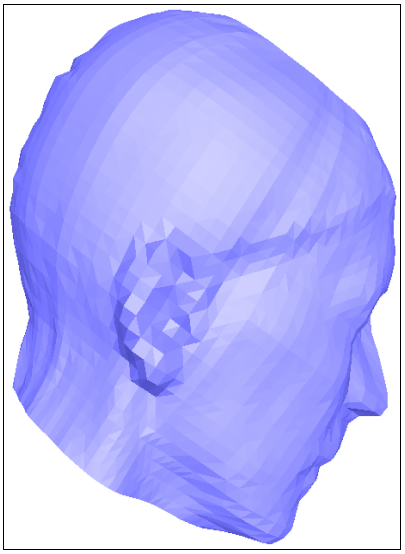
• *Transformaciones sin modificación al modelo.*
Aprovechando la funcionalidad en tiempo real que ofrece Mesa, este grupo de transformaciones sólo aplica en el módulo visor, es decir, *no hacen cambios a los datos del modelo* (y por ende, no altera a su tensor de inercia): únicamente se perciben en el despliegue en pantalla, lo cual permite revisar y observar a detalle a los modelos sin perder su alineación geométrica. Este grupo esta compuesto por *transformaciones de vista* (traslaciones, rotaciones y escalas de Mesa/OpenGL) y *transformaciones de proyección* (dos perspectivas: Frustum y Ortho, véase *fig.4.13*).





Proyección Ortogonal (Ortho).





Proyección Piramidal (Frustum).

Ejemplo con cabeza03.pi

Figura 4.13. *Transformaciones de Proyección*.

Los dos tipos de transformación cuentan con los seis planos que definen a su volumen visible: *superior, inferior, derecha, izquierda, cercano, lejano*. La diferencia fundamental es que, en Ortho la proyección es un plano, mientras que en Frustum la proyección es un punto:

El ejemplo con *cabeza03.pi* (usando *la misma posición* en el espacio 3D), ilustra al párrafo previo: en la *proyección Ortho* se obtiene la apariencia plana (por ejemplo, se aprecia al corte de línea recta en el cuello), mientras que la *proyección Frustum* simula la apariencia de segmentos cercanos y lejanos (por ejemplo, la curvatura en el corte del cuello).

4.3 Modelos simplificados.

La representación analítica de los modelos de cabezas humanas se hizo mediante un grupo de modelos simplificados ya antes mencionado: *los modelos canónicos de elipsoides homo-*

géneas. Independientemente de ser elipsoide homogénea o no, el algoritmo del cual nos basamos [Ahn95], que es conocido como *método de subdivisión angular (angular subdivision method),* aplica para la creación de una aproximación poligonal (triángulos) de elipsoides canónicas con *centroide* en el origen, mediante tres pasos generales:

1. *Inicialización de valores trigonométricos.* Valores de seno y coseno se obtienen en correspondencia a los ángulos de azimut y de incidencia en vértices del primer octante de la esfera con radio unitario.

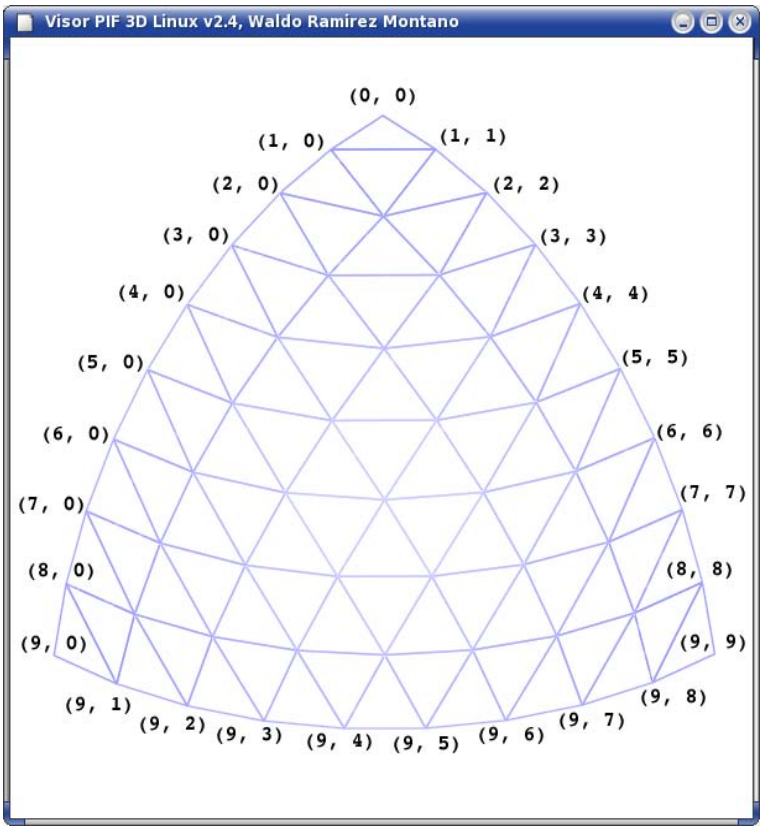


Figura 4.14. Primer octante de esfera unitaria. Ejemplo con subdivisión angular de nivel 9 (cantidad de separaciones; 81 triángulos en total).

2. *Generación del primer octante de la elipsoide.* El primer octante de la esfera unitaria es escalado de forma apropiada en cada dirección de eje.

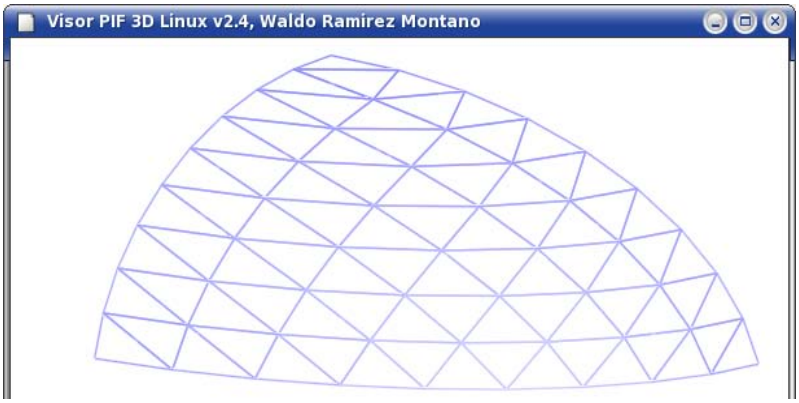


Figura 4.15. Aplicación de transformaciones de escala al primer octante de la esfera unitaria. Con ello se obtiene la representación del primer octante de la elipsoide esperada. Las escalas dependen de los valores *a, b, c* de la ecuación canónica elipsoidal.

3. *Generación de la elipsoide.* Vértices (e índices de planos) de los siete octantes restantes son generados como reflexiones de aquellos en el primer octante, obteniendo a todo polígono triangular que representa a la elipsoide canónica.

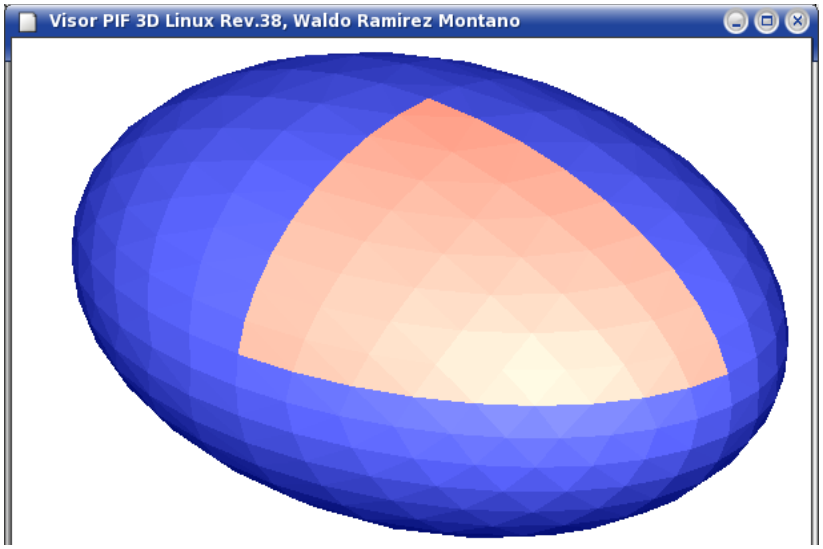


Figura 4.16. La reflexión de vértices y normales completa a la aproximación poligonal del modelo elipsoidal. En esta figura, se resalta al octante raíz de la elipsoide resultante.

Es importante indicar que mediante la reducción de cálculos numéricos en el método descrito, se obtiene una considerable mejora en el rendimiento del trabajo para la generación de aproximaciones de modelos elipsoidales: se implementaron a las estructuras *triángulo* y *vector_xyz*, con lo cual los procesos de alta demanda de cálculo (aquellos que hacen uso de funciones trigonométricas para definir a estructuras *vector_xyz)*, así como la indexación inicial de cada *triángulo* sólo se aplican en una octante de la esfera unitaria. Posteriormente, la carga de cálculos numéricos se reduce, ya que los procesos de escala son básicamente las multiplicaciones definidas por los valores (*a, b, c*) en la ecuación canónica de la elipsoide (implementada con la estructura *ecuación3d*), que se aplica sólo en este primer octante a cada *vector_xyz* (no hay necesidad de cambios en su indexación). Para obtener a los siete octantes adicionales, basta el hacer la reflexión del primer octante y almacenar al modelo en su *archivo PIF* destino: asignar el signo (+)positivo ó (-)negativo (dependiendo del octante objetivo) a cada estructura *vector_xyz* correspondiente, así como completar la indexación con la reflexión de cada *triángulo,* mediante la relación entre sus índices y la cantidad de puntos en el primer octante.
Como punto final, tomamos alta precaución para evitar la presencia de puntos redundantes, debido a la alta sensibilidad del tensor de inercia. En el *apéndice D.2* se ejemplifica al crecimiento del mallado (cantidad de puntos y triángulos) en relación al incremento del nivel de la subdivisión angular.

4.4 Alineación geométrica iterada y robusta.

Previamente (capítulo II) se mencionó al procedimiento base que usamos en la alineación geométrica: *PCA*. A continuación se presenta al diseño iterativo, el cual refina a la elipsoide

homogénea representativa del modelo de cabeza, ya que considera el *ajuste a los puntos externos (ó intrusos)* y permite el *reuso de PCA*, disminuyendo así la medida resultante del error morfológico (hecha con *campos de distancia*).

A continuación se presenta la nomenclatura basada en [Som92], de los diagramas de flujo de datos en el diseño funcional (*function-oriented design*) del proyecto.

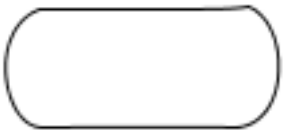
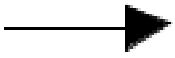
Símbolo	*Descripción*
	*Rectángulos con extremos curvos*. Representan transformaciones del flujo de datos de entrada, al flujo de salida. Deben tener una etiqueta descriptiva.
	*Rectángulos*. Representan un repositorio de datos. Deben tener una etiqueta descriptiva.
	*Círculos*. Representan interacciones del usuario con los programas, los cuales pueden proporcionar una entrada de datos ó recibir una salida de datos. Es posible etiquetarles con un número que indique la continuación del diagrama de flujo.
	*Flechas.* Muestran la dirección del flujo de datos; se les asigna un nombre que describe al flujo de datos que viaja a lo largo de dicho camino.
	*Palabras clave "AND" y "OR"*. Mantienen las interpretaciones usuales que tienen en expresiones booleanas. Se emplean para ligar a flujos de datos cuando más de un flujo puede ser entrada ó salida de una transformación.
	*Símbolo arco que une a flujos de datos*. Se emplea sobre "OR" con dos o más flujos para indicar OR exclusivo (XOR).

Tabla 4.3. Nomenclatura de los diagramas de flujo de datos.

El diseño de las herramientas busca la interacción con el usuario, por lo cual se observa que en distintas etapas de los diagramas, se requiere su intervención para aplicar algunos cambios de estado esperados (por ejemplo, el usuario encamina la alineación de un modelo de cabeza, mediante transformaciones paso a paso). *MVC* proporciona ventajas en su mantenimiento, ya que la reducción de intervenciones del usuario no requiere un completo rediseño: es viable con modificar únicamente a la capa del controlador (por ejemplo, que el usuario encaminara la alineación de un modelo de cabeza con una simple instrucción, es decir, con un solo paso que invocara a todas las transformaciones necesarias).

A continuación se presentan las primeras dos partes del flujo completo del proceso iterativo: la alineación geométrica del modelo de cabeza (flujo 1) y la creación de la elipsoide homogénea representativa (flujo 2). En la *sección 4.6* se completa al flujo con la porción que hace la medición del error y aplica el criterio de corrección de puntos externos (flujo 3), con lo cual se inicia una nueva iteración (flujo 1, flujo 2, flujo 3) hasta satisfacer la tolerancia de error.

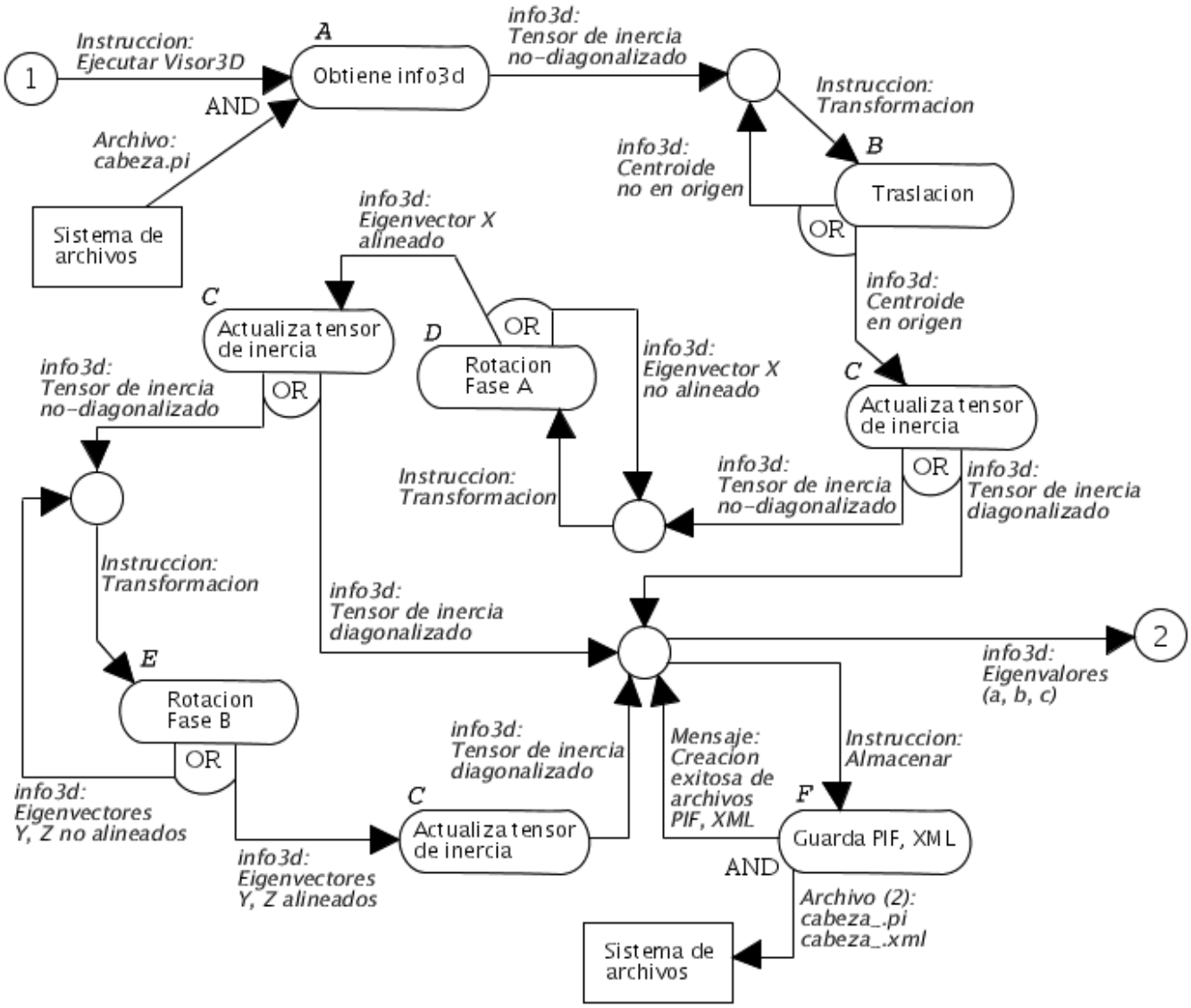


Figura 4.17. Flujo 1, alineación geométrica del modelo cabeza.

Detalle en las transformaciones de flujos de datos empleadas en el Flujo 1.

- *(A) Obtiene info3d.* Creación de la estructura info3d del modelo cabeza bajo análisis.
- *(B) Traslación*. Se aplica uniformemente a todo el modelo cabeza, en donde el argumento es su centro geométrico, previamente calculado y almacenado en su estructura info3d.
- *(C) Actualiza tensor de inercia.* Recalcula al tensor de inercia del modelo cabeza después de aplicarle una transformación geométrica de alineación.
- *(D) Rotación Fase A.* Conjunto de rotaciones geométricas (sobre ejes Y, Z) que alinean al *eigenvector X* del modelo cabeza.
- *(E) Rotación Fase B.* Conjunto de rotaciones geométricas (sobre el eje X) que alinean a los *eigenvectores Y, Z* del modelo cabeza.
- *(F) Guarda PIF, XML.* Traducción del flujo de datos del modelo cabeza a los archivos que se deben almacenar en el repositorio.

Observación: Toda transformación ocurre en el programa visor (Visor PIF 3D).

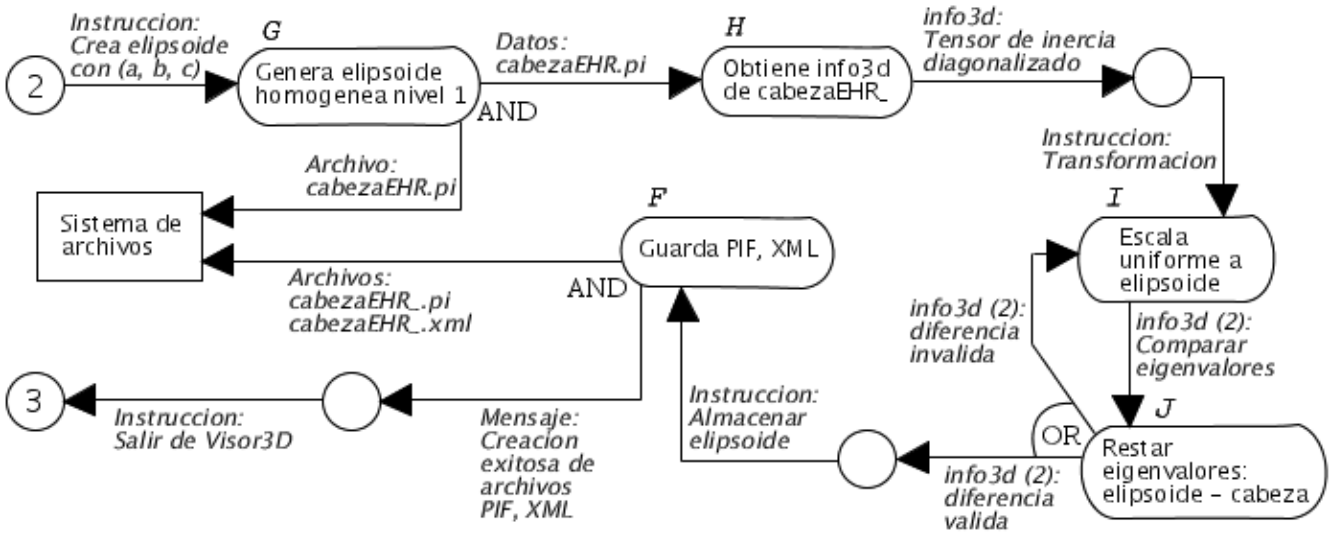


Figura 4.18. Flujo 2, Creación de la elipsoide homogénea representativa.

Detalle en las transformaciones de flujos de datos empleadas en el Flujo 2.

- *(G) Genera elipsoide homogénea nivel 1.* Creación del modelo PIF que, **para cada iteración,** corresponde a la primer versión[12] de la elipsoide homogénea del modelo de cabeza; se incluye la traducción del flujo de datos al archivo que se almacena en el repositorio.
- *(H) Obtiene info3d de cabezaEHR_.* Creación de la estructura info3d del modelo de elipsoide nivel 1, primer versión en la iteración.
- *(I) Escala uniforme a elipsoide.* Aplica la transformación geométrica de escala uniforme al modelo de elipsoide nivel 1.
- *(J) Restar eigenvalores: elipsoide-cabeza.* Verifica que los eigenvalores del modelo de elipsoide cumplan con el criterio de similitud para validar la obtención de los valores analíticos (*a, b, c*) de la elipsoide equivalente (elipsoide sólida homogénea).
- *(F) Guarda PIF, XML.* Traducción del flujo de datos del modelo de elipsoide equivalente a los archivos que se deben almacenar en el repositorio.

Observación: Toda transformación ocurre en el programa Visor3D (Visor PIF 3D).

En caso de que no se cumpla el trayecto esperado del flujo de datos en alguna de sus transformaciones (por ejemplo, que no se cumpla algún *XOR* ó *AND*), la respuesta al usuario es un mensaje de información (INFO), advertencia (WARNING) ó error (ERROR) que refleja al camino inesperado. Con ello se puede definir la causa del evento.

Antes de presentar a la tercera parte del flujo iterativo, es preferible detallar a la base de la estrategia que usamos para la medición del error entre el modelo de cabeza y su elipsoide equivalente: los *campos de distancia con métrica euclidiana.*

12 Cabe aclarar que la *primer versión de la elipsoide homogénea*, **en cada iteración**, no corresponde a la elipsoide equivalente del modelo cabeza: debe satisfacer el criterio de similitud antes de ser equivalente.

4.5 Campos de distancia Euclidianos.

Dentro de las técnicas de modelado en el espacio tridimensional discreto, se han desarrollado métodos que hacen uso de *campos escalares* [WW89] para satisfacer la demanda de procesos de cómputo en la representación de modelos dinámicos (por ejemplo, el modelado de un líquido en movimiento). Para el caso específico del análisis de modelos derivados de planos triangulares, existe una alternativa que hace uso de un campo escalar en particular: el *campo de distancia* [PT92] (en dos dimensiones se le conoce como *mapa de distancias*). Con ello, en nuestro caso es posible aplicar un proceso de comparación entre muestras de campos de distancia, con el cual se obtiene un mapa de la distribución de error punto a punto entre los modelos de cabeza humana y su elipsoide equivalente; además lo anterior permite establecer la similitud de los modelos de cabeza con la elipsoide promedio de una población. Es importante señalar que es realizable la obtención de distintos campos de distancia en el análisis de un mismo modelo, gracias a contar con la posibilidad de emplear distintas *métricas* para interpretar la distancia. Para obtener una representación viable de la distancia de acuerdo a los requerimientos, se emplea la *métrica Euclidiana,*

$$D_2(X_j, X_k) = \sqrt{\sum_{i=1}^{n} |x_{ij} - x_{ik}|^2}$$

Donde: $X_j = (x_{1j}, x_{2j}, ..., x_{nj}), X_k = (x_{1k}, x_{2k}, ..., x_{nk})$, en el espacio $\mathbb{R}^n$.

Un campo de distancia dentro del ambiente tridimensional obtiene a la *menor distancia* de cada punto al *objeto* del modelo que le corresponda [SOM04]. La definición formal de un campo de distancia Euclidiano en 3D, con signo, asociado a la superficie (frontera) ∂A de un objeto A (la malla triangular de una cabeza humana) es:

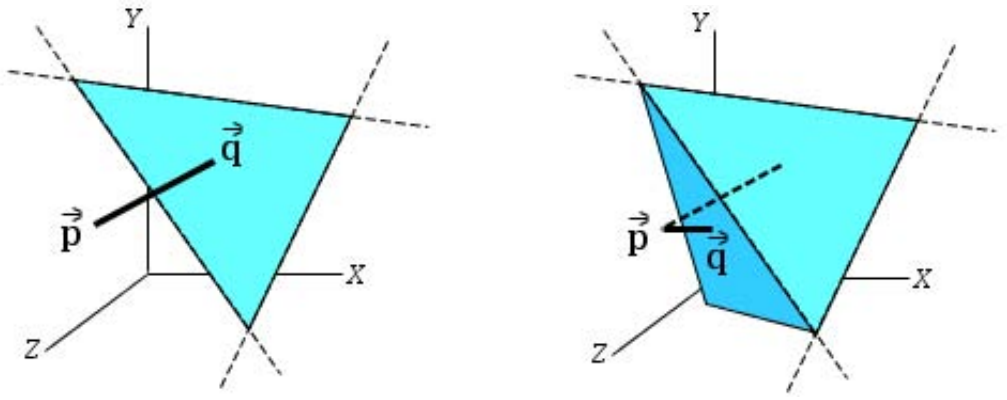
$$\mathbf{D}(\partial \mathrm{A}) \triangleq \{(\vec{\mathbf{p}}, d(\vec{\mathbf{p}})) \mid d = \mathrm{sgn} \cdot \min_{\vec{\mathbf{q}} \in \partial \mathrm{A}} \|\vec{\mathbf{p}} - \vec{\mathbf{q}}\|, \vec{\mathbf{p}} \in \mathbb{R}^3\}$$

$$\mathrm{sgn} = \begin{cases} +1 & \text{si} \quad \vec{\mathbf{p}} \in \mathrm{A}^c \\ -1 & \text{si} \quad \vec{\mathbf{p}} \in \mathrm{A} \end{cases}$$

Nótese que una muestra del campo **D** es una imagen escalar, cuyo atributo es la distancia del punto $\vec{\mathbf{p}} = (x_p, y_p, z_p)$ al punto $\vec{\mathbf{q}} = (x_q, y_q, z_q)$ más cercano de la superficie de A (es decir, $\vec{\mathbf{q}} \in \partial A$). La convención de signo es arbitraria y depende de las aplicaciones: la diferencia principal es que, un campo sin signo (sgn = +1 siempre) no permite distinguir si el punto $\vec{\mathbf{p}}$ está en el interior ($\vec{\mathbf{p}} \in \mathrm{A}$) ó en el exterior ($\vec{\mathbf{p}} \in \mathrm{A}^c$) de A.
La frontera ∂A puede recuperarse a partir de **D** si se toman a todos los puntos $\vec{\mathbf{p}}$ cuyo atributo sea cero (ya que por definición, corresponden justo a puntos en ∂A). En el espacio discreto ocurre cierta pérdida de información: la distancia Euclidiana se calcula en punto flotante, pero al redondearla al valor entero de la imagen ($[0,\ldots,255]$, en escala de grises), algunos puntos presentan (1) valores que ya no corresponden a la superficie original ∂A, ó (2) valores que "sobren" produciendo una superficie de espesor mayor al original.

Al proceso que genera un campo de distancia Euclidiano, se le denomina *Transformación de Distancia Euclidiana* (Euclidian Distance Transform, EDT), y para el proyecto presente, un objeto se puede considerar en particular como una primitiva geométrica (un triángulo), del modelo tridimensional:

Considerando a cada primitiva como t_i, compuesta de puntos $\vec{\mathbf{q}} \in \partial A$, es posible definir a una función escalar $dist(\vec{\mathbf{p}}, t_i)$ que denota a la distancia del punto $\vec{\mathbf{p}} \in \mathbb{R}^3$, al punto $\vec{\mathbf{q}}$ más cercano en t_i. A su vez, la distancia mínima de $\vec{\mathbf{p}}$ al grupo de primitivas $T = \{t_1, t_2, \ldots, t_n\}$ es representada como: $dist(\vec{\mathbf{p}}, T) = \min_{t_i \in T}(dist(\mathbf{p}, t_i))$.



(a) ***(b)***

Figura 4.19. Cada primitiva contiene puntos de la superficie del modelo, de los cuales (a) uno es el de menor distancia con respecto al punto bajo análisis, aunque el ganador (b) debe ser el punto que se encuentra en la primitiva que reporta la menor distancia.

Por lo tanto, el campo de distancia $\mathbf{D}_{\boldsymbol{M}}(T)$ para un dominio $\boldsymbol{M} \subset \mathbb{R}^3$, es el *campo escalar* que representa al modelo, dado por la función de distancia mínima $dist(\vec{\mathbf{p}}, T)$ para todo punto $\vec{\mathbf{p}} \in \boldsymbol{M}$. Cabe señalar, que al considerar un subgrupo de primitivas, $\chi \subset T$, se debe cumplir que: $dist(\vec{\mathbf{p}}, \chi) \geq dist(\vec{\mathbf{p}}, T), \forall\, \vec{\mathbf{p}} \in \boldsymbol{M}$. Como se puede apreciar, éste campo de distancia es sin signo (toda distancia resultante es mayor o igual a cero).
La implementación de $\mathbf{D}_{\boldsymbol{M}}(T)$ con la métrica Euclidiana, conlleva a la necesidad de gran cantidad de cálculos para obtener el campo de distancia [SJ01], razón por la cual han existido aproximaciones a la verdadera métrica Euclidiana. Para éste caso, es importante la precisión de la distancia a escala milimétrica, por lo cual se tomó la decisión de emplear un *método híbrido* para el cálculo del campo de distancia, como se describe a continuación.

Cálculo de la distancia entre un punto y un triángulo.
Durante el desarrollo iterativo del algoritmo, fue necesario diseñar un método que facilite la obtención de la distancia Euclidiana punto-triángulo en el espacio tridimensional, debido a eliminación de defectos y obtención de mejoras en la reducción del tiempo que demora el cálculo de las muestras del campo de distancia, como se ejemplifica en la *fig.4.20*. El emplear un método matemático que haga uso de operaciones vectoriales para definir a la distancia desde el punto de vista tridimensional [Jon95], requiere demasiado tiempo y esfuerzo computacional, por lo que fue de mayor conveniencia el emplear al *Método de 2D* propio de Mark W. Jones, ya que transforma al cálculo de tres dimensiones en dos dimensiones, mediante la definición de *Ecuaciones de Frontera* (Edge Equations) que aprovechan al plano que define el triángulo y hacen uso de diferenciales para establecer criterios de relación entre el triángulo y el punto:

$$E(x, y) = (x - X)dY - (y - Y)dX$$

La ecuación de frontera *E(x,y)* establece que para el punto (x,y) existen tres alternativas:
- $E<0$, el punto se encuentra *a la izquierda* de la línea (hacia afuera del triángulo).
- $E>0$, el punto se encuentra *a la derecha* de la línea (hacia adentro del triángulo).
- $E=0$, el punto se encuentra sobre la línea.

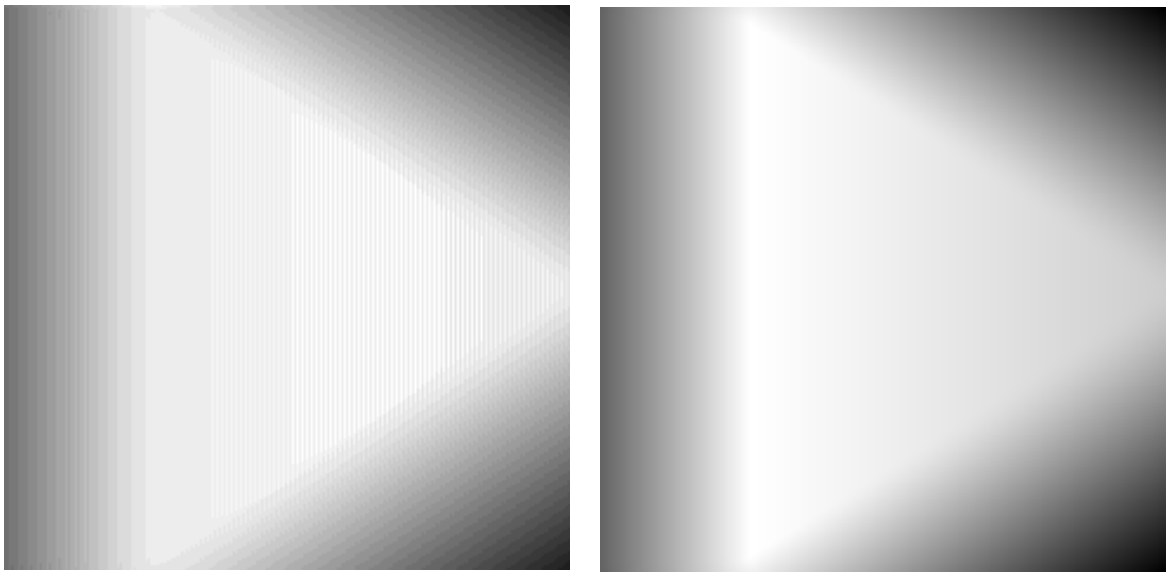


Figura 4.20. Ejemplo de resultados en la evolución iterativa del programa *dfields*. Izquierda: Quinta revisión del programa. Derecha: Décimo octava revisión del programa. *Se empleó el mismo modelo (un sólo triángulo)*[13].

En la ecuación anterior se hace referencia a un espacio no discreto, por lo cual, respetando la nomenclatura del autor, se aclara que en el diseño del algoritmo se implementó el perfil discreto de la ecuación, haciendo uso de diferencias: en las ecuaciones de frontera, se requiere el uso de dos puntos (*P1, P2*), que definan a la recta correspondiente. Con lo anterior, es posible determinar los coeficientes de la ecuación de frontera,

$$dY = P1_Y - P2_Y$$
$$dX = P1_X - P2_X$$
$$E(x,y) = (x - P1_X)dY - (y - P1_Y)dX = xdY - ydX + P1_Y dX - P1_X dY$$

Es importante recordar que los campos de distancia, a pesar de ser de una sola dimensión, tienen un valor escalar para todo punto del espacio $\mathbb{R}^3$. En la estimación de errores morfológicos (sección 4.7) se detalla qué muestras, del campo de distancia, fueron consideradas para definir al criterio de similitud.

Cálculo del campo de distancia euclidiano.
El algoritmo propuesto evolucionó de acuerdo a las pruebas de funcionamiento, haciendo uso de las estructuras *info3d, triangulos2D, filtroPuntos* y de las Ecuaciones de Frontera. Se tomó énfasis en la disminución de tiempo necesario para obtener a las muestras de los campos de distancia. Con ello, el algoritmo se resume en el diseño siguiente[13]:
1) Asignación de zona representativa a todo triángulo (mediante *filtroPuntos*).
2) Calculo de la distancia mínima del punto representativo de cada zona.
3) Obtención del triángulo significativo para cada zona.

13 Consulte al apéndice D.3 para mayor detalle.

4) Calculo de la distancia mínima de todo punto $\vec{\mathbf{p}} \in \boldsymbol{M}$ de la muestra (plano) buscada, de acuerdo al criterio de triángulos significativos.
5) Creación de resultados en archivos de texto ASCII: dos de distancias y una bitácora de transformaciones:
   1. Distancias Euclidianas en punto flotante, con 10 dígitos decimales de precisión.
   2. Imagen en escala de grises, en formato PGM.
   3. Historial de transformaciones de triángulos 3D a 2D.

Con *triangulos2D,* se implementan en el plano XY a las tres *Ecuaciones de Frontera* de cada triángulo, permitiendo evaluar en cada ecuación, al punto $\vec{\mathbf{p}}$ transformado. Con ello se obtienen los tres resultados (positivo, cero ó negativo) que determinan cual es el vector (proyectado en Z) con la menor magnitud (distancia), del punto al triángulo.

4.6 Diseño del proceso iterativo de ajuste de modelos.

Como se establece en flujo 2 de la sección 4.4, para la medición del error, los campos de distancia son la raíz del criterio de similitud[14] que establece al error entre el modelo de cabeza humana y su elipsoide equivalente: la relación entre las muestras de los campos de distancia es su *diferencia,* por lo cual el resultado ideal sería la obtención del mismo campo de distancia en ambos modelos, provocando el mismo valor de distancia para cualquier punto del espacio tridimensional.

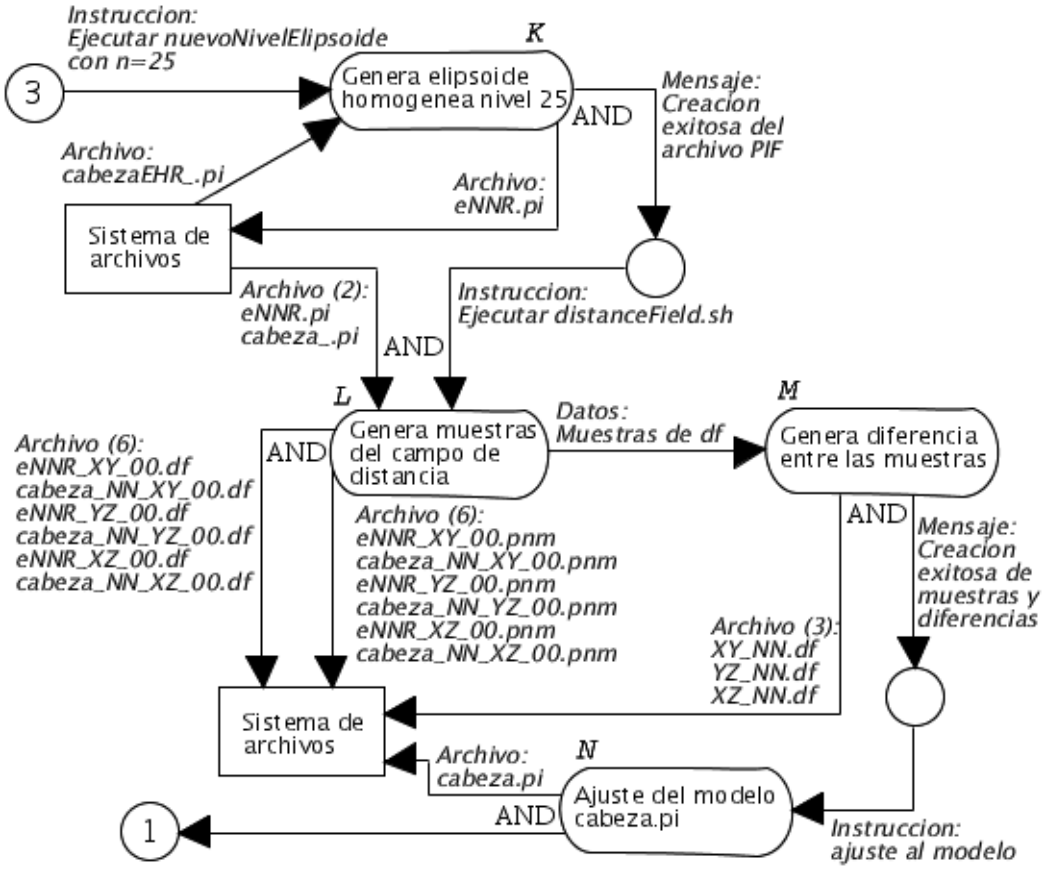


Figura 4.21. Flujo 3, medición de error y corrección de puntos externos.

14 Éste criterio es independiente al criterio de similitud para validar a la elipsoide equivalente.

Con el afán de minimizar a dicha diferencia (en la búsqueda del caso ideal), parte del proceso iterativo consiste en ajustar a los *puntos externos* (outliers) presentes en los modelos cabeza (por ejemplo, los componentes de la nariz) mediante transformaciones de traslación, para así dar pié a la re-alineación geométrica de mejor ajuste (inicio de una nueva iteración, en el flujo 1) y así, obtener una nueva elipsoide equivalente que causa menor diferencia entre las muestras de los campos de distancia correspondientes.

Detalle en las transformaciones de flujos de datos empleadas en el Flujo 3.

- *(K) Genera elipsoide homogénea nivel 25.* Creación de una elipsoide equivalente a mayor escala discreta (5000 triángulos).
- *(L) Genera muestras del campo de distancia.* Mediante el script *distanceField.sh*, se obtienen tres muestras de cada uno de los dos campos de distancia involucrados.
- *(M) Genera diferencias entre muestras.* Obtiene al criterio de similitud para asignar valores numéricos al error morfológico. Mediante herramientas independientes, se hicieron los ajustes de mínimos cuadrados para obtener a la recta representativa del error.
- *(N) Ajuste del modelo cabeza.pi.* A través de distintos programas creados a partir del modelo PIF, se aplicaron criterios de ajuste con la traslación apropiada de los puntos externos. Con apoyo visual mediante las imágenes de los campos de distancia y el programa Visor PIF 3D, se decidió el criterio de ajuste de cada modelo. Para finalizar la iteración, se almacena a la versión de iteración del modelo cabeza y se da origen a la nueva versión que ingresa al flujo 1 para repetir al proceso.

Observación: Las transformaciones ocurren en diversos programas,
Campos de distancia: Programa dfields.
Elipsoides y ajustes a modelos: Herramientas PIF.
Ajuste de mínimos cuadrados: Gnumeric ó Calc (OppenOffice).

4.7 Estimación de errores morfológicos.

Ante la comparación de modelos 3D, los campos de distancia proporcionan una ventaja crucial, ya que generan un patrón homogéneo: permite representarlos en un espacio real unidimensional, lo cual facilita su comparación, reduciendo de 3 variables (vértices) a 1 variable (distancia) para cada elemento (triángulo) del modelo 3D. Las diferencias entre las muestras de campos de distancia son el antesala de la estimación de errores morfológicos, ya que para poder generar a una expresión analítica que le represente, decidimos emplear un método matemático previamente explicado en el capítulo II (*sección 2.3*): *regresión lineal mediante mínimos cuadrados.*

La métrica de los modelos de cabeza humana está en milímetros, por lo cual el programa que calcula la muestra del campo de distancia *(dfields),* devuelve distancias en milímetros, con 10 dígitos decimales adicionales. Cada muestra del campo de distancia es un conjunto de 8100 distancias obtenidas de un área cuadrada (300X300[mm^2]) de uno de los planos del campo; dicha área está definida en el intervalo de (-150.0, ... , 150.0) [mm], tanto para el eje horizontal como para el eje vertical, en escala de 3:10 milímetros. Es posible inspeccionar gráficamente a la muestra, gracias a su imagen (en escala de grises) de 90X90[pixels], que también es creada por *dfields.* En cada iteración, el criterio de similitud consiste en generar a tres muestras (plano axial, plano coronal y plano sagital: *fig.4.22*) del campo de distancia

de (a) el modelo de cabeza humana y de (b) su modelo de elipsoide equivalente, y a cada par de muestras cabeza-elipsoide aplicar:

1) Evaluación de N_D^{Δ} = 8100 diferencias entre las distancias. *Consiste en la resta matemática ordinaria entre cada uno de los N_D^{Δ} dúos de distancias (cabeza-elipsoide).*
2) Al conjunto de N_D^{Δ} diferencias, aplicar regresión lineal mediante mínimos cuadrados.
   *a) Establecer al conjunto de diferencias de distancia como la variable aleatoria (**y**).*
   b) Enumerar [1, ... , N_D^{Δ}] *a la variable aleatoria para generar la variable controlada (**x**).*
   c) Generar a la curva de regresión -la línea recta- mediante las ecuaciones siguientes:

$$y-\bar{y}=k_1(x-\bar{x}) \quad \text{...(curva de regresión).}$$

Donde:

$$\bar{y}=\frac{1}{n}(y_1+\ldots+y_n),\ \bar{x}=\frac{1}{n}(x_1+\ldots+x_n) \quad \text{...(medias de } x, y\text{).}$$

$$k_1=\frac{n\sum x_j y_j-\sum x_i \sum y_j}{n(n-1)s_x{}^2} \quad \text{...(coeficiente de regresión).}$$

$$s_x{}^2=\frac{1}{n-1}\left[\sum_{j=1}^{n} x_j{}^2-\frac{1}{n}\left(\sum_{j=1}^{n} x_j\right)^2\right] \quad \text{...(varianza de los valores } x\text{).}$$

Nota: Se asume que n= N_D^{Δ} **.**

3) Validar al resultado de la estimación analítica de errores morfológicos.
   a) En caso de satisfacer la tolerancia de error, finaliza el proceso iterativo y se concluye con la obtención del modelo de elipsoide equivalente de mejor ajuste al modelo de cabeza humana.
   b) En caso de no satisfacer a la tolerancia de error, mediante las herramientas PIF se aplica ajuste a los puntos externos (supresión de outliers) del modelo de cabeza humana [15], con lo cual se prepara para una nueva iteración.

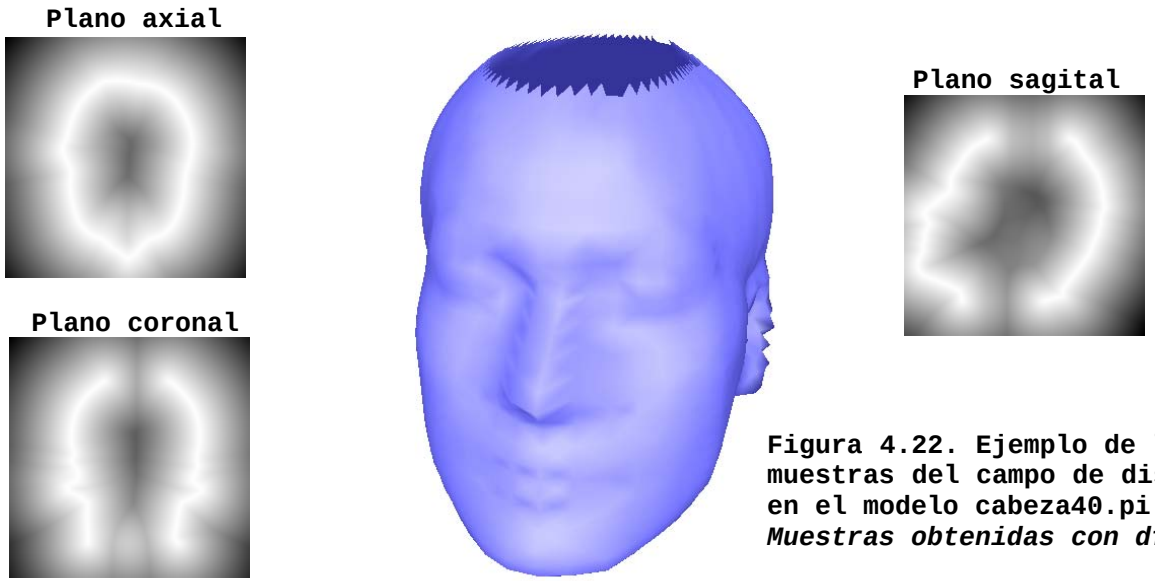


Figura 4.22. Ejemplo de las tres muestras del campo de distancia, en el modelo cabeza40.pi. *Muestras obtenidas con dfields.*

15 El ajuste apropiado a los puntos externos, depende de cada modelo de cabeza, por lo cual el criterio se analizó de forma independiente.

CAPÍTULO V

Resultados y discusión

5.1 Pruebas al software desarrollado.

5.2 Ejemplo del proceso iterativo.

5.3 Resumen de resultados y conclusiones.

Yo sólo sé que no sé nada
Sócrates

5.1 Pruebas al software desarrollado.

En el proyecto, las pruebas al software desarrollado se clasifican en dos grupos: pruebas de funcionamiento y pruebas de compatibilidad.
En los capítulos previos (en especial en el de diseño) se describe el camino que debe seguir el proceso iterativo. Además del análisis conceptual del proyecto, un apoyo implícito en el desarrollo de software son las *pruebas de funcionamiento,* que fueron aplicadas dentro del ajuste robusto de modelos elipsoidales. Gracias a las pruebas de funcionamiento, se implementaron mejoras en los programas desarrollados (por ejemplo, en el programa *dfields:* la subdivisión del dominio en "27+1" zonas redujo el tiempo necesario para obtener las muestras del campo de distancia), así como apoyo en la toma de decisiones (por ejemplo, en el programa *Visor3D:* se comprobó gráficamente el por qué usar a la elipsoide equivalente en lugar de la elipsoide de inercia). Como punto adicional, las pruebas motivaron la creación de seis *shell-scripts sh*[16], que buscan (1) simplificar la invocación de la mayoría de los programas desarrollados (al minimizar la cantidad de argumentos que debe ingresar el usuario) y (2) establecer el orden de los pasos de toda iteración, el cual es el siguiente:

1) **abc.sh.** Solicita un parámetro (# del modelo). Invoca al Visor3D, lo cual permite seguir al flujo 1 y flujo 2 (ver sección 4.4).
2) **e25.sh.** Solicita dos parámetros (# del modelo y # de iteración). Crea a la elipsoide equivalente nivel 25 del flujo 3 (ver sección 4.6).
3) **max_ev.sh.** Solicita dos parámetros (# del modelo y # de iteración). Invoca al Visor3D para poder trasladar a la elipsoide equivalente y hacer coincidir el máximo punto del casquete del modelo, con el eigenvalor correspondiente de la elipsoide (flujo 3, parte de la creación del modelo elipsoide nivel 25).
4) **distanceField.sh.** Solicita dos parámetros (# del modelo y # de iteración). Obtiene las seis muestras de campos de distancia, imprescindibles para la medición del error (flujo 3).
5) **sup_out.sh.** Solicita dos parámetros (# del modelo y # de iteración). Mueve los archivos de iteración a su carpeta correspondiente (*cNN* ó *eNN*, sección 5.2), para preparar el espacio de supresión de outliers (puntos alejados).
6) ***aproximaElipsoideX0Y0Z0ConPlano.*** *No es un shell-script.* Se trata de un programa que aplica el criterio de supresión de outliers (ver apéndice E, punto 2).
7) **end.sh.** Solicita dos parámetros (# del modelo y # de iteración). Respalda a los archivos de la versión alineada del modelo de cabeza, en la iteración correspondiente, abriendo paso a la siguiente iteración.

En el caso de las *pruebas de compatibilidad,* cabe recordar que uno de los resultados esperados fue el contar con la ejecución de los programas desarrollados, en distintas distribuciones del sistema operativo Linux (sin necesidad de recompilarlos), las cuales cuentan con la implementación el sistema X11. Este tipo de pruebas se efectuaron de manera exitosa en las siguientes distribuciones estándar: *RedHat 7.0, Fedora 3, Mandrake 10.0.*
Es preciso señalar que también hicimos pruebas de compatibilidad con un tipo especial de distribuciones Linux, conocidas como *LiveCD,* las cuales evitan la necesidad de contar con el sistema operativo Linux instalado en la computadora: permiten ejecutar a los programas desarrollados, para exponer a los datos e información del proyecto, sin tener que sustituir al sistema operativo de la computadora y sin afectar al contenido de su(s) disco(s) duro(s).

16 Bourne shell, mejor conocido como bin/sh, es uno de los intérpretes de consola (shell) más populares en las distribuciones Unix/Linux. Además de comunicar al usuario con el sistema operativo, ofrece alto nivel de lógica de programación, lo cual permite ordenar comandos mediante shell-scripts.

Especificaciones y resultados en las pruebas de compatibilidad:

Hardware involucrado	***Distribuciones Linux LiveCD***
CPU: Intel P4 1.6GHz. 256MB de memoria RAM. Unidad de CD-ROM. Monitor analógico de 15 pulgadas. Tarjeta de video: Nvidia GeForce2 MX/MX 400. Teclado y ratón PS/2 estándar. 4 Puertos USB 1.0. Dispositivo USB de almacenamiento extraíble, capacidad de 128MB.	CDMedic 6 (700MB). Knoppix 3.9 (700MB). SLAXStandard Edition 5.0.5 (**200MB**). *La computadora empleada permite iniciar sesión de trabajo (boot) mediante la unidad CD-ROM.*
Tabla 5.0.1. Especificaciones de las pruebas de compatibilidad. Las distribuciones cuentan con implementación de X11 junto con librerías compatibles de OpenGL y GLUT. Fuente de acceso a las distribuciones: http://www.frozentech.com/content/livecd.php	

CDMedic 6 (distribución basada en Knoppix 3.7).

Resolución gráfica: *800X600 [pixels].*
X11 mediante *KDE 3.3.2.*
Permite el uso de dispositivos extraíbles USB.
Funcionamiento satisfactorio de los programas *(Visor3D, dfields, y herramientas PIF).*
Lectura correcta de información XML, mediante *Mozilla FireFox*.

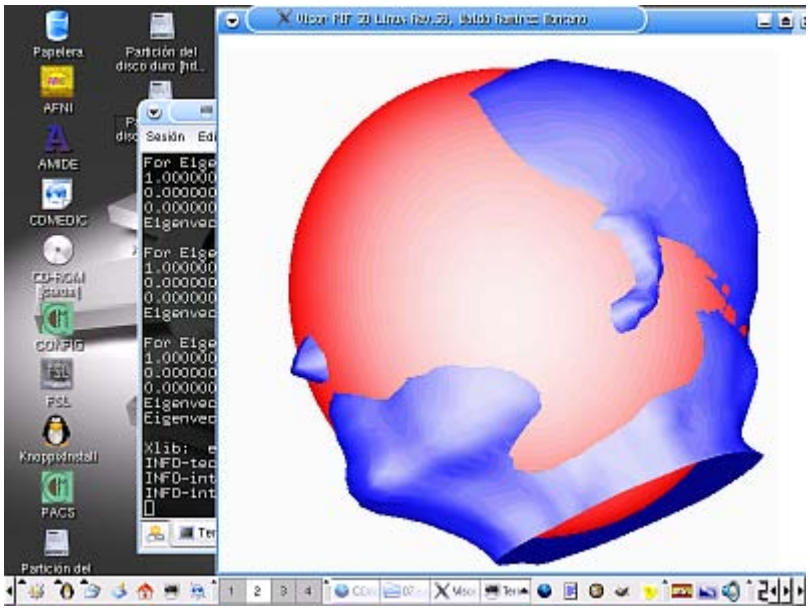


Figura 5.0.1. Ejemplo del uso del programa Visor3D en CDMedic 6. Modelo cabeza07.pi con su primer versión de elipsoide equivalente nivel 25. *Proyección Ortogonal y normales Gouraud.*

Knoppix 3.9.

Resolución gráfica: *800X600 [pixels].*
X11 mediante *KDE 3.4.0.*
Permite el uso de dispositivos extraíbles USB.

Funcionamiento satisfactorio de los programas *(Visor3D, dfields, y herramientas PIF).*
Lectura correcta de información XML, mediante *Mozilla FireFox*.

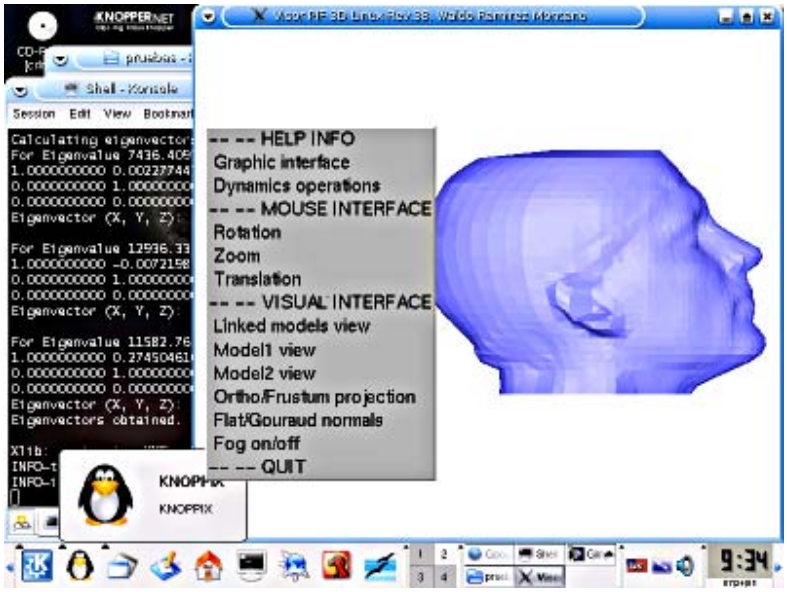


Figura 5.0.2. Ejemplo del uso del programa Visor3D en Knoppix 3.9. Modelo cabeza15.pi, con *proyección ortho y normales planas.*

SLAX Standard Edition 5.0.5.

Resolución gráfica: *1024X768 [pixels].*
X11 mediante *KDE 3.4.0.*
Permite el uso de dispositivos extraíbles USB.
Funcionamiento satisfactorio de los programas *(Visor3D, dfields, y herramientas PIF).*
Lectura correcta de información XML, mediante *Mozilla FireFox*.

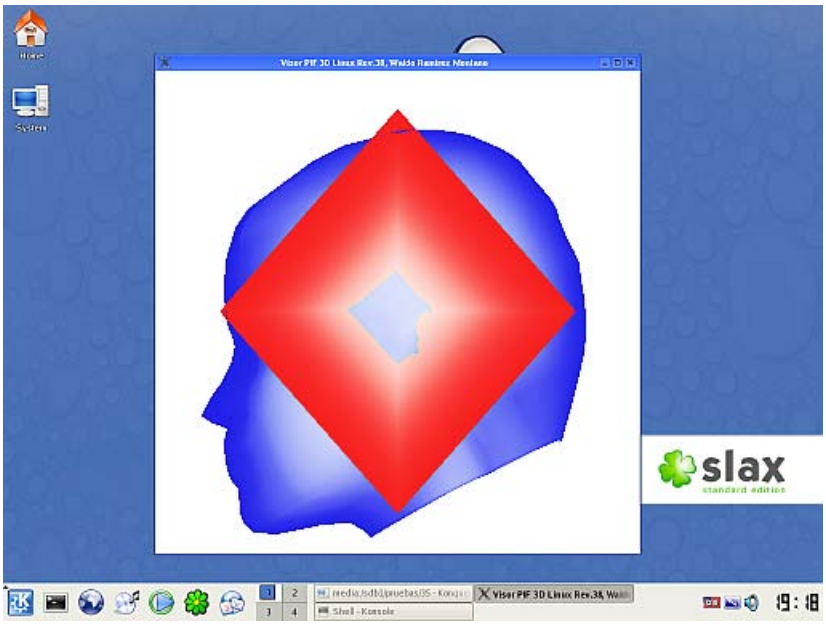


Figura 5.0.3. Ejemplo de uso del programa Visor3D en SLAX 5.0.5. Modelo cabeza35.pi con su primer versión de elipsoide equivalente nivel 1. *Proyección ortogonal con corte en el plano frontal y efecto fog.*

5.2 Ejemplo del proceso iterativo.

Antes de presentar la tabla de resultados junto con la elipsoide representativa de la población de modelos de cabezas humanas, a continuación se detalla la bitácora del proceso iterativo en un modelo de la población. Se tiene la finalidad de demostrar la funcionalidad del proceso, destacar algunos detalles en la sensibilidad del tensor de inercia y mostrar los cambios en la estimación del error morfológico ante el avance de iteraciones.

Pasos preliminares. Para el análisis de todo modelo de cabeza humana, previamente se preparó a su espacio de almacenamiento en el sistema de archivos local, mediante una carpeta nombrada con el número que identifica al modelo de cabeza, con el siguiente contenido:

- Archivo auxiliar para poder desplegar su información XML: *information.xsl.*
- Carpeta que almacena al modelo de la elipsoide equivalente (tanto nivel 1 como nivel 25) de toda iteración: *eNN.*
- Carpeta que almacena al modelo de la cabeza de toda iteración: *cNN.*
- Tres carpetas que almacenan las muestras del campo de distancia: *XY, YZ, XZ.*
- Carpeta que almacena a las hojas de cálculo con el criterio de similitud: *ERROR.*
- Carpeta (opcional) que almacena archivos auxiliares: *AUX* ó *AUX2.*

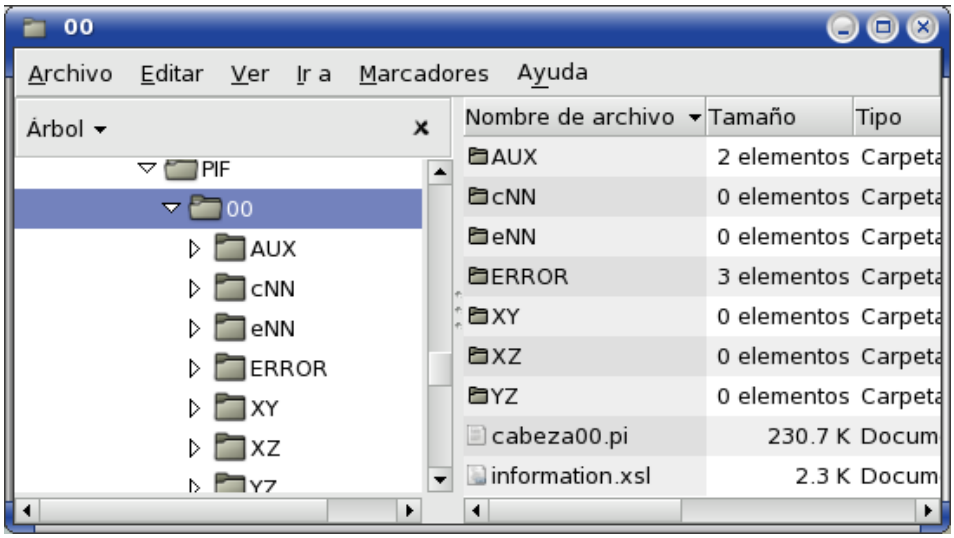


Figura 5.0.4. Espacio de almacenamiento inicial, para el modelo cabeza00.pi.
Todo programa desarrollado se encuentra en la carpeta que contiene al espacio de almacenamiento del modelo: PIF.

Los programas desarrollados que fueron usados para la obtención de resultados, consiste en el Visor3D y un conjunto de herramientas PIF:

- **cortaPlano.** Filtro de puntos, definido por un plano.
- **aproximaElipsoideX0Y0Z0ConPlano.** Aplica el criterio de supresión de outliers.
- **dfields.** Obtiene la muestra del campo de distancias.
- **nuevoNivelElipsoide.** Genera la elipsoide de nivel-N con los parámetros *a, b, c.*
- **visor.** Programa Visor3D para la alineación geométrica y despliegue de modelos PIF.

La ubicación de los programas y los seis shell-scripts fue la carpeta que guardó a todo espacio de almacenamiento de los modelos de cabeza (en la *fig.5.4,* se aprecia que es la carpeta llamada *PIF*). Cabe señalar que aplicamos un trato especial a la supresión de outliers

de los modelos de cabeza, en especial a los puntos de la nariz y el mentón (distinguibles en las muestras del plano sagital de los campos de distancia), ya que en uno de los objetivos a nivel aplicación, buscamos completar al casquete de cada modelo de cabeza. Para ello, se establece un filtro binario (un plano) a los candidatos de outliers antes de aplicarles su traslación. Como apoyo al enunciado previo, creamos un modelo PIF de un plano (*plano.pi,* 300X300 [mm], compuesto por dos triángulos), con el fin de poder apreciar la zona de supresión de outliers en el plano sagital, como se ejemplifica en la *fig.5.5.*

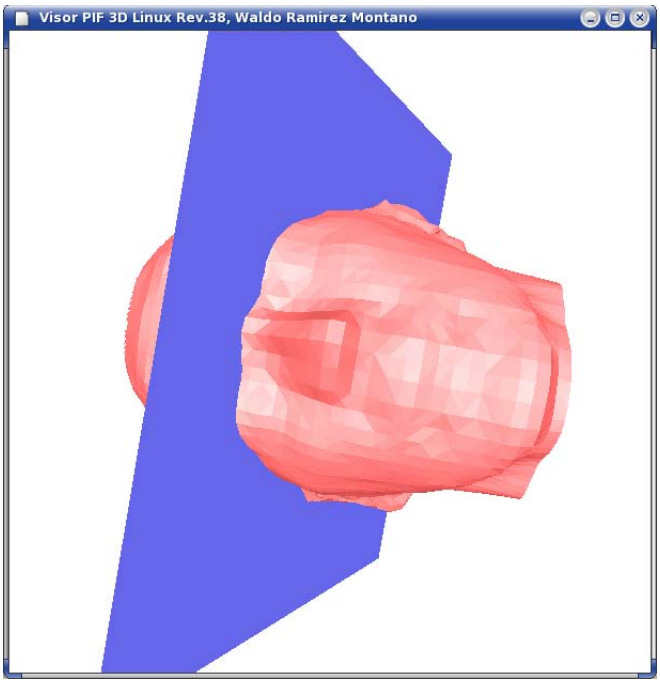


Figura 5.0.5. Modelo plano.pi ubicado en YZ, X=0.0. En este ejemplo, se observa a una posible discriminación de candidatos a outliers en el modelo cabeza03.pi. *Proyección Frustum con normales planas.*

En el *apéndice E* se encuentran las referencias de uso de los programas y shell-scripts mencionados. A continuación se presenta la bitácora detallada del proceso iterativo aplicado al modelo cabeza00.pi, en donde los comentarios de los pasos del proceso usan la simbología siguiente:

➔➔➔ Interpretación de ciertos datos obtenidos durante el proceso.
☞ Interacción con el usuario, tanto en consola como en el programa Visor3D.
✗ Indica incumplimiento de una meta del proceso ó iteración.
✓ Indica cumplimiento de una meta del proceso ó iteración.

- El texto de los símbolos se encuentra en formato tipo negrita y cursivo.
- El texto de entrada del usuario en consola se encuentra en formato tipo negrita.
- El texto de la bitácora en consola sólo presenta formato tipo negrita en datos ó información importante para la justificación de la toma de decisiones en el proceso iterativo.

☞ ***Desde consola, en la carpeta de programas, se ejecuta al shell abc.sh:***

`./abc.sh 00`

Visor3D for cabeza00.pi...
./visor ./00/cabeza00.pi
INFO-visor: Model1, stored in file "./00/cabeza00.pi".
INFO-pif: Succesfull file opening, obtaining model's name... "./00/cabeza00_", done.
INFO-pif: Max. file's line length: 94
INFO-pif: Getting points from 3D model... done.
INFO-pif: Getting indexes from 3D model... done.
WARNING-pif: Valid dynamic values of the model had not been calculated.
INFO-pif: Closing model's PIF file and freeing memory...
INFO-visor: Model1 loaded,
INFO-visor: Number of points: 3301, number of triangles: 6442
INFO-visor: Allocating memory for vectorNormal... done
INFO-visor: Allocating memory for vectorNormal... done
INFO-visor: Calculating normals... done
INFO-visor: Creating Gouraud vertex list... done
INFO-visor: Creating Gouraud triangles list... done
INFO-cuerpoRigido: Calculating geometric centroid of ./00/cabeza00_... done.
INFO-cuerpoRigido: Calculating intertia tensor of ./00/cabeza00_... done.
INFO-cuerpoRigido: Calculating eigenvalues-eigenvectors of ./00/cabeza00_...
Calculating eigenvalues from Lambda: 1.0L3 + -34060.143L2 + 379463812.508L1 + -1375559593060.709L0...
(8330.6127382153, 13482.1190071831, 12247.4114332309), eigenvalues obtained.
Calculating eigenvectors...
For Eigenvalue 8330.612738:
1.0000000000 0.0682625334 0.2135852000
0.0000000000 1.0000000000 0.3467383767
0.0000000000 0.0000000000 -0.0000000000
Eigenvector (X, Y, Z): (**-0.176615, -0.322454, 0.929963**) ➔➔➔ ***Eigenvector no alineado***

For Eigenvalue 13482.119007:
1.0000000000 -0.4519172293 -1.4139942806
0.0000000000 1.0000000000 -1.6909839561
0.0000000000 0.0000000000 -0.0000000000
Eigenvector (X, Y, Z): (**0.742585, 0.576490, 0.340920**) ➔➔➔ ***Eigenvector no alineado***

For Eigenvalue 12247.411433:
1.0000000000 0.5468364700 1.7109850895
0.0000000000 1.0000000000 -5.4550144072
0.0000000000 0.0000000000 0.0000000000
Eigenvector (X, Y, Z): (**-0.646046, 0.750788, 0.137633**) ➔➔➔ ***Eigenvector no alineado***
Eigenvectors obtained.

☞ ***Se presionó la tecla '1'***
INFO-teclado: Model1 view selected
☞ ***Se presionó la tecla 'i'***
INFO-interfaz: Model1 information:
# of points: 3301, # of triangles: 6442
Geometric centroid: (**12.7257113756, 19.1222005580, -74.4912946168**)
Inertia tensor:

12806.093344692548271	**305.507644298218509**	**955.896420301262083**
305.507644298218509	**12250.500737444248443**	**1417.196379230369530**
955.896420301262083	**1417.196379230369530**	**9003.549096492455647**

Eigenvalues: (1)8330.6127382153, (2)13482.1190071831, (3)12247.4114332309
Eigenvector (1): (X)-0.176615, (Y)-0.322454, (Z)0.929963
Eigenvector (2): (X)0.742585, (Y)0.576490, (Z)0.340920
Eigenvector (3): (X)-0.646046, (Y)0.750788, (Z)0.137633

✗ La información del modelo indica que no está alineado

☞ ***Se presionó la tecla 'G' para trasladar al modelo cabeza00.pi, tomando como argumento al vector_xyz de su centroide con signo inverso: (-12.7257113756, -19.1222005580, 74.4912946168), y así trasladar su centroide al origen***

INFO-interfaz: **Moving Model1's geometric centroid to origin... done.** Updating rigid body data...
INFO-cuerpoRigido: Calculating geometric centroid of ./00/cabeza00_... done.
INFO-cuerpoRigido: Calculating intertia tensor of ./00/cabeza00_... done.
INFO-cuerpoRigido: Calculating eigenvalues-eigenvectors of ./00/cabeza00_...
Calculating eigenvalues from Lambda: 1.0L3 + -21907.033L2 + 158607378.390L1 + -379436149311.818L0...
(6139.1343107871, 8475.9981623710, 7291.9001897084), eigenvalues obtained.
Calculating eigenvectors...
For Eigenvalue 6139.134311:
1.0000000000 0.7295182679 0.0105559000
0.0000000000 1.0000000000 -179.2696384995
0.0000000000 0.0000000000 -0.0000001215
Eigenvector (X, Y, Z): (**-1.000000, -1.000000, -1.000000**) ➔➔➔ ***Eigenvector no alineado***

For Eigenvalue 8475.998162:
1.0000000000 -0.3463840881 -0.0050120688
0.0000000000 1.0000000000 0.0025713024
0.0000000000 0.0000000000 -0.0000000000
Eigenvector (X, Y, Z): (**0.004121, -0.002571, 0.999988**) ➔➔➔ ***Eigenvector no alineado***

For Eigenvalue 7291.900190:
1.0000000000 -1.3706944702 -0.0198335180
0.0000000000 1.0000000000 324.9242077213
0.0000000000 0.0000000000 -0.0000072936
Eigenvector (X, Y, Z): (**-1.000000, -1.000000, -1.000000**) ➔➔➔ ***Eigenvector no alineado***
Eigenvectors obtained.

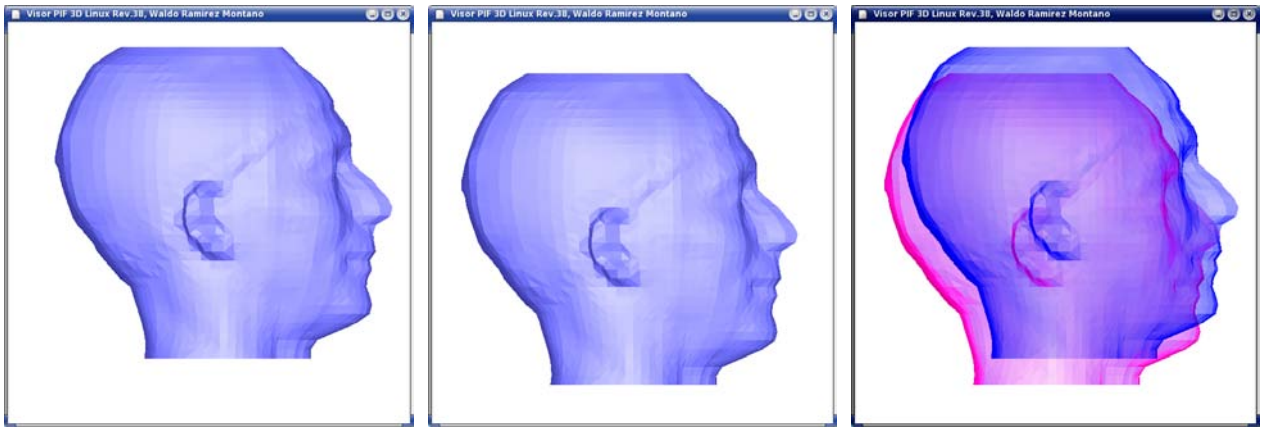


Figura 5.1.1. Antes y después de traslación en el modelo cabeza00.pi. Izquierda: Con centroide fuera del origen. Central: Con centroide en el origen. Derecha: Comparación gráfica. Proyección orto, y normales planas.

☞ ***Se presionó la tecla 'i' para verificar el centroide en el origen***

INFO-interfaz: Model1 information:
# of points: 3301, # of triangles: 6442

Geometric centroid: (**0.0000000000, -0.0000000000, -0.0000000000**)
Inertia tensor:
6891.481816827368675 548.851249466661102 7.941705010041034
548.851249466661102 6539.604033743700711 -7.241096257763307
7.941705010041034 -7.241096257763307 8475.946812295398558
Eigenvalues: (1)6139.1343107871, (2)8475.9981623710, (3)7291.9001897084
Eigenvector (1): (X)-1.000000, (Y)-1.000000, (Z)-1.000000
Eigenvector (2): (X)0.004121, (Y)-0.002571, (Z)0.999988
Eigenvector (3): (X)-1.000000, (Y)-1.000000, (Z)-1.000000

✓ ***Su centroide ya está en el origen***

☞ ***El siguiente paso es la alineación de los eigenvectores del modelo cabeza00.pi, para lo cual se inicia con la Rotación Fase A, presionando la tecla 'R' tantas veces como sea necesario (resultado en fig.5.1.2)***

INFO-interfaz: Alligning Model
----Z:-45.000000 ----Y:-45.000000 ➔➔➔ ***Rotación Fase A: (1) eje Z y (2) eje Y***
INFO-cuerpoRigido: Calculating geometric centroid of ./00/cabeza00_... done.
INFO-cuerpoRigido: Calculating intertia tensor of ./00/cabeza00_... done.
INFO-cuerpoRigido: Calculating eigenvalues-eigenvectors of ./00/cabeza00_...
Calculating eigenvalues from Lambda: 1.0L3 + -21907.033L2 + 158607378.390L1 + -379436149311.818L0...
(6139.1343107870, 8475.9981623709, 7291.9001897085), eigenvalues obtained.
Calculating eigenvectors...
For Eigenvalue 6139.134311:
1.0000000000 -0.0675027045 -0.3500503002
0.0000000000 1.0000000000 -8.7886795233
0.0000000000 0.0000000000 0.0000000000
Eigenvector (X, Y, Z): (**0.106043, 0.987987, 0.112416**) ➔➔➔ ***Eigenvector no alineado***

For Eigenvalue 8475.998162:
1.0000000000 0.1926632643 0.9990982443
0.0000000000 1.0000000000 0.0066853852
0.0000000000 0.0000000000 0.0000000000
Eigenvector (X, Y, Z): (**-0.706323, -0.004732, 0.707873**) ➔➔➔ ***Eigenvector no alineado***

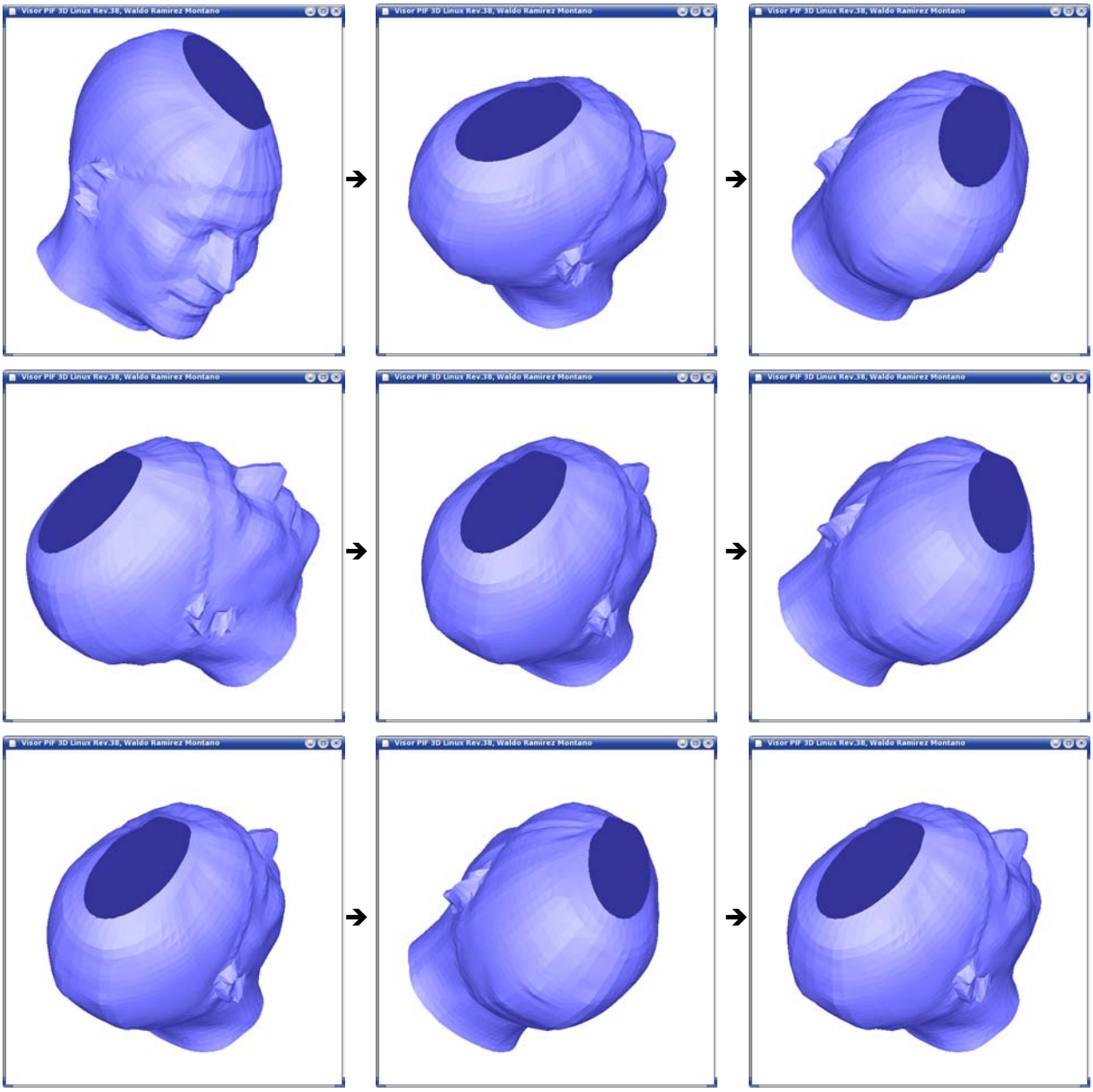
For Eigenvalue 7291.900190:
1.0000000000 -0.2021828621 -1.0484642379
0.0000000000 1.0000000000 0.2215099667
0.0000000000 0.0000000000 -0.0000000000
Eigenvector (X, Y, Z): (**0.699902, -0.154467, 0.697336**) ➔➔➔ ***Eigenvector no alineado***
Eigenvectors obtained.

INFO-interfaz: Alligning Model
----Z:83.873761 ----Y:46.670953 ➔➔➔ ***Rotación Fase A: (1) eje Z y (2) eje Y***
INFO-cuerpoRigido: Calculating geometric centroid of ./00/cabeza00_... done.
INFO-cuerpoRigido: Calculating intertia tensor of ./00/cabeza00_... done.
INFO-cuerpoRigido: Calculating eigenvalues-eigenvectors of ./00/cabeza00_...
Calculating eigenvalues from Lambda: 1.0L3 + -21907.033L2 + 158607378.390L1 + -379436149311.820L0...
(6139.1343107870, 8475.9981623709, 7291.9001897085), eigenvalues obtained.
Calculating eigenvectors...
For Eigenvalue 6139.134311:
1.0000000000 -0.2420651521 0.8112160504
0.0000000000 1.0000000000 -0.2691486464
0.0000000000 0.0000000000 0.0000000000
Eigenvector (X, Y, Z): (**-0.584533, 0.210875, 0.783488**) ➔➔➔ ***Eigenvector no alineado***

For Eigenvalue 8475.998162:
1.0000000000 0.1863569217 -0.6245249457
0.0000000000 1.0000000000 1.3083939398
0.0000000000 0.0000000000 0.0000000000
Eigenvector (X, Y, Z): (**0.466430, -0.702795, 0.537143**) ➔➔➔ ***Eigenvector no alineado***

For Eigenvalue 7291.900190:
1.0000000000 1.8057188671 -6.0513796165
0.0000000000 1.0000000000 -2.1745043247
0.0000000000 0.0000000000 -0.0000000000
Eigenvector (X, Y, Z): (**0.663901, 0.679420, 0.312448**) ➔➔➔ ***Eigenvector no alineado***
Eigenvectors obtained.

✓ ***Se omite la impresión del resto de Rotaciones Fase A, resumiendo sus resultados en la fig.5.1.2: cuando se logra alinear al eigenvector X***



(fig. continua ...)

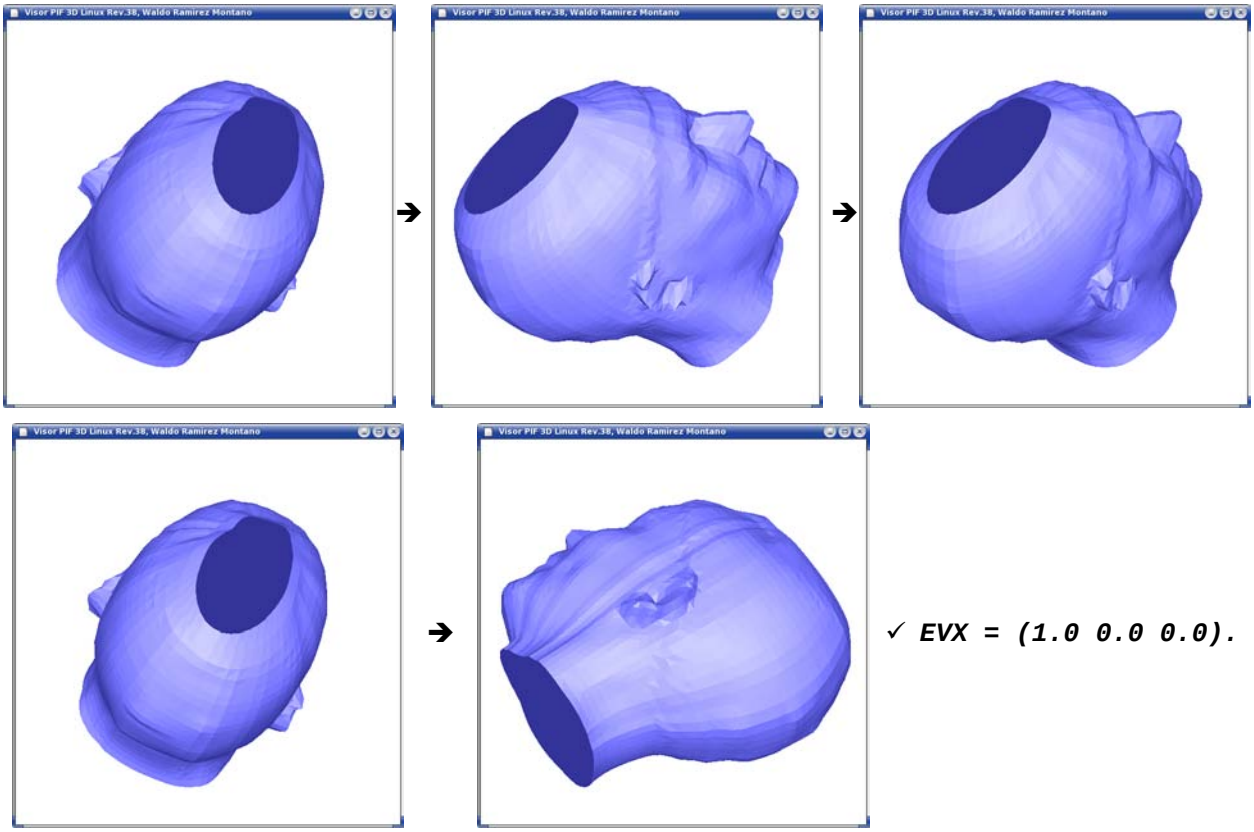


Figura 5.1.2. Resultados de la Rotación Fase A en el modelo cabeza00.pi. El algoritmo de la Rotación Fase A busca que el eigenvector X (EVX) converja a (1.0, 0.0, 0.0); en este modelo particular, se requirieron 16 Rotaciones Fase A para lograr el objetivo, cantidad que varía dependiendo de cada modelo. Proyección ortogonal y normales planas.

☞ ***Se presionó la tecla 'i' para verificar que el eigenvector X esta alineado***

INFO-interfaz: Model1 information:
# of points: 3301, # of triangles: 6442
Geometric centroid: (0.0000000000, 0.0000000000, -0.0000000000)
Inertia tensor:
6139.134310787046161 -0.000000000003800 -0.000000000001985
-0.000000000003800 7591.258594924716817 514.639899291126881
-0.000000000001985 514.639899291126881 8176.639757154727704
Eigenvalues: (1)6139.1343107870, (2)8475.9981623709, (3)7291.9001897085
Eigenvector (1): (X)1.000000, (Y)0.000000, (Z)0.000000 ➔➔➔ ***Eigenvector X alineado***
Eigenvector (2): (X)-0.000000, (Y)0.502808, (Z)0.864398
Eigenvector (3): (X)0.000000, (Y)-0.864398, (Z)0.502808

☞ ***A continuación, se debe llevar a cabo la Rotación Fase B, presionando la tecla 'T' tantas veces como sea necesario (resultado en fig.5.1.3)***

INFO-interfaz: Alligning Model
----X:-59.814070 ➔➔➔ ***Rotación Fase B: eje X.***
INFO-cuerpoRigido: Calculating geometric centroid of ./00/cabeza00_... done.
INFO-cuerpoRigido: Calculating intertia tensor of ./00/cabeza00_... done.
INFO-cuerpoRigido: Calculating eigenvalues-eigenvectors of ./00/cabeza00_...
Calculating eigenvalues from Lambda: 1.0L3 + -21907.033L2 + 158607378.390L1 + -379436149311.819L0...
(6139.1343107870, 8475.9981623709, 7291.9001897086), eigenvalues obtained.
Calculating eigenvectors...

For Eigenvalue 6139.134311:
0.0000000000 -0.0000000000 0.0000000000
-0.0000000000 2336.8638515839 0.0000000000
0.0000000000 0.0000000000 1152.7658789215
Eigenvector (X, Y, Z): (**1.000000, 0.000000, 0.000000**) ➔➔➔ ***Eigenvector X alineado***

For Eigenvalue 8475.998162:
-2336.8638515839 -0.0000000000 0.0000000000
-0.0000000000 0.0000000000 0.0000000000
0.0000000000 0.0000000000 -1184.0979726624
Eigenvector (X, Y, Z): (**0.000000, 1.000000, 0.000000**) ➔➔➔ ***Eigenvector Y alineado***

For Eigenvalue 7291.900190:
-1152.7658789215 -0.0000000000 0.0000000000
-0.0000000000 1184.0979726624 0.0000000000
0.0000000000 0.0000000000 -0.0000000001
Eigenvector (X, Y, Z): (**0.000000, 0.000000, 1.000000**) ➔➔➔ ***Eigenvector Z alineado***
Eigenvectors obtained.

✓ ***Sus eigenvectores X, Y, Z ya están alineados***

☞ ***Debido a que el rostro del modelo de cabeza no es visible en el plano XY (fig.5.1.3 izquierda), se aplicó una rotación de 180º sobre el eje Y al presionar a la tecla '8' en dos ocasiones (fig.5.1.3, centro y derecha). Lo anterior no altera a la alineación de los eigenvectores. Se presionó la tecla '8'***

INFO-interfaz: Model Rotation, 90 degrees over Yaxis.
INFO-cuerpoRigido: Calculating geometric centroid of ./00/cabeza00_... done.
INFO-cuerpoRigido: Calculating intertia tensor of ./00/cabeza00_... done.
INFO-cuerpoRigido: Calculating eigenvalues-eigenvectors of ./00/cabeza00_...
Calculating eigenvalues from Lambda: 1.0L3 + -21907.033L2 + 158607378.390L1 + -379436149311.819L0...
(6139.1343107870, 8475.9981623709, 7291.9001897086), eigenvalues obtained.
Calculating eigenvectors...
For Eigenvalue 6139.134311:
1152.7658789215 0.0000000000 -0.0000000000
0.0000000000 2336.8638515839 0.0000000000
-0.0000000000 0.0000000000 0.0000000000
Eigenvector (X, Y, Z): (**0.000000, 0.000000, 1.000000**) ➔➔➔ ***Eigenvector Z alineado***

For Eigenvalue 8475.998162:
-1184.0979726624 0.0000000000 -0.0000000000
0.0000000000 0.0000000000 0.0000000000
-0.0000000000 0.0000000000 -2336.8638515839
Eigenvector (X, Y, Z): (**0.000000, 1.000000, 0.000000**) ➔➔➔ ***Eigenvector Y alineado***

For Eigenvalue 7291.900190:
-0.0000000001 0.0000000000 -0.0000000000
0.0000000000 1184.0979726624 0.0000000000
-0.0000000000 0.0000000000 -1152.7658789215
Eigenvector (X, Y, Z): (**1.000000, 0.000000, 0.000000**) ➔➔➔ ***Eigenvector X alineado***
Eigenvectors obtained.

☞ ***Se presionó la tecla '8'***
INFO-interfaz: Model Rotation, 90 degrees over Yaxis.
INFO-cuerpoRigido: Calculating geometric centroid of ./00/cabeza00_... done.
INFO-cuerpoRigido: Calculating intertia tensor of ./00/cabeza00_... done.
INFO-cuerpoRigido: Calculating eigenvalues-eigenvectors of ./00/cabeza00_...
Calculating eigenvalues from Lambda: 1.0L3 + -21907.033L2 + 158607378.390L1 + -379436149311.819L0...
(6139.1343107870, 8475.9981623709, 7291.9001897086), eigenvalues obtained.

Calculating eigenvectors...
For Eigenvalue 6139.134311:
0.0000000000 0.0000000000 0.0000000000
0.0000000000 2336.8638515839 -0.0000000000
0.0000000000 -0.0000000000 1152.7658789215
Eigenvector (X, Y, Z): (**1.000000, 0.000000, 0.000000**) ➔➔➔ ***Eigenvector X alineado***

For Eigenvalue 8475.998162:
-2336.8638515839 0.0000000000 0.0000000000
0.0000000000 0.0000000000 -0.0000000000
0.0000000000 -0.0000000000 -1184.0979726624
Eigenvector (X, Y, Z): (**0.000000, 1.000000, 0.000000**) ➔➔➔ ***Eigenvector Y alineado***

For Eigenvalue 7291.900190:
-1152.7658789215 0.0000000000 0.0000000000
0.0000000000 1184.0979726624 -0.0000000000
0.0000000000 -0.0000000000 -0.0000000001
Eigenvector (X, Y, Z): (**0.000000, 0.000000, 1.000000**) ➔➔➔ ***Eigenvector Y alineado***
Eigenvectors obtained.

✓ ***Sus eigenvectores X, Y, Z se mantienen alineados***

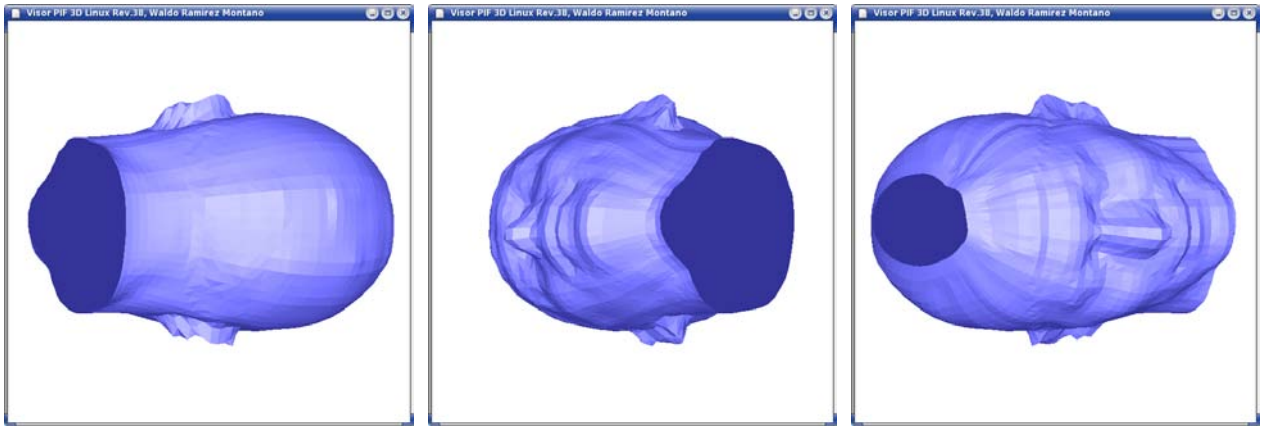


Figura 5.1.3. Resultado de la Rotación Fase B.
Izquierda: Con su primer aplicación se alinearon los eigenvectores Y y Z del modelo cabeza00.pi. Con la finalidad de mantener un estándar y visualizar al rostro del modelo de cabeza en el plano XY, se le rotó en 90º (Centro) y 90º (Derecha), con un total de 180º sobre el eje Y, respetando la alineación de sus eigenvectores. Proyección ortogonal y normales planas.

☞ ***Se presionó la tecla 'i'***
INFO-interfaz: Model1 information:
# of points: 3301, # of triangles: 6442
Geometric centroid: (**-0.0000000000, 0.0000000000, 0.0000000000**)
Inertia tensor:

6139.134310787047980	**0.000000000002994**	**0.000000000005065**
0.000000000002994	**8475.998162370946375**	**-0.000000000001256**
0.000000000005065	**-0.000000000001256**	**7291.900189708496328**

Eigenvalues: **(1)6139.1343107870, (2)8475.9981623709, (3)7291.9001897086** ➔➔➔ ***Eigenvalores***
Eigenvector (1): **(X)1.000000, (Y)0.000000, (Z)0.000000**
Eigenvector (2): **(X)0.000000, (Y)1.000000, (Z)0.000000**
Eigenvector (3): **(X)0.000000, (Y)0.000000, (Z)1.000000**

✓ ***La información del modelo cabeza00.pi, indica que está alineado***

☞ ***Es necesario almacenar la versión del modelo alineado, en la primera iteración, por lo cual se presionó la tecla 'o'***

INFO-exporta: Creating model's XML file: "./00/cabeza00_.xml"... finished.
INFO-exporta: Creating the PIF file: "./00/cabeza00_.pi"... finished.

☞ ***El siguiente paso consiste en crear la elipsoide homogénea del modelo, para lo cual es viable consultar la ayuda de operaciones mecánicas (en caso de ser necesario), mediante la tecla 'H':***

Visor PIF 3D v2.4, dynamics operations help.
ABOUT: Waldo Ramirez Montano, Thesis 2005.
h: help info with visual interface.
i or I: model's info (must be Model1 or Model2 view).
+, -: Sets (+)increment or (-)decrement to be applied in some transformations.
G: Model1's or Model2's centroid to origin [0,0,0].
R, T: Phase A and phase B rotation criteria to allign Model1 or Model2.
7, 8, 9: Model1 or Model2 rotation of 90 degrees over (7)X, (8)Y or (9)Z axis.
A, B, C: Scale Model2 to Model1; principal axis: eigenvalue A or B or C (must be Linked models view).
J, K, L: Translate Model2 to (+)max or (-)min Model1's (J)X or (K)Y or (L)Z (must be Linked models view).
E: Create homogeneous ellipsoid L1 (must be Linked models view with only Model1 loaded and alligned in standard mode).
o,O: Export model's 3dInfo to a XML file and model's data to (o)PIF format or (O)VRML1 format.

✓ ***Al estar alineado el modelo de forma estándar (es decir, el eigenvector [1] es X, el [2] es Y, y el [3] es Z), se procede a generar a la elipsoide homogénea:***

☞ ***Se presionó la tecla 'E' (resultado en fig.5.1.4 izquierda)***

Copyright, Waldo Ramirez Montano 2005.
Creating level-1 unit sphere... finished. Closing created file and applying the no-redundancy criteria...
Searching redundancy (each "." means one redundant point):
.....
Done with points...
Fixing Indexes for triangles:
Done with indexes...
Closing files and freeing memory.
Finished, created no-redundancy ellipsoid's file.
INFO-cuerpoRigido: Removing redundant model...
INFO-pif: Succesfull file opening, obtaining model's name... "./00/cabeza00EHR_", done.
INFO-pif: Max. file's line length: 53
INFO-pif: Getting points from 3D model... done.
INFO-pif: Getting indexes from 3D model... done.
WARNING-pif: Valid dynamic values of the model had not been calculated.
INFO-pif: Closing model's PIF file and freeing memory...
INFO-cuerpoRigido: Calculating geometric centroid of ./00/cabeza00EHR_... done.
INFO-cuerpoRigido: Calculating intertia tensor of ./00/cabeza00EHR_... done.
INFO-cuerpoRigido: Calculating eigenvalues-eigenvectors of ./00/cabeza00EHR_...
Calculating eigenvalues from Lambda: 1.0L3 + -36511.721L2 + 440576051.084L1 + -1756648839406.107L0...
(10231.8905179808, 14126.6636039461, 12153.1669828461), eigenvalues obtained.
Calculating eigenvectors...
For Eigenvalue 10231.890518:
0.0000000000 -0.0000000000 -0.0000000000
-0.0000000000 3894.7730859653 -0.0000000000
-0.0000000000 -0.0000000000 1921.2764648652
Eigenvector (X, Y, Z): (**1.000000, 0.000000, 0.000000**) ➔➔➔ ***Eigenvector X alineado***

For Eigenvalue 14126.663604:
-3894.7730859652 -0.0000000000 -0.0000000000
-0.0000000000 0.0000000001 -0.0000000000

-0.0000000000 -0.0000000000 -1973.4966211000
Eigenvector (X, Y, Z): (**0.000000, 1.000000, 0.000000**) ➔➔➔ ***Eigenvector Y alineado***

For Eigenvalue 12153.166983:
-1921.2764648653 -0.0000000000 -0.0000000000
-0.0000000000 1973.4966211000 -0.0000000000
-0.0000000000 -0.0000000000 -0.0000000001
Eigenvector (X, Y, Z): (**0.000000, 0.000000, 1.000000**) ➔➔➔ ***Eigenvector Z alineado***
Eigenvectors obtained.

INFO-visor: Allocating memory for vectorNormal... done
INFO-visor: Allocating memory for vectorNormal... done
INFO-visor: Calculating normals... done
INFO-visor: Creating Gouraud vertex list... done
INFO-visor: Creating Gouraud triangles list... done

✓ ***Es necesario que el modelo de elipsoide sea equivalente, por lo cual su tensor de inercia debe ser equivalente al del modelo de cabeza, lo cual se logra mediante la escala uniforme aplicada al modelo de la elipsoide homogénea:***

☞ ***Se presionó la tecla 'A' (resultado en fig.5.1.4 derecha)***
INFO-cuerpoRigido: Calculating intertia tensor of ./00/cabeza00EHR_... done.
INFO-cuerpoRigido: Calculating intertia tensor of ./00/cabeza00EHR_... done.

⋮ ➔➔➔ ***Se omite la repetición del mismo aviso [INFO]***

INFO-cuerpoRigido: Calculating intertia tensor of ./00/cabeza00EHR_... done.
INFO-cuerpoRigido: Calculating intertia tensor of ./00/cabeza00EHR_... done.
Calculating eigenvalues from Lambda: 1.0L3 + -21907.034L2 + 158607395.171L1 + -379436209528.843L0...
(6139.1346355520, 8475.9986107525, 7291.9005754530), eigenvalues obtained.
Calculating eigenvectors...
For Eigenvalue 6139.134636:
0.0000000000 -0.0000000000 -0.0000000000
-0.0000000000 2336.8639752005 -0.0000000000
-0.0000000000 -0.0000000000 1152.7659399010
Eigenvector (X, Y, Z): (1.000000, 0.000000, 0.000000)

For Eigenvalue 8475.998611:
-2336.8639752005 -0.0000000000 -0.0000000000
-0.0000000000 -0.0000000000 -0.0000000000
-0.0000000000 -0.0000000000 -1184.0980352995
Eigenvector (X, Y, Z): (0.000000, 1.000000, 0.000000)

For Eigenvalue 7291.900575:
-1152.7659399010 -0.0000000000 -0.0000000000
-0.0000000000 1184.0980352995 -0.0000000000
-0.0000000000 -0.0000000000 0.0000000000
Eigenvector (X, Y, Z): (0.000000, 0.000000, 1.000000)
Eigenvectors obtained.

INFO-interfaz: Eigenvalue A difference: -0.000325 ➔➔➔ ***Tolerancia válida (tres decimales)***[17]

17 La tolerancia se refiere a la diferencia entre el eigenvalor A del modelo de cabeza y el eigenvalor A del modelo de elipsoide. Se puede comprobar que, al ser resultado de la escala homogénea sobre el modelo de elipsoide homogénea, se respeta la misma tolerancia en los eigenvalores B y C (mediante las teclas 'B' y 'C', respectivamente).

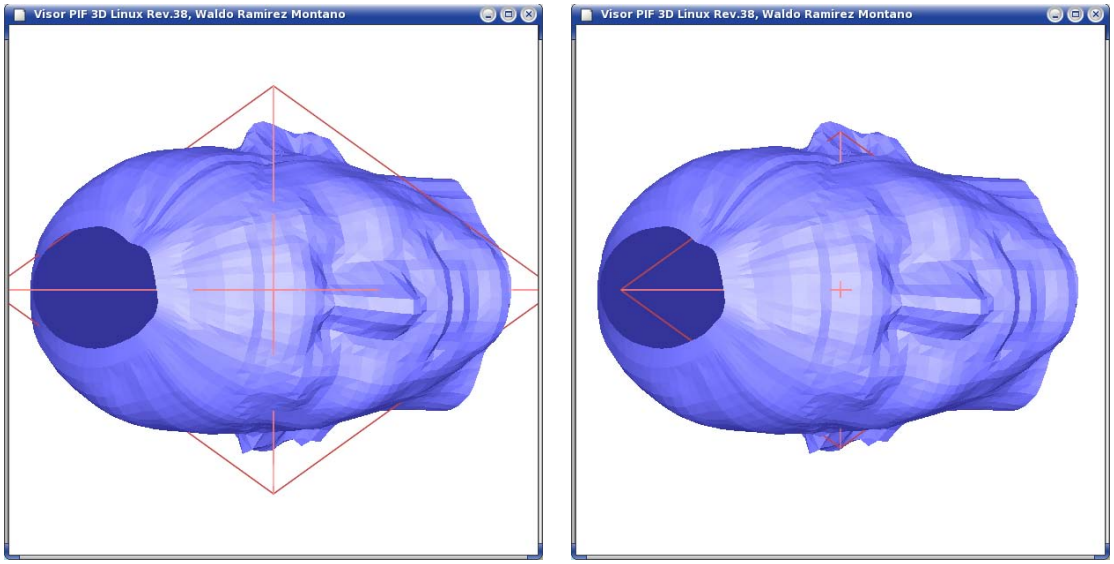


Figura 5.1.4. Obtención de la elipsoide equivalente. Izquierda: Diferencia de eigenvalores no aceptable. Derecha: Diferencia de eigenvalores aceptable. Proyección orto y normales planas.

☞ ***Se presionó la tecla '1'***
INFO-teclado: Model1 view selected
☞ ***Se presionó la tecla 'i'***
INFO-interfaz: Model1 information:
# of points: 3301, # of triangles: 6442
Geometric centroid: (-0.0000000000, 0.0000000000, 0.0000000000)
Inertia tensor:

6139.134310787047980	**0.000000000002994**	**0.000000000005065**
0.000000000002994	**8475.998162370946375**	**-0.000000000001256**
0.000000000005065	**-0.000000000001256**	**7291.900189708496328**

Eigenvalues: (1)6139.1343107870, (2)8475.9981623709, (3)7291.9001897086
Eigenvector (1): (X)1.000000, (Y)0.000000, (Z)0.000000
Eigenvector (2): (X)0.000000, (Y)1.000000, (Z)0.000000
Eigenvector (3): (X)0.000000, (Y)0.000000, (Z)1.000000

☞ ***Se presionó la tecla '2'***
INFO-teclado: Model2 view selected
☞ ***Se presionó la tecla 'i'***
INFO-interfaz: Model2 information:
# of points: 6, # of triangles: 8
Geometric centroid: (0.0000000000, 0.0000000000, 0.0000000000)
Inertia tensor:

6139.134635551993597	**-0.000000000000000**	**-0.000000000000000**
-0.000000000000000	**8475.998610752487366**	**-0.000000000000000**
-0.000000000000000	**-0.000000000000000**	**7291.900575453012607**

Eigenvalues: (1)6139.1346355520, (2)8475.9986107525, (3)7291.9005754530
Eigenvector (1): (X)1.000000, (Y)0.000000, (Z)0.000000
Eigenvector (2): (X)0.000000, (Y)1.000000, (Z)0.000000
Eigenvector (3): (X)0.000000, (Y)0.000000, (Z)1.000000

✓ ***Elipsoide equivalente obtenida, se comprobó con la información de los modelos. Es necesario almacenar a la elipsoide equivalente antes de salir del Visor3D:***

☞ ***Se presionó la tecla 'o'***
INFO-exporta: Creating model's XML file: "./00/cabeza00EHR_.xml"... finished.
INFO-exporta: Creating the PIF file: "./00/cabeza00EHR_.pi"... finished.
☞ ***Se presionó la tecla 'q'***
INFO-teclado: Quit request...
INFO-pif: Unloading info3d "./00/cabeza00EHR_"... done
INFO-pif: Unloading info3d "./00/cabeza00_"... done

☞ ***El siguiente paso es generar al modelo de elipsoide equivalente nivel 25, para lo cual desde la consola en la carpeta de programas, se ejecuta al shell e25.sh:***
`./e25.sh 00 01`
Generating level25 ellipsoid...
./nuevoNivelElipsoide ./00/cabeza00EHR_.pi ./00/e00_01.pi 25
INFO-nuevoNivelElipsoide: PIF Model, stored in file "./00/cabeza00EHR_.pi".
INFO-pif: Succesfull file opening, obtaining model's name... "./00/cabeza00EHR__", done.
INFO-pif: Max. file's line length: 42
INFO-pif: Getting points from 3D model... done.
INFO-pif: Getting indexes from 3D model... done.
WARNING-pif: Valid dynamic values of the model had not been calculated.
INFO-pif: Closing model's PIF file and freeing memory...
INFO-nuevoNivelElipsoide: **Ellipsoid's 'b' value: 86.2122665308**
INFO-nuevoNivelElipsoide: **Ellipsoid's 'a' value: 120.1796439751**
INFO-nuevoNivelElipsoide: **Ellipsoid's 'c' value: 104.8086303998**
Copyright, Waldo Ramirez Montano 2005.
Creating level-25 unit sphere... finished. Closing created file and applying the no-redundancy criteria...
Searching redundancy (each "." means one redundant point):
.....
.....
Done with points...
Fixing Indexes for triangles:
Done with indexes...
Closing files and freeing memory.
Finished, created no-redundancy ellipsoid's file.
INFO-nuevoNivelElipsoide: Succesfull new ellipsoid creation!
INFO-pif: Unloading info3d "./00/cabeza00EHR__"... done

✓ ***Se generó la elipsoide equivalente nivel 25. Es necesario recordar los valores A, B, C de la elipsoide equivalente, con el fin de poder aplicar la supresión de outliers (en caso de ser necesario).***

Elipsoide equivalente	Valor de iteración actual [mm]	Valor de iteración previa [mm]
A	***120.1796439751***	NA
B	***86.2122665308***	NA
C	***104.8086303998***	NA

Tabla 5.1.1. Obtención de elipsoide equivalente nivel 25.

A continuación, antes de la medición del error, se debe trasladar a la elipsoide equivalente a lo largo del eigenvector cuyo eigenvalor mejor se aproxime (desde el punto de vista geométrico) a completar el casquete del modelo de cabeza. Para ello se emplea al Visor3D.

☞ ***Desde consola, en la carpeta de programas se ejecuta al shell max_ev.sh:***
`./max_ev.sh 00 01`

Visor3D for cabeza00_.pi and equivalent ellipsoid...
./visor ./00/cabeza00_.pi ./00/e00_01R.pi ➔➔➔ ***Carga de los modelos:***
cabeza00_.pi (cabeza00.pi alineada)
e00_01R.pi (elipsoide equivalente nivel 25)

<Se omite la impresión de la bitácora en la consola, sobre la carga de los modelos PIF>

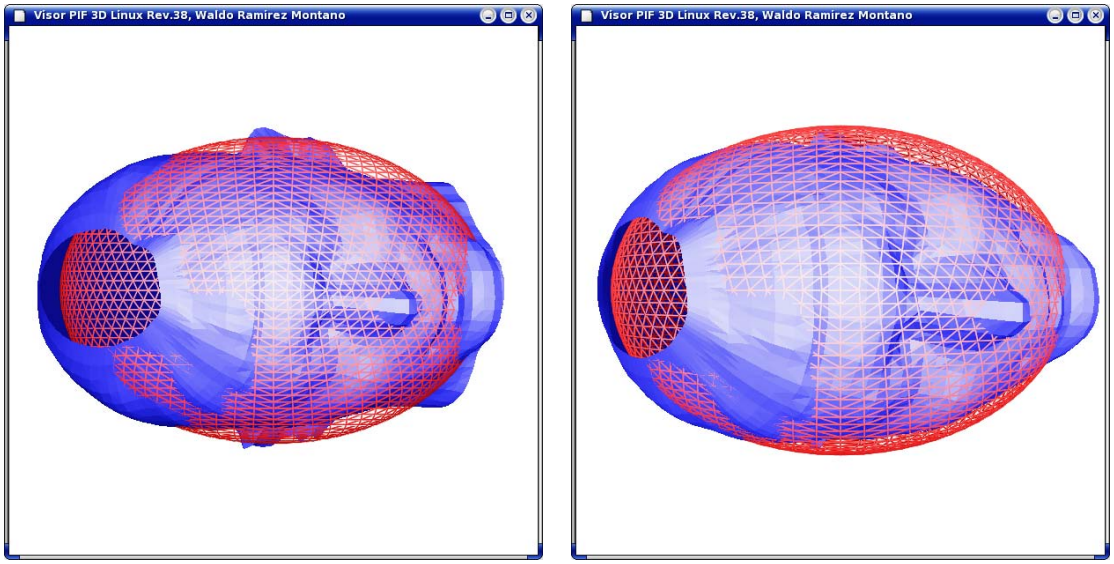


Figura 5.1.5. Plano XY del modelo de cabeza alineado (cabeza00_.pi) y su elipsoide equivalente nivel 25 (e00_01R.pi), en su primera iteración. Se observa que el eigenvalor que mejor se aproxima al casquete del modelo de cabeza, es el correspondiente al eigenvector X, en el sentido negativo. Izquierda: Proyección ortogonal. Derecha: Proyección Frustum.

☞ ***La traslación del Modelo2 en el Visor3D (la elipsoide), se efectúa (1) eligiendo al sentido de traslación (en este caso, negativo) y (2) eligiendo la trayectoria de la traslación (en este caso, el eje X): Se presionó la tecla '-' para elegir el sentido de traslación negativo***

INFO-teclado: **Attributes decrement set**

☞ ***Se mantuvo presionada a la tecla 'J' para trasladar al modelo de elipsoide equivalente a lo largo del eje X. Se dejo de presionar a dicha tecla, cuando la traslación fue cero***

INFO-interfaz: Obtaining Model1's and Model2's minimal Xvalue... (Model1, Model2)
(-133.1347642135, -120.1796439751) ➔➔➔ ***El segundo valor debería ser igual al primero***
INFO-cuerpoRigido: Calculating geometric centroid of ./00/e00_01R_... done.
INFO-cuerpoRigido: Calculating intertia tensor of ./00/e00_01R_... done.
INFO-cuerpoRigido: Calculating eigenvalues-eigenvectors of ./00/e00_01R_...
Calculating eigenvalues from Lambda: 1.0L3 + -22287.282L2 + 163512042.967L1 + -394993663402.249L0...
(6069.9094247431, 8931.3213085564, 7286.0509596845), eigenvalues obtained.
Calculating eigenvectors...
For Eigenvalue 6069.909425:
0.0000000000 -0.0000000000 0.0000000000
-0.0000000000 2861.4118838133 0.0000000000
0.0000000000 0.0000000000 1216.1415349415
Eigenvector (X, Y, Z): (1.000000, 0.000000, 0.000000)

For Eigenvalue 8931.321309:
-2861.4118838133 -0.0000000000 0.0000000000
-0.0000000000 0.0000000000 0.0000000000

0.0000000000 0.0000000000 -1645.2703488719
Eigenvector (X, Y, Z): (0.000000, 1.000000, 0.000000)

For Eigenvalue 7286.050960:
-1216.1415349415 -0.0000000000 0.0000000000
-0.0000000000 1645.2703488719 0.0000000000
0.0000000000 0.0000000000 -0.0000000000
Eigenvector (X, Y, Z): (0.000000, 0.000000, 1.000000)
Eigenvectors obtained.

INFO-interfaz: Translated Model2 with -1.2955120238 ➔➔➔ ***Valor de traslación***

⋮ ➔➔➔ ***Se omite la bitácora de consola, hasta la última traslación***

INFO-interfaz: Obtaining Model1's and Model2's minimal X value... (Model1, Model2)
(-133.1347642135, -133.1347642135) ➔➔➔ ***El segundo valor fue igual al primero***
INFO-cuerpoRigido: Calculating geometric centroid of ./00/e00_01R_... done.
INFO-cuerpoRigido: Calculating intertia tensor of ./00/e00_01R_... done.
INFO-cuerpoRigido: Calculating eigenvalues-eigenvectors of ./00/e00_01R_...
Calculating eigenvalues from Lambda: 1.0L3 + -22619.595L2 + 168251390.866L1 + -411517380738.276L0...
(6069.9094247431, 9097.4780975429, 7452.2077486710), eigenvalues obtained.
Calculating eigenvectors...
For Eigenvalue 6069.909425:
-0.0000000000 0.0000000000 0.0000000000
0.0000000000 3027.5686727998 0.0000000000
0.0000000000 0.0000000000 1382.2983239280
Eigenvector (X, Y, Z): (1.000000, 0.000000, 0.000000)

For Eigenvalue 9097.478098:
-3027.5686727998 0.0000000000 0.0000000000
0.0000000000 0.0000000000 0.0000000000
0.0000000000 0.0000000000 -1645.2703488718
Eigenvector (X, Y, Z): (0.000000, 1.000000, 0.000000)

For Eigenvalue 7452.207749:
-1382.2983239280 0.0000000000 0.0000000000
0.0000000000 1645.2703488718 0.0000000000
0.0000000000 0.0000000000 0.0000000000
Eigenvector (X, Y, Z): (0.000000, 0.000000, 1.000000)
Eigenvectors obtained.

INFO-interfaz: Translated Model2 with -0.0000000000 ➔➔➔ ***Valor esperado de traslación***

Ante cada traslación hecha con el shell max_ev.sh, se busca completar al casquete del modelo de cabeza con la elipsoide equivalente, y en este caso se toma como límite al eigenvalor y eigenvector negativo en el eje X. Mediante este método, se acelera a la supresión de outliers, ya que se destaca a los puntos del mentón y de la nariz. En esta iteración, la traslación total del modelo de elipsoide se muestra en la tabla 5.1.2. Para algunos modelos de cabeza fue necesario trasladar también sobre el eje Z y/o Y, ya que la alineación del modelo provocaba que su casquete incompleto se percibiera a mayor grado en otra orientación.

Centroide de la elipsoide equivalente trasladada	Valores de iteración actual [mm]	Valores de iteración previa [mm]
X	***-12.9551202384***	N/A
Y	***0.0***	N/A
Z	***0.0***	N/A

Tabla 5.1.2. Traslación de la elipsoide equivalente.

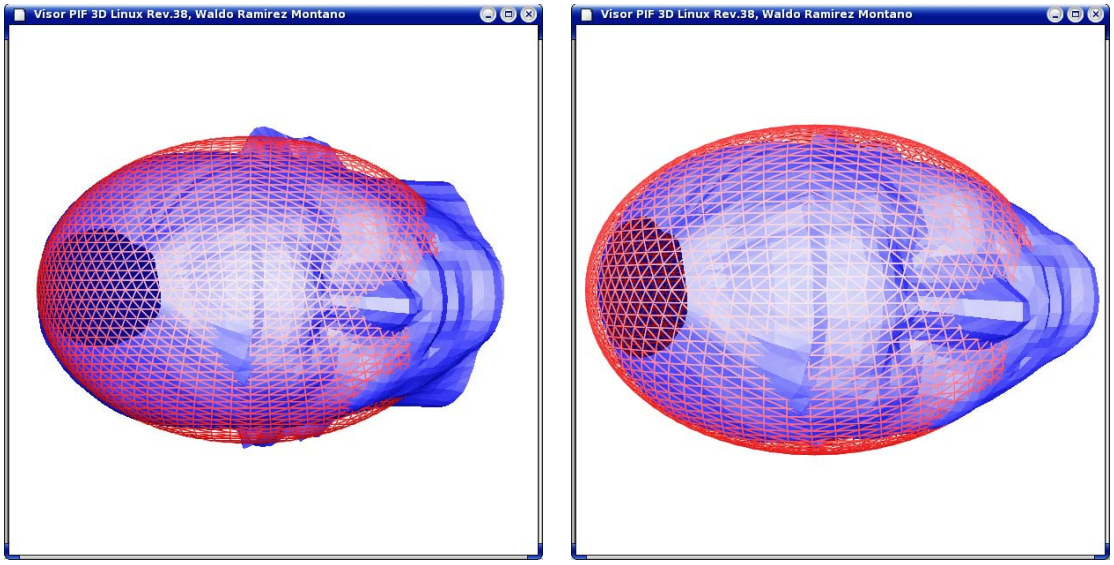


Figura 5.1.6. Plano XY del modelo de cabeza alineado (cabeza00_.pi) y su elipsoide equivalente nivel 25 (e00_01R.pi) después de su traslación. Se observa que los outliers destacan en la región de nariz y mentón del modelo de cabeza. Izquierda: Proyección ortogonal. Derecha: Proyección Frustum.

✓ Es necesario recordar la nueva posición del centroide de la elipsoide equivalente, así como almacenar su nueva versión para poder efectuar la supresión de outliers en caso de ser necesario:

☞ ***Se presionó la tecla '2' para elegir la vista del modelo de elipsoide***

INFO-teclado: Model2 view selected

☞ ***Se presionó la tecla 'i' para observar al centroide de la elipsoide***

INFO-interfaz: Model2 information:
# of points: 2502, # of triangles: 5000
Geometric centroid: (**-12.9551202384, 0.0000000000, 0.0000000000**) ➔➔➔ ***Nuevo centroide***
Inertia tensor:
6069.909424743067575 0.000000000000298 0.000000000000126
0.000000000000298 9097.478097542862088 0.000000000000019
0.000000000000126 0.000000000000019 7452.207748671035915
Eigenvalues: (1)6069.9094247431, (2)9097.4780975429, (3)7452.2077486710
Eigenvector (1): (X)1.000000, (Y)0.000000, (Z)0.000000
Eigenvector (2): (X)0.000000, (Y)1.000000, (Z)0.000000
Eigenvector (3): (X)0.000000, (Y)0.000000, (Z)1.000000

☞ ***Se presionó la tecla 'o' para almacenar la nueva versión de la elipsoide***

INFO-exporta: Creating model's XML file: "./00/e00_01R_.xml"... finished.
INFO-exporta: Creating the PIF file: "./00/e00_01R_.pi"... finished.

☞ ***Se presionó la tecla 'q' para salir del programa Visor3D***

INFO-pif: Unloading info3d "./00/e00_01R_"... done
INFO-pif: Unloading info3d "./00/cabeza00__"... done

☞ ***Es necesaria la medición del error, al ser el criterio de similitud entre el modelo de cabeza y su elipsoide equivalente. Desde consola, en la carpeta de programas se ejecuta al shell distanceField.sh:***

`./distanceField.sh 00 01`

Getting the three planes of the distance flield in model cabeza00.pi...
Model, stored in file "**./00/cabeza00_.pi**": ➔➔➔ ***Modelo de cabeza alineada***
Obtaining model's name... ./00/cabeza00__, done.
Getting points from 3D model... done.
Getting indexes from 3D model... done.
Model loaded: Number of points: 3301, number of triangles: 6442, creating the output file: "./00/YZ/cabeza00_YZ_01_.pnm"...
Waldo Ramirez Montano, Euclidian 3D Distance Fields, 2005.
Creating "triang_a_2D.txt" file...
53, 122, 105, 483, 445, 108, 66, 137, 105
244, 565, 216, 357, 0, 738, 243, 527, 233 ➔➔➔ ***"27+1" zonas con su cantidad de triángulos***
16, 148, 5, 347, 716, 262, 22, 171, 8
None: 0... done with the file...
Plane YZwith X=(0)X(3.333330) selected... ➔➔➔ ***Muestra YZ(X=0) del campo de distancia***

Getting zone-triangle ID... 1(4,148.4),2(2,171.0),3(3,131.5),4(4,134.3),5(6,155.2),6(6,141.2),7(4,146.9),8(9,167.5),9(9,127.5),10(10,149.6),11(11,169.4),12(3,161.0),13(13,127.5),14(13,134.0),15(15,131.4),16(16,148.4),17(17,171.8),18(18,156.4),19(22,158.9),20(20,169.9),21(24,163.8),22(22,149.3),23(23,157.8),24(24,141.7),25(22,158.5),26(26,165.9),27(18,162.2), done...
Max. distance: 123.696953. Min. distance: 0.000543
256 color eq.: g = (distance - 0.0005431867) *2.0614988036 ➔➔➔ ***Regresión lineal simple***
Creating distances file: "./00/YZ/cabeza00_YZ_01_.df"...
Start time: 1123223035 sec., 498414 ms, end time: 1123223187 sec., 161155 ms
Time taken (seconds): 152 ➔➔➔ ***Tiempo que tomó la obtención de la muestra***
➔➔➔ ***Inicio de la siguiente muestra***
Model, stored in file "./00/cabeza00_.pi":
Obtaining model's name... ./00/cabeza00__, done.
Getting points from 3D model... done.
Getting indexes from 3D model... done.
Model loaded: Number of points: 3301, number of triangles: 6442, creating the output file: "./00/XY/cabeza00_XY_01_.pnm"...
Waldo Ramirez Montano, Euclidian 3D Distance Fields, 2005.
Creating "triang_a_2D.txt" file...
53, 122, 105, 483, 445, 108, 66, 137, 105
244, 565, 216, 357, 0, 738, 243, 527, 233
16, 148, 5, 347, 716, 262, 22, 171, 8
None: 0... done with the file...
Plane XYwith Z=(0)X(3.333330) selected... ➔➔➔ ***Muestra XY(Z=0) del campo de distancia***

Getting zone-triangle ID... 1(4,148.4),2(2,171.0),3(3,131.5),4(4,134.3),5(6,155.2),6(6,141.2),7(4,146.9),8(9,167.5),9(9,127.5),10(10,149.6),11(11,169.4),12(3,161.0),13(13,127.5),14(13,134.0),15(15,131.4),16(16,148.4),17(17,171.8),18(18,156.4),19(22,158.9),20(20,169.9),21(24,163.8),22(22,149.3),23(23,157.8),24(24,141.7),25(22,158.5),26(26,165.9),27(18,162.2), done...
Max. distance: 117.651586. Min. distance: 0.023744

256 color eq.: g = (distance - 0.0237441804) *2.1678541130 ➔➔➔ ***Regresión lineal simple***
Creating distances file: "./00/XY/cabeza00_XY_01_.df"...
Start time: 1123223187 sec., 198194 ms, end time: 1123223320 sec., 910503 ms
Time taken (seconds): 133 ➔➔➔ ***Tiempo que tomó la obtención de la muestra***
➔➔➔ ***Inicio de la siguiente muestra***
Model, stored in file "./00/cabeza00_.pi":
Obtaining model's name... ./00/cabeza00__, done.
Getting points from 3D model... done.
Getting indexes from 3D model... done.
Model loaded: Number of points: 3301, number of triangles: 6442, creating the output file: "./00/XZ/cabeza00_XZ_01_.pnm"...
Waldo Ramirez Montano, Euclidian 3D Distance Fields, 2005.
Creating "triang_a_2D.txt" file...
53, 122, 105, 483, 445, 108, 66, 137, 105
244, 565, 216, 357, 0, 738, 243, 527, 233
16, 148, 5, 347, 716, 262, 22, 171, 8
None: 0... done with the file...
Plane XZwith Y=(0)X(3.333330) selected... ➔➔➔ ***Muestra XZ(Y=0) del campo de distancia***

Getting zone-triangle ID... 1(4,148.4),2(2,171.0),3(3,131.5),4(4,134.3),5(6,155.2),6(6,141.2),7(4,146.9),8(9,167.5),9(9,127.5),10(10,149.6),11(11,169.4),12(3,161.0),13(13,127.5),14(13,134.0),15(15,131.4),16(16,148.4),17(17,171.8),18(18,156.4),19(22,158.9),20(20,169.9),21(24,163.8),22(22,149.3),23(23,157.8),24(24,141.7),25(22,158.5),26(26,165.9),27(18,162.2), done...
Max. distance: 102.353640. Min. distance: 0.000892
256 color eq.: g = (distance - 0.0008917365) *2.4913839952 ➔➔➔ ***Regresión lineal simple***
Creating distances file: "./00/XZ/cabeza00_XZ_01_.df"...
Start time: 1123223320 sec., 944332 ms, end time: 1123223443 sec., 611225 ms
Time taken (seconds): 123 ➔➔➔ ***Tiempo que tomó la obtención de la muestra***

Done with model cabeza00_.pi ➔➔➔ ***Fin del muestreo en el modelo de cabeza***

Getting the three planes of the distance flield in model e00_01R_.pi...
Model, stored in file "**./00/e00_01R_.pi**": ➔➔➔ ***Modelo de elipsoide equivalente***
Obtaining model's name... ./00/e00_01R__, done.
Getting points from 3D model... done.
Getting indexes from 3D model... done.
Model loaded: Number of points: 2502, number of triangles: 5000, creating the output file: "./00/YZ/e00_YZ_01_.pnm"...
Waldo Ramirez Montano, Euclidian 3D Distance Fields, 2005.
Creating "triang_a_2D.txt" file...
29, 116, 5, 358, 476, 226, 29, 116, 5
196, 290, 114, 540, 0, 540, 196, 290, 114 ➔➔➔ ***"27+1" zonas con su cantidad de triángulos***
29, 116, 5, 358, 476, 226, 29, 116, 5
None: 0... done with the file...
Plane YZwith X=(0)X(3.333330) selected... ➔➔➔ ***Muestra YZ(X=0) del campo de distancia***

Getting zone-triangle ID... 1(1,150.4),2(2,164.9),3(6,163.4),4(4,139.1),5(5,155.4),6(6,156.4),7(7,150.4),8(8,164.9),9(6,163.4),10(10,149.8),11(11,174.0),12(12,165.6),13(13,126.8),14(13,133.0),15(15,152.7),16(16,149.8),17(17,174.0),18(18,165.6),19(22,150.4),20(20,164.9),21(24,163.4),22(22,139.1),23(23,155.4),24(24,156.4),25(25,150.4),26(26,164.9),27(24,163.4), done...
Max. distance: 117.388286. Min. distance: 0.004761
256 color eq.: g = (distance - 0.0047614172) *2.1723661812 ➔➔➔ ***Regresión lineal simple***
Creating distances file: "./00/YZ/e00_YZ_01_.df"...
Start time: 1123223443 sec., 636454 ms, end time: 1123223559 sec., 40789 ms
Time taken (seconds): 116 ➔➔➔ ***Tiempo que tomó la obtención de la muestra***

➔➔➔ *Inicio de la siguiente muestra*
Model, stored in file "./00/e00_01R_.pi":
Obtaining model's name... ./00/e00_01R__, done.
Getting points from 3D model... done.
Getting indexes from 3D model... done.
Model loaded: Number of points: 2502, number of triangles: 5000, creating the output file: "./00/XY/e00_XY_01_.pnm"...
Waldo Ramirez Montano, Euclidian 3D Distance Fields, 2005.
Creating "triang_a_2D.txt" file...
29, 116, 5, 358, 476, 226, 29, 116, 5
196, 290, 114, 540, 0, 540, 196, 290, 114
29, 116, 5, 358, 476, 226, 29, 116, 5
None: 0... done with the file...
Plane XYwith Z=(0)X(3.333330) selected... ➔➔➔ *Muestra XY(Z=0) del campo de distancia*

Getting zone-triangle ID... 1(1,150.4),2(2,164.9),3(6,163.4),4(4,139.1),5(5,155.4),6(6,156.4),7(7,150.4),8(8,164.9),9(6,163.4),10(10,149.8),11(11,174.0),12(12,165.6),13(13,126.8),14(13,133.0),15(15,152.7),16(16,149.8),17(17,174.0),18(18,165.6),19(22,150.4),20(20,164.9),21(24,163.4),22(22,139.1),23(23,155.4),24(24,156.4),25(25,150.4),26(26,164.9),27(24,163.4), done...
Max. distance: 116.034809. Min. distance: 0.030486
256 color eq.: g = (distance - 0.0304859179) * 2.1981939378 ➔➔➔ *Regresión lineal simple*
Creating distances file: "./00/XY/e00_XY_01_.df"...
Start time: 1123223559 sec., 66764 ms, end time: 1123223667 sec., 171501 ms
Time taken (seconds): 108 ➔➔➔ *Tiempo que tomó la obtención de la muestra*
➔➔➔ *Inicio de la siguiente muestra*
Model, stored in file "./00/e00_01R_.pi":
Obtaining model's name... ./00/e00_01R__, done.
Getting points from 3D model... done.
Getting indexes from 3D model... done.
Model loaded: Number of points: 2502, number of triangles: 5000, creating the output file: "./00/XZ/e00_XZ_01_.pnm"...
Waldo Ramirez Montano, Euclidian 3D Distance Fields, 2005.
Creating "triang_a_2D.txt" file...
29, 116, 5, 358, 476, 226, 29, 116, 5
196, 290, 114, 540, 0, 540, 196, 290, 114
29, 116, 5, 358, 476, 226, 29, 116, 5
None: 0... done with the file...
Plane XZwith Y=(0)X(3.333330) selected... ➔➔➔ *Muestra XZ(Y=0) del campo de distancia*

Getting zone-triangle ID... 1(1,150.4),2(2,164.9),3(6,163.4),4(4,139.1),5(5,155.4),6(6,156.4),7(7,150.4),8(8,164.9),9(6,163.4),10(10,149.8),11(11,174.0),12(12,165.6),13(13,126.8),14(13,133.0),15(15,152.7),16(16,149.8),17(17,174.0),18(18,165.6),19(22,150.4),20(20,164.9),21(24,163.4),22(22,139.1),23(23,155.4),24(24,156.4),25(25,150.4),26(26,164.9),27(24,163.4), done...
Max. distance: 106.346963. Min. distance: 0.008772
256 color eq.: g = (distance - 0.0087723250) * 2.3980095866 ➔➔➔ *Regresión lineal simple*
Creating distances file: "./00/XZ/e00_XZ_01_.df"...
Start time: 1123223667 sec., 201168 ms, end time: 1123223764 sec., 301296 ms
Time taken (seconds): 97 ➔➔➔ *Tiempo que tomó la obtención de la muestra*
Done. Waldo Ramirez Montano, Thesis 2005.

Para el criterio de similitud, se obtienen tres muestras (planos XY, YZ, XZ) del campo de distancias del modelo de cabeza y tres muestras del campo de distancias del modelo de elipsoide equivalente. La diferencia de cada par de muestras establece la ventaja primordial de este criterio: son restas de las magnitudes de las distancias, es decir, valores unidimensionales.

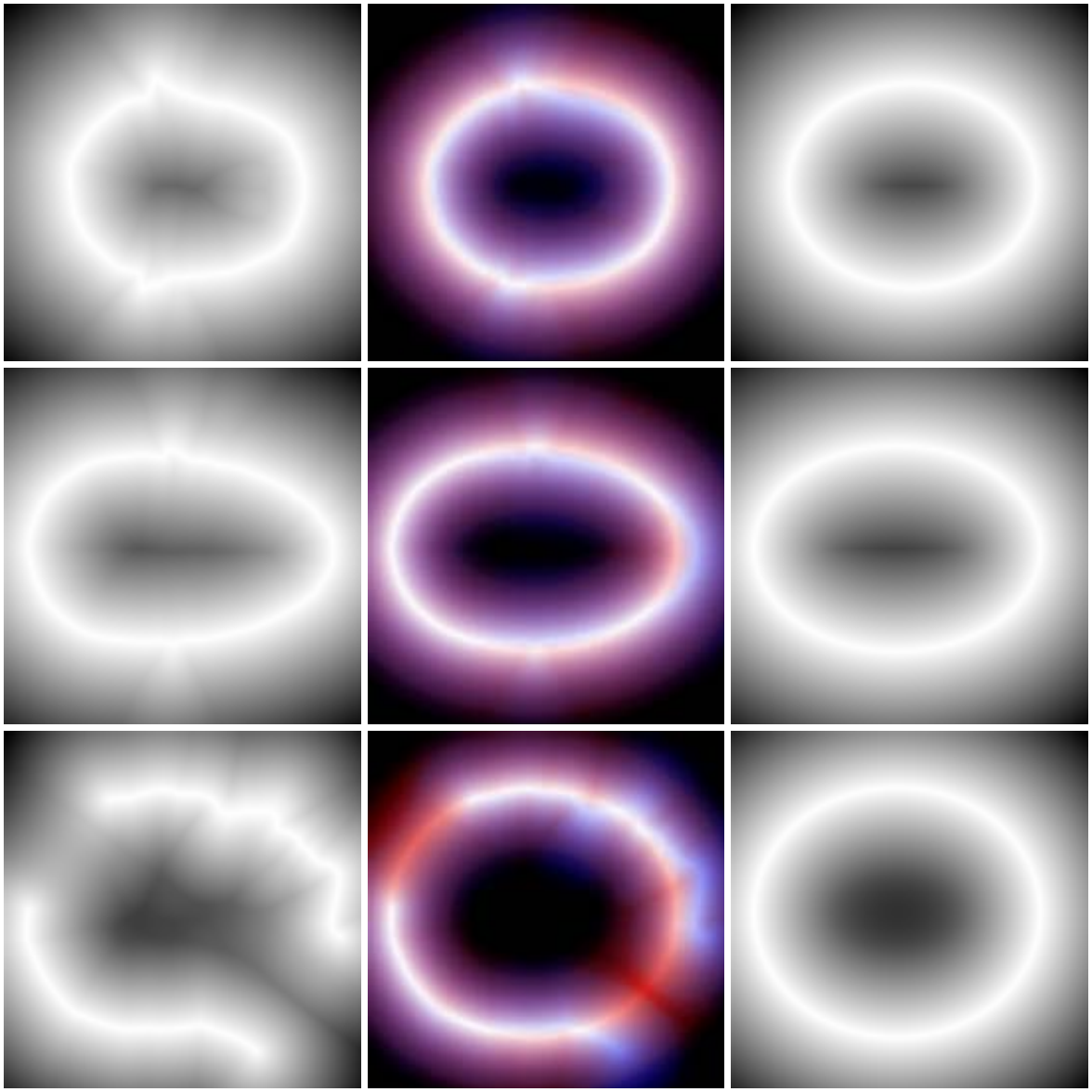


Figura 5.1.7. Imágenes representativas de las seis muestras del campo de distancia de los modelos cabeza00_.pi y e00_01R_.pi. Primera fila: Plano YZ (axial). Segunda fila: Plano XY (coronal). Tercera fila: Plano XZ (sagital).
La primera columna corresponde al modelo de cabeza, la tercera columna corresponde al modelo de elipsoide, mientras que la segunda columna corresponde a la comparación entre muestras de cabeza y elipsoide, hechas mediante GIMP. El tamaño de las seis imágenes es de 90X90 [pixels].

✓ ***En el plano sagital se observa la ubicación de la mayor cantidad de outliers.***

Mediante GIMP se crean tres imágenes de dos capas (segunda columna de la *fig.5.1.7*), en donde a la capa base (muestra del modelo de cabeza) se le aplica realce en tonos de color rojo, mientras que a la capa superior (muestra del modelo de la elipsoide) se le aplica realce en tonos de color azul. Finalmente, se implementa el modo de "multiplicación (quemar)" a la capa superior, con lo cual, los colores con tendencia al color blanco, representan a diferencias de distancias con tendencia al valor cero en la superficie de los modelos.

☞ ***El siguiente paso es obtener las diferencias entre las muestras, para aplicar el criterio de mínimos cuadrados y posiblemente modificar al modelo cabeza00_.pi. Utilizando la hoja de cálculo preparada en Gnumeric 1.2.6***

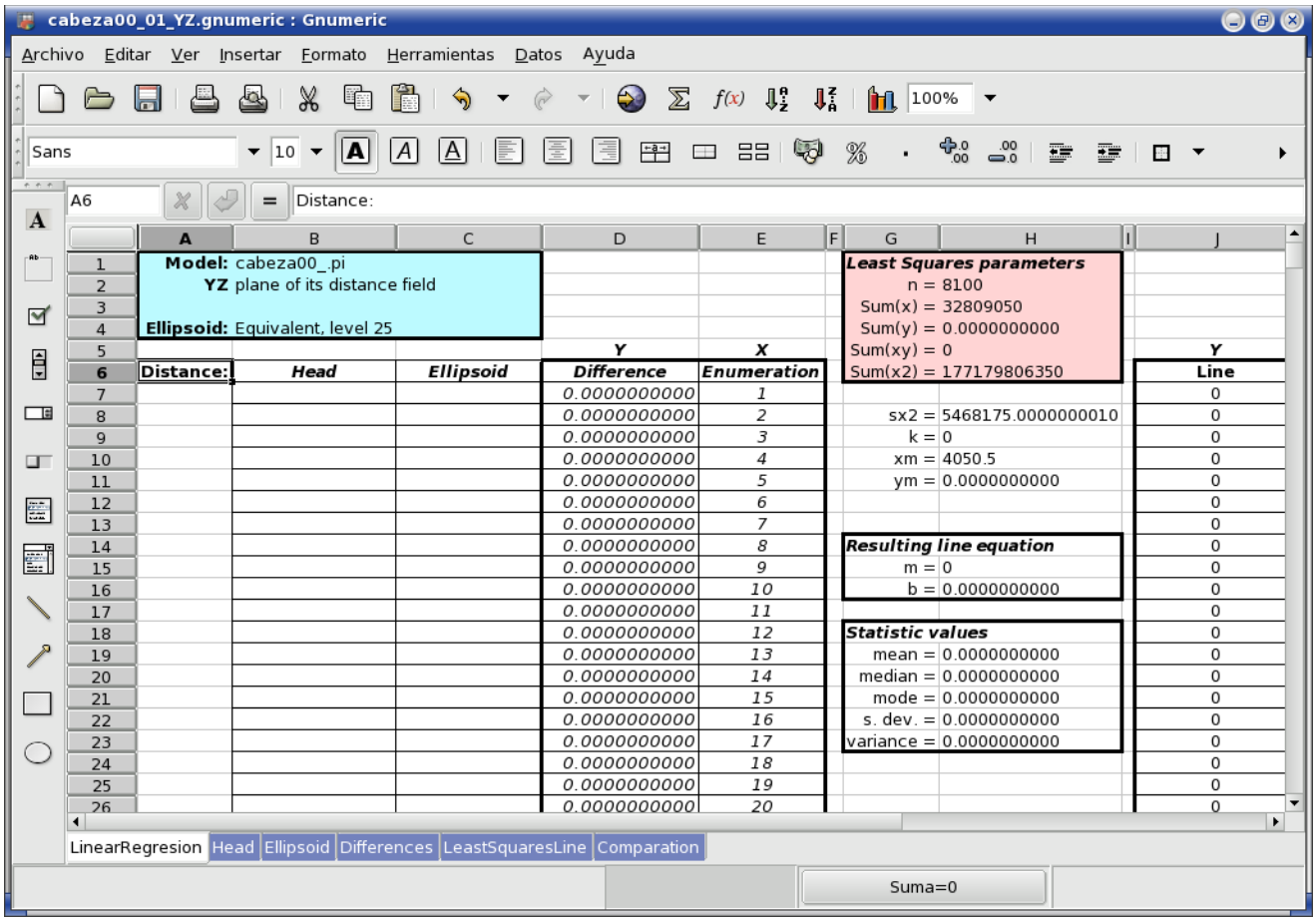


Figura 5.1.8. Hoja de cálculo empleada para la medición del error.

Es necesario introducir a los datos de las columnas “Head” (cabeza) y “Ellipsoid” (elipsoide) en las hojas de cálculo para al análisis de error entre los modelos cabeza00_.pi y e00_01R_.pi. Los datos se encuentran en los archivos de texto con las muestras de distancias capturadas por el programa dfields:

	Modelo de cabeza	***Modelo de elipsoide***
Plano YZ:	***cabeza00_YZ_01_.df***	***e00_YZ_01_.df***
Plano XY:	***cabeza00_XY_01_.df***	***e00_XY_01_.df***
Plano XZ:	***cabeza00_XZ_01_.df***	***e00_XZ_01_.df***

Mediante Gnumeric es posible importar a los datos almacenados en los archivos de texto, lo cual facilita el ingreso de las 8100 distancias de cada modelo a la hoja de cálculo. Se cuenta con tres hojas de cálculo (una para cada muestra de plano ortogonal), en las cuales, después de importar de los datos se obtienen los resultados del análisis de error de las diferencias de distancias: las ecuaciones de rectas de regresión lineal por mínimos cuadrados, las sumas cuadráticas, su interpretación estadística y gráficas de distancias (cabeza, elipsoide y se incluye a la gráfica de diferencias de distancias junto con su recta de regresión lineal por mínimos cuadrados).

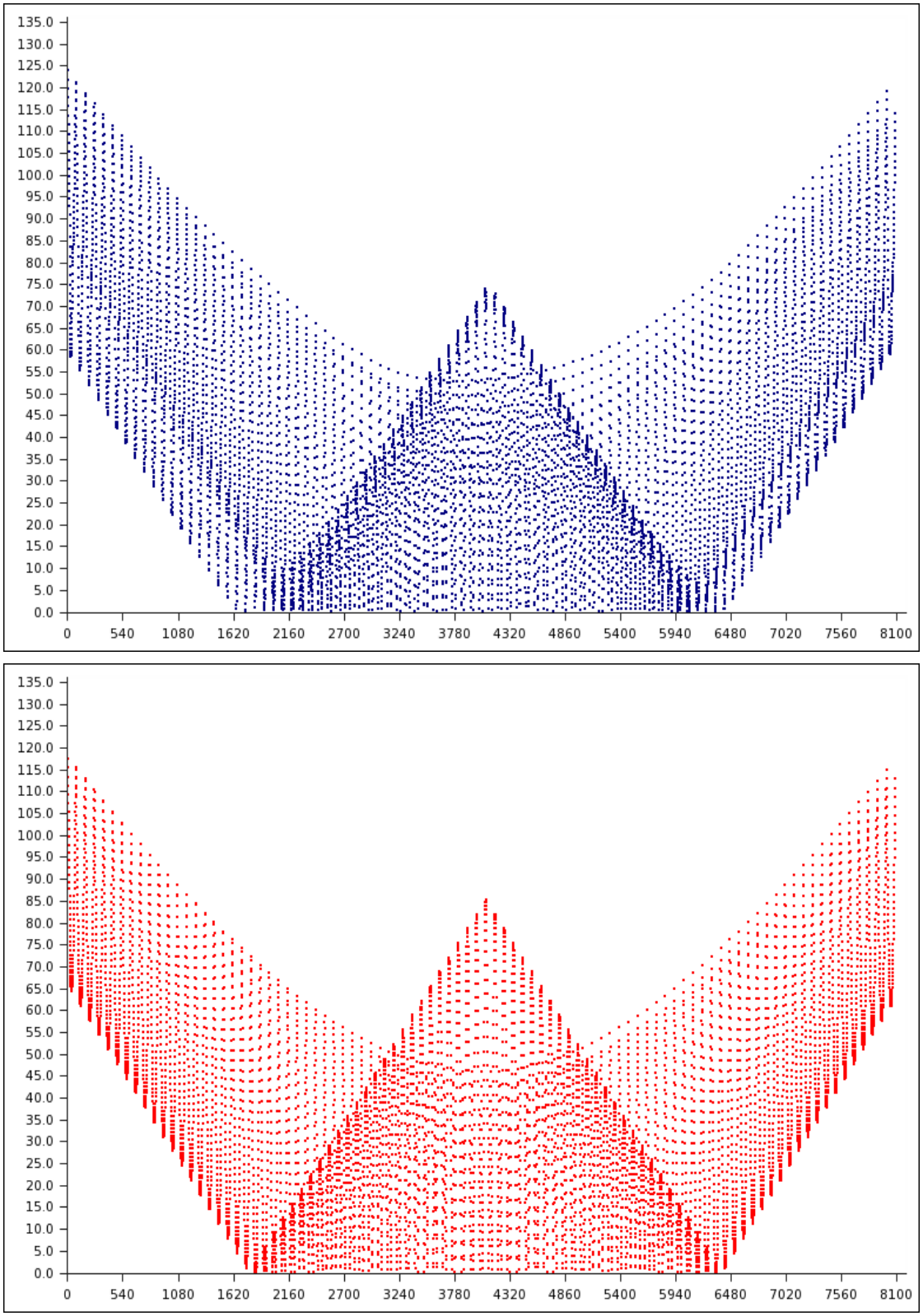


Figura 5.1.9A. Gráficas de las distancias en las muestras del plano YZ.
Superior: modelo cabeza00_.pi. Inferior: modelo e00_01R_.pi.
Eje X: Numeración de los 8100 puntos en el plano YZ.
Eje Y: Valores de distancia (en milímetros).

La *fig.5.1.9A* presenta a los 8100 valores de distancia en la muestra del plano YZ del modelo de cabeza y del modelo de elipsoide. Como se mencionó previamente, el conjunto de distancias es unidimensional, por lo cual se le puede representar de diversas formas (como las imágenes de la *fig.5.1.7*). En estas gráficas, a las cuales llamamos *gráficas de numeración de distancias,* el eje horizontal es simplemente su numeración (1, 2, 3, ..., 8100), mientras que el eje vertical es el valor de distancia (en milímetros). En este diseño, cada 90 valores de distancia en la gráfica, representa a un renglón del plano YZ (ya que el muestreo es de 90X90 distancias). Por ejemplo, en la siguiente figura se presenta únicamente a los primeros 90 valores de distancia (el primer renglón del plano YZ) de la gráfica de numeración de distancias del modelo de cabeza bajo análisis:

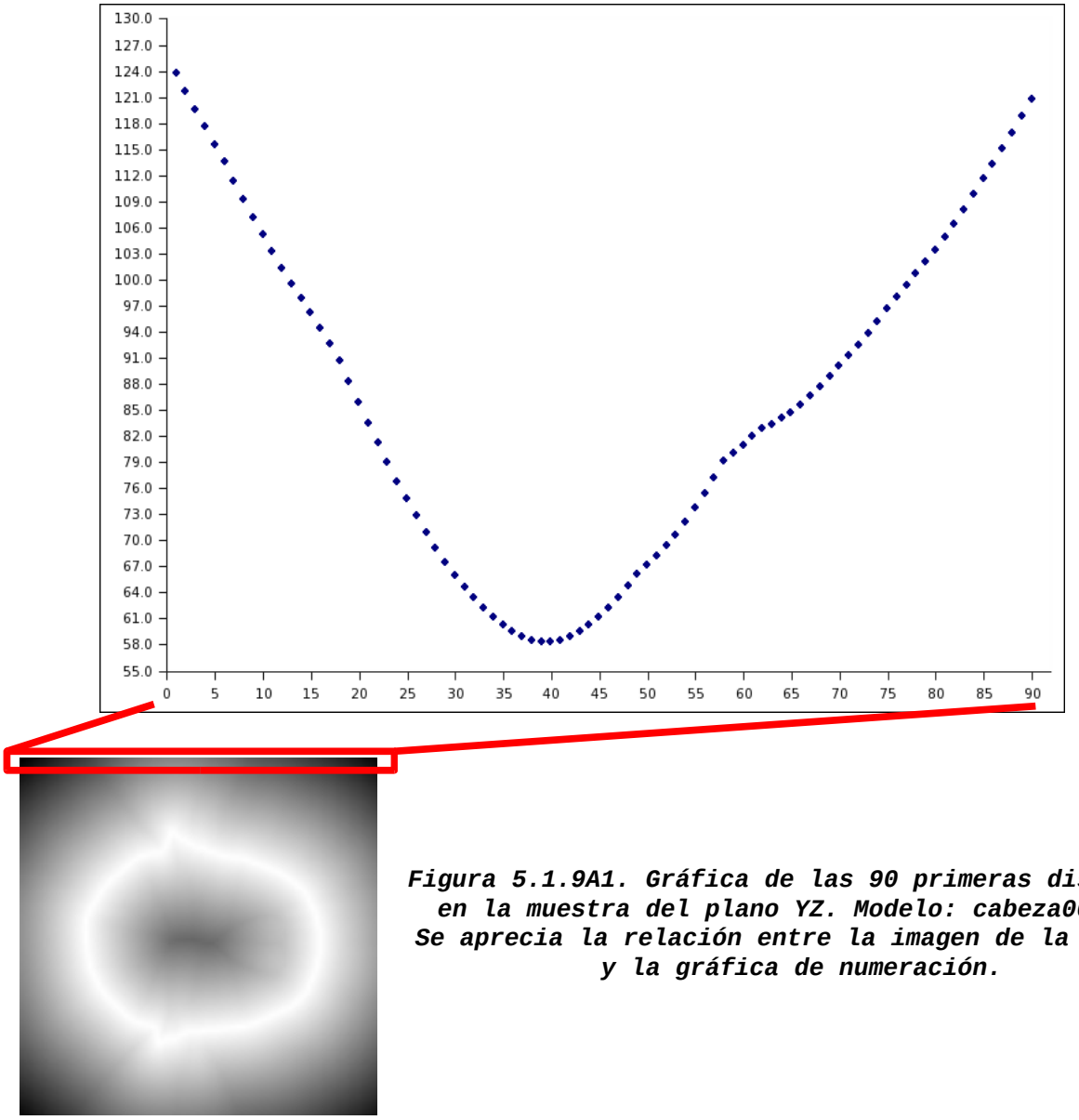


Figura 5.1.9A1. Gráfica de las 90 primeras distancias en la muestra del plano YZ. Modelo: cabeza00_.pi. Se aprecia la relación entre la imagen de la muestra y la gráfica de numeración.

De la misma forma, la siguiente figura representa al primer renglón de la gráfica de numeración de distancias del modelo de elipsoide bajo análisis:

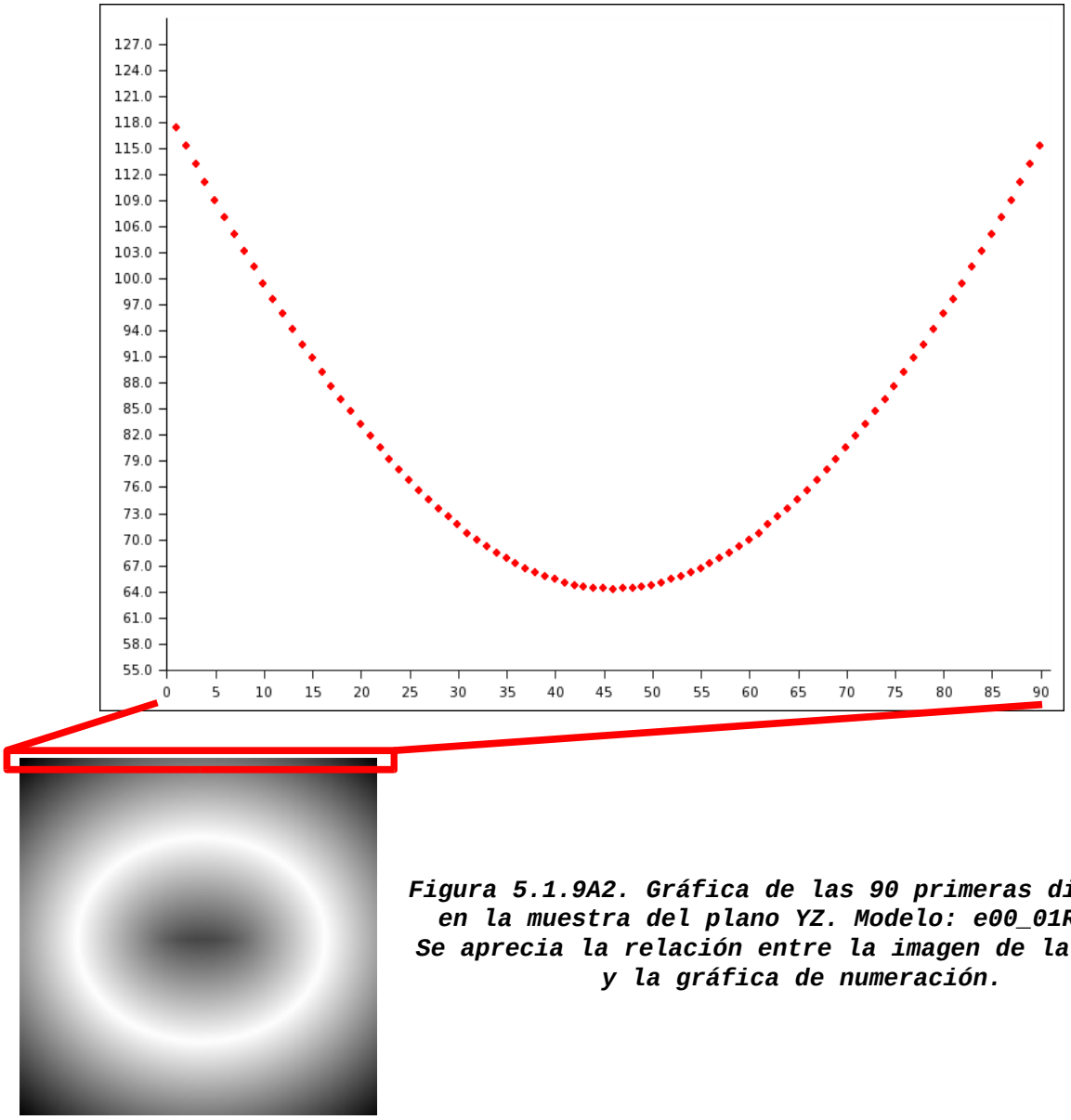


Figura 5.1.9A2. Gráfica de las 90 primeras distancias en la muestra del plano YZ. Modelo: e00_01R_.pi. Se aprecia la relación entre la imagen de la muestra y la gráfica de numeración.

A diferencia del modelo de cabeza, el primer renglón del modelo de elipsoide presenta una gráfica similar a una parábola: se deduce que la gráfica del primer renglón de distancias del modelo de cabeza (*fig.5.1.9A1*) **no** presenta una gráfica similar a una parábola debido a la presencia de una de sus orejas. Se pretende que en las iteraciones siguientes, las gráficas de numeración de distancias de ambos modelos tiendan a ser iguales.

En caso de comparar a las gráficas de numeración de distancias de un mismo modelo (el análisis de error de un modelo con sigo mismo, lo cual causaría gráficas 100% idénticas), su diferencia (la resta de distancias) sería cero en los 8100 valores, lo cual causaría una recta de regresión lineal con pendiente y ordenada al origen iguales a cero. Para este caso, a continuación se presenta a la comparación de las gráficas de numeración de distancias correspondientes al plano YZ (la resta de las 8100 distancias), de los modelos de cabeza y elipsoide (cabeza00_.pi y e00_01R_.pi):

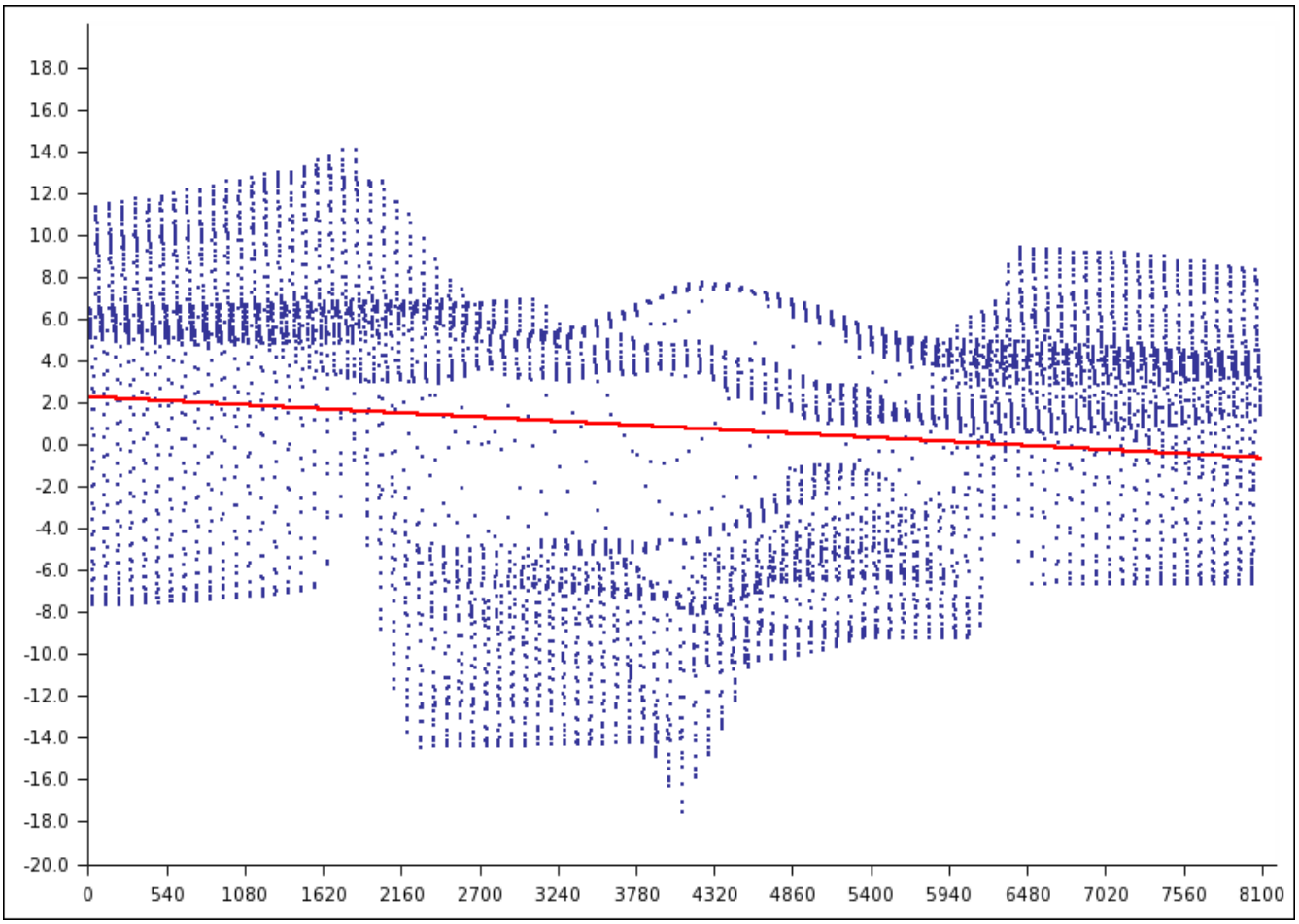


Figura 5.1.9B. Gráficas de las diferencias de distancias en el plano YZ.
Diferencias de distancia: cabeza00_ – e00_01R_ y recta de mínimos cuadrados.
Eje X: Numeración de los 8100 puntos en el plano YZ.
Eje Y: Valores de diferencias de distancias (en milímetros).

➔➔➔ *La ecuación de la recta de mínimos cuadrados YZ es la siguiente:*

$$y_{YZ} = -0.000358957891542719\,x + 2.2311840466$$

La ecuación representa al comportamiento del error en el plano YZ: la pendiente distinta de cero indica que existen diferencias de distancias con valores distintos, mientras que la ordenada al origen es una interpretación del promedio de las diferencias de distancias entre los modelos de cabeza y elipsoide.
Por otra parte, una interpretación adicional de la muestra del campo de distancia (la cual nos es útil para definir la convergencia del proceso iterativo), es la suma de las diferencias cuadráticas de distancias:

$$E_{YZ}^2 = 345621.1023283845\ [\text{mm}^2], \qquad E_{YZ} = 587.895485888763\ [\text{mm}].$$

Otra característica que acompañaría a una recta con pendiente y ordenada al origen iguales a cero, sería una suma de diferencias cuadráticas también con valor cero. A continuación se presenta al primer renglón de las diferencias de distancias, con el fin de complementar lo indicado con la *fig.5.1.9.A1* y la *fig.5.1.9.A2*.

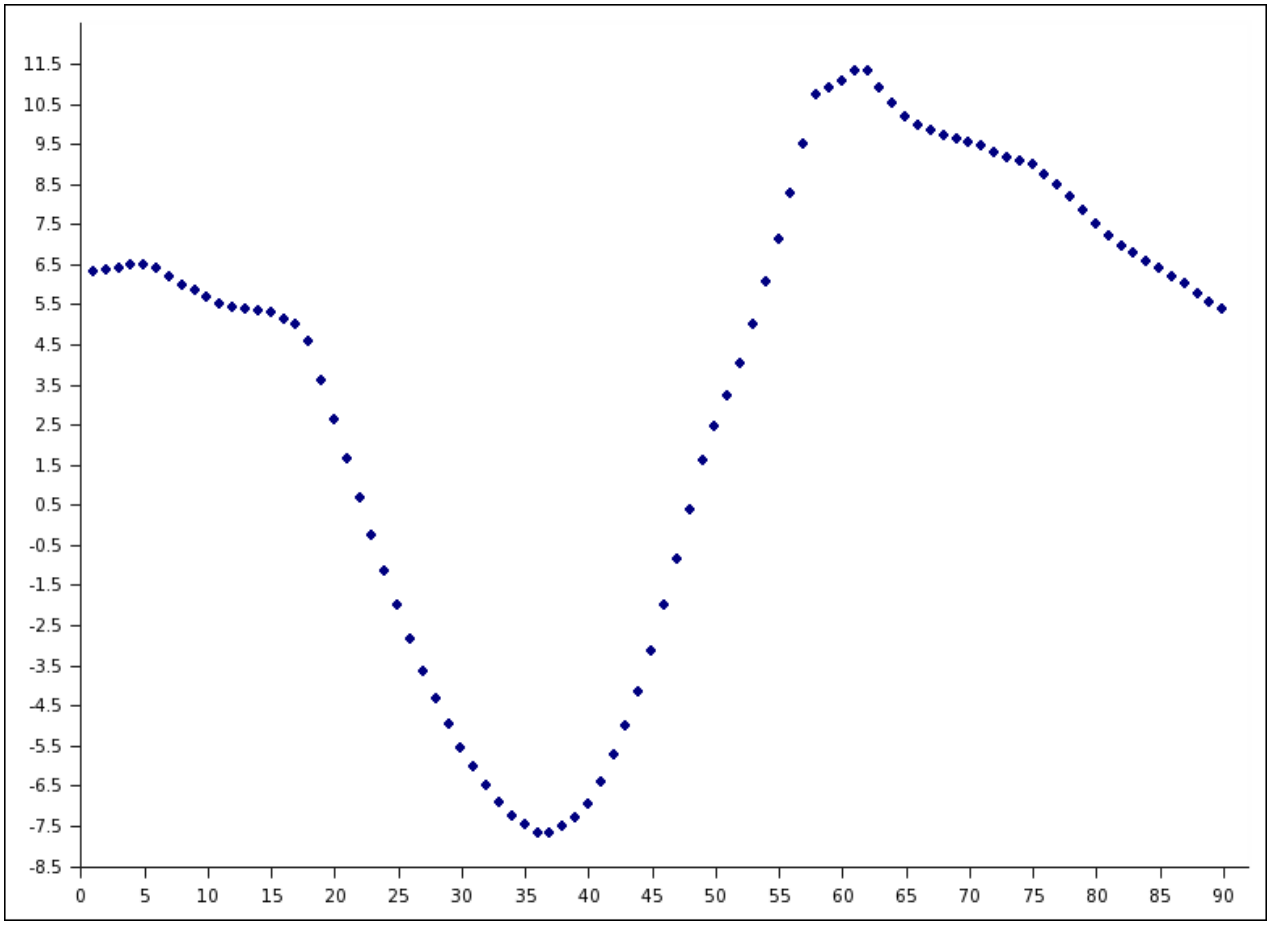


Figura 5.1.9B1. Gráfica de las 90 primeras diferencias de distancias entre el modelo cabeza00_.pi y el modelo e00_01R_.pi, en el plano YZ.

Se observa que las diferencias de distancia tienden al mismo rango de valores, (en este primer renglón las diferencias no superan magnitudes de 12 milímetros). En el caso de la comparación de dos modelos que no tienen relación alguna (desde el punto de vista morfológico), en las gráficas de diferencias de distancias se observaría un rango de valores amplio e inusual.

En las siguientes iteraciones se pretende minimizar al rango de valores en las diferencias de distancias, buscando su tendencia a cero. Para este punto interviene la suma de diferencias cuadráticas: si en la siguiente iteración la suma E_{YZ}^2 es menor, se obtiene una mejora de similitud entre los modelos; en caso contrario, el proceso tiende a finalizar, ya que una suma E_{YZ}^2 mayor indica menor similitud entre los modelos.

Por supuesto, para determinar el fin del proceso (su convergencia), es necesario obtener a las otras dos muestras del campo de distancia: los planos ortogonales XY y XZ, con el fin de contar con las sumas de diferencias cuadráticas E_{XY}^2, E_{XZ}^2. A continuación se resume el análisis de error en los planos ortogonales XY y XZ.

✓ ***El análisis de error del plano YZ se almacenó en el archivo: cabeza00_01_YZ.gnumeric***

☞ ***Utilizando la hoja de cálculo preparada en Gnumeric 1.2.6, se analizó al error en las muestras del plano XY.***

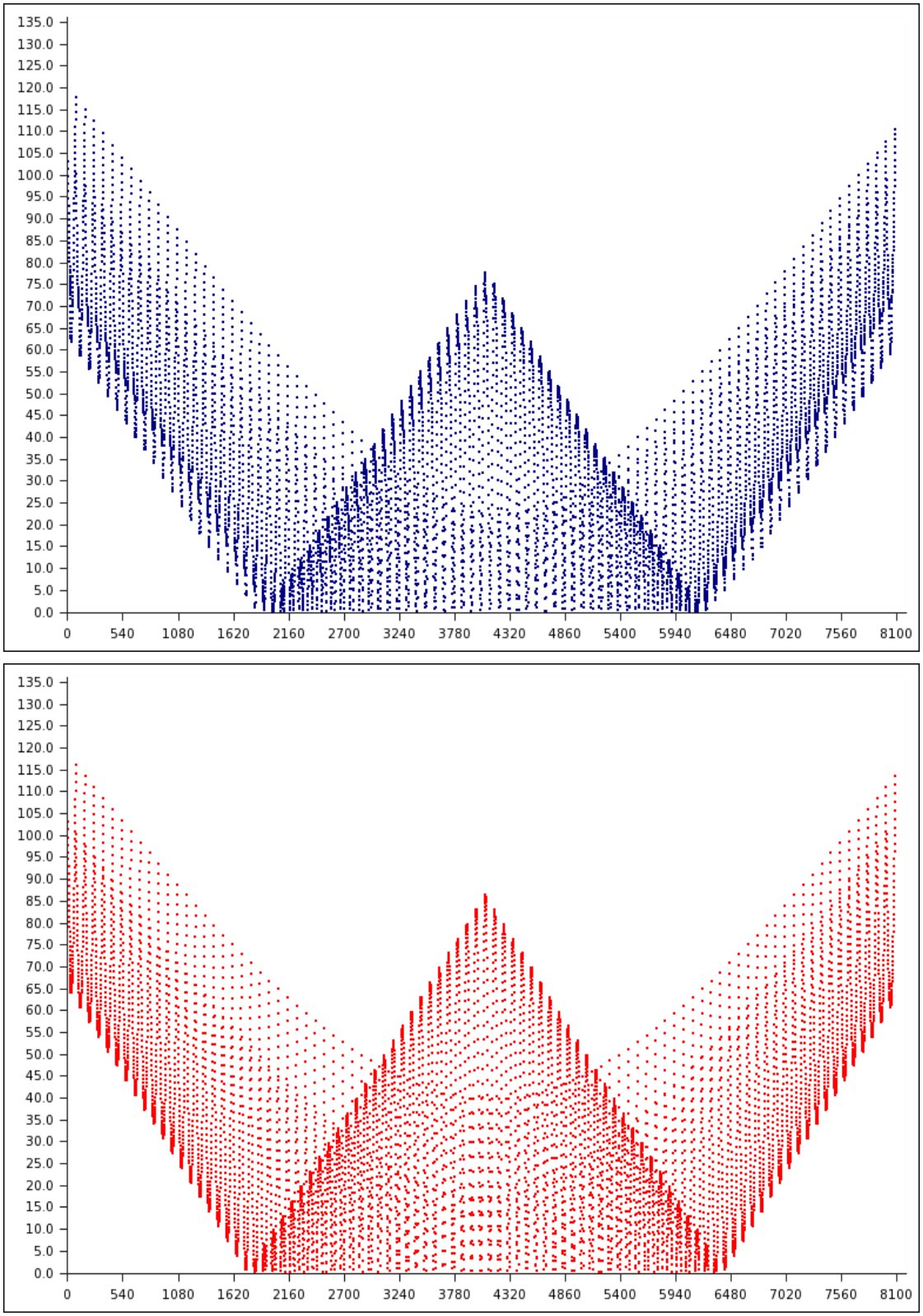


Figura 5.1.10A. Gráficas de las distancias en las muestras del plano XY.
Superior: modelo cabeza00_.pi. Inferior: modelo e00_01R_.pi.
Eje X: Numeración de los 8100 puntos en el plano XY.
Eje Y: Valores de distancia (en milímetros).

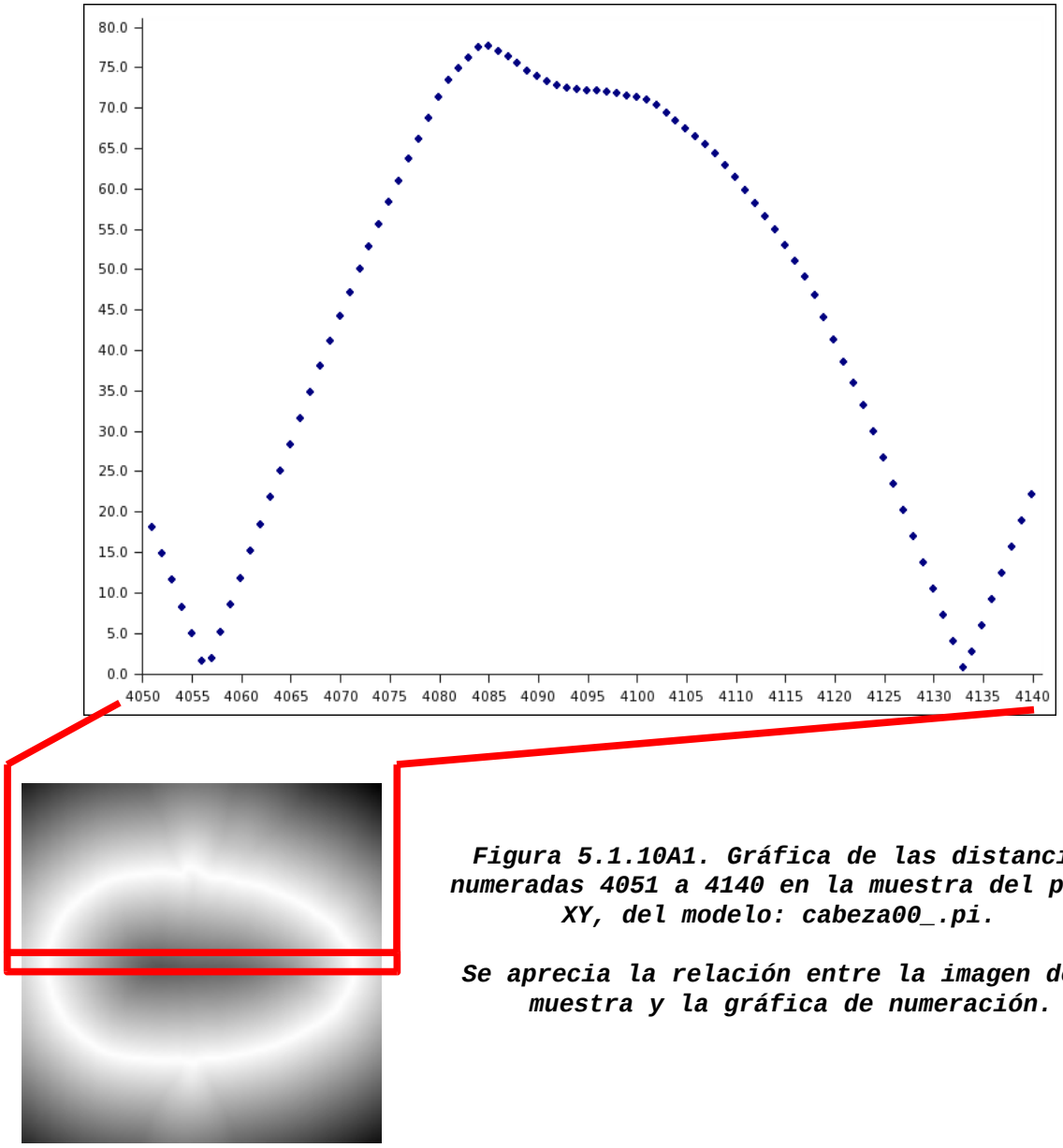


Figura 5.1.10A1. Gráfica de las distancias numeradas 4051 a 4140 en la muestra del plano XY, del modelo: cabeza00_.pi.

Se aprecia la relación entre la imagen de la muestra y la gráfica de numeración.

Para el plano XY, en la *fig.5.1.10.A1* se observa al renglón 45 de las distancias numeradas del modelo de cabeza. De nueva cuenta se presenta una figura no simétrica, ya que la presencia de outliers como la nariz y las orejas, causan que las distancias varíen en un modo no uniforme. Cabe señalar que en este renglón se presentan distancias muy cercanas a valor cero, gracias a que su serie de puntos atraviesa a la superficie del modelo, lo cual puede ser empleado como patrón para determinar si el punto de captura de distancia se encuentra en el interior o exterior del modelo.

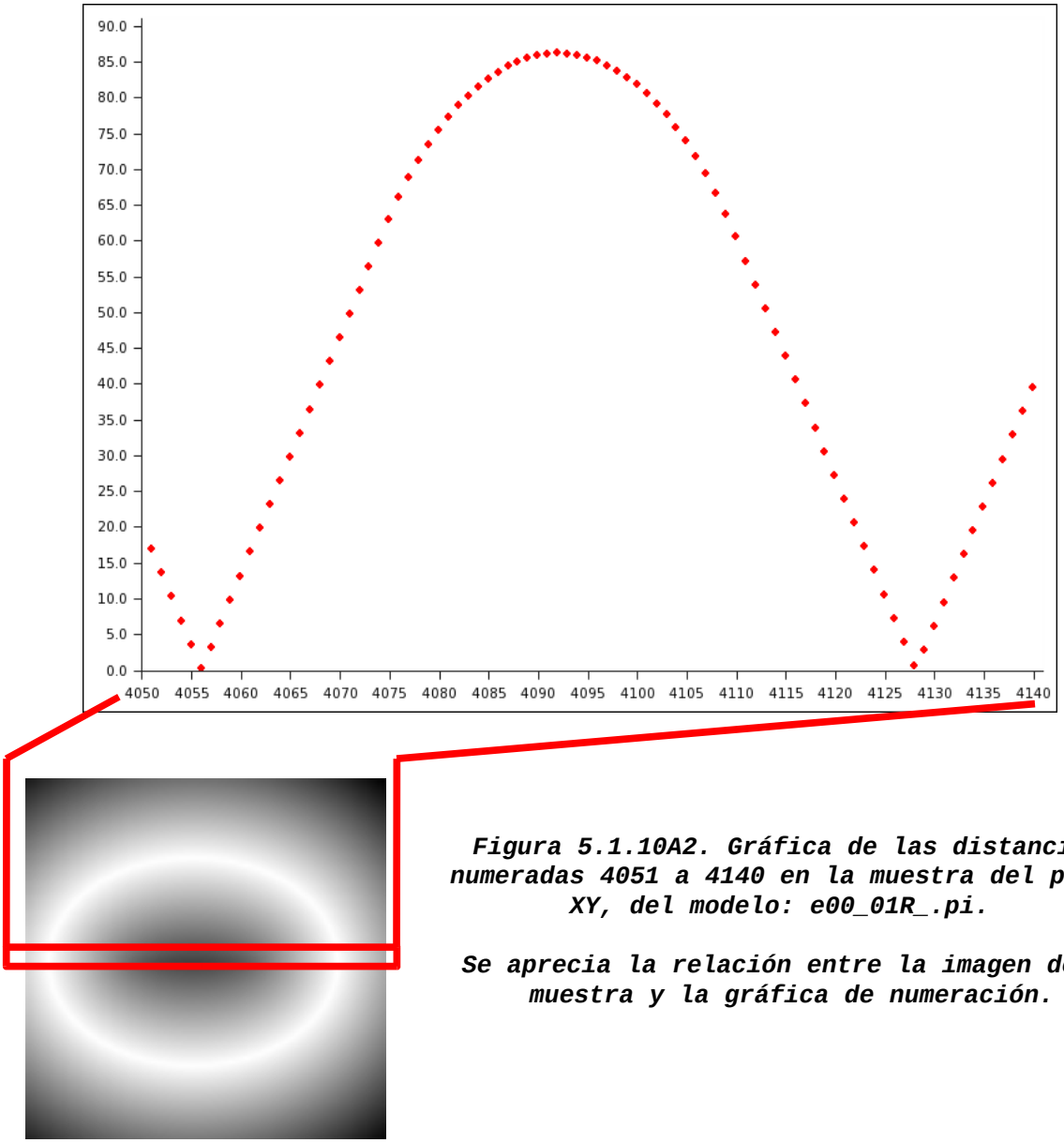


Figura 5.1.10A2. Gráfica de las distancias numeradas 4051 a 4140 en la muestra del plano XY, del modelo: e00_01R_.pi.

Se aprecia la relación entre la imagen de la muestra y la gráfica de numeración.

A diferencia de la *fig.5.1.10A1,* la *fig5.1.10A2* ejemplifica a un renglón de distancias propio de un modelo simétrico (nuevamente se obtiene una parábola, aunque en esta ocasión es del interior del modelo). Además, así como en la imagen del campo de distancia se despliega la traslación (sobre el eje X) a la cual fue sometido el modelo de elipsoide, la gráfica del renglón 45 de distancia también lo refleja con la numeración de las distancias que tienden a tocar a la superficie de la elipsoide y limitan a la parábola.

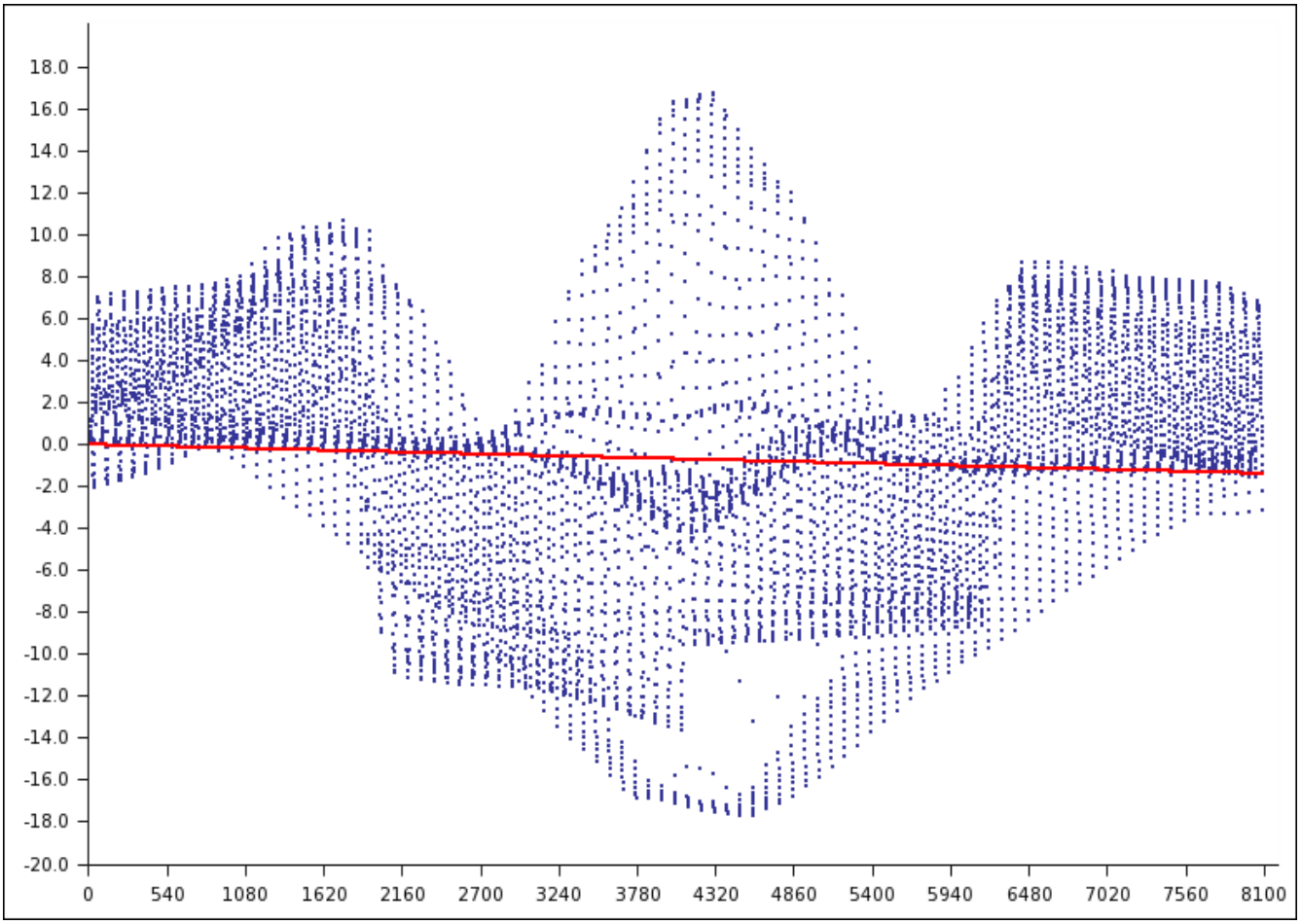


Figura 5.1.10B. Gráficas de las diferencias de distancias en el plano XY.
Diferencias de distancia: cabeza00_ – e00_01R_ y recta de mínimos cuadrados.
Eje X: Numeración de los 8100 puntos en el plano XY.
Eje Y: Valores de diferencias de distancias (en milímetros).

➔➔➔ *La ecuación de la recta de mínimos cuadrados XY es la siguiente:*

$$y_{XY} = -0.000170240286857869\,x - 0.0624768208$$

La ecuación representa al comportamiento del error en el plano XY:la pendiente distinta de cero indica que existen diferencias de distancias con valores distintos, mientras que la ordenada al origen es una interpretación del promedio de las diferencias de distancias entre los modelos de cabeza y elipsoide. El valor del error en el plano XYes la suma de diferencias cuadráticas de distancias:

$$E_{XY}^2 = 267291.2277374494\ [\text{mm}^2], \qquad E_{XY} = 517.002154480471\ [\text{mm}].$$

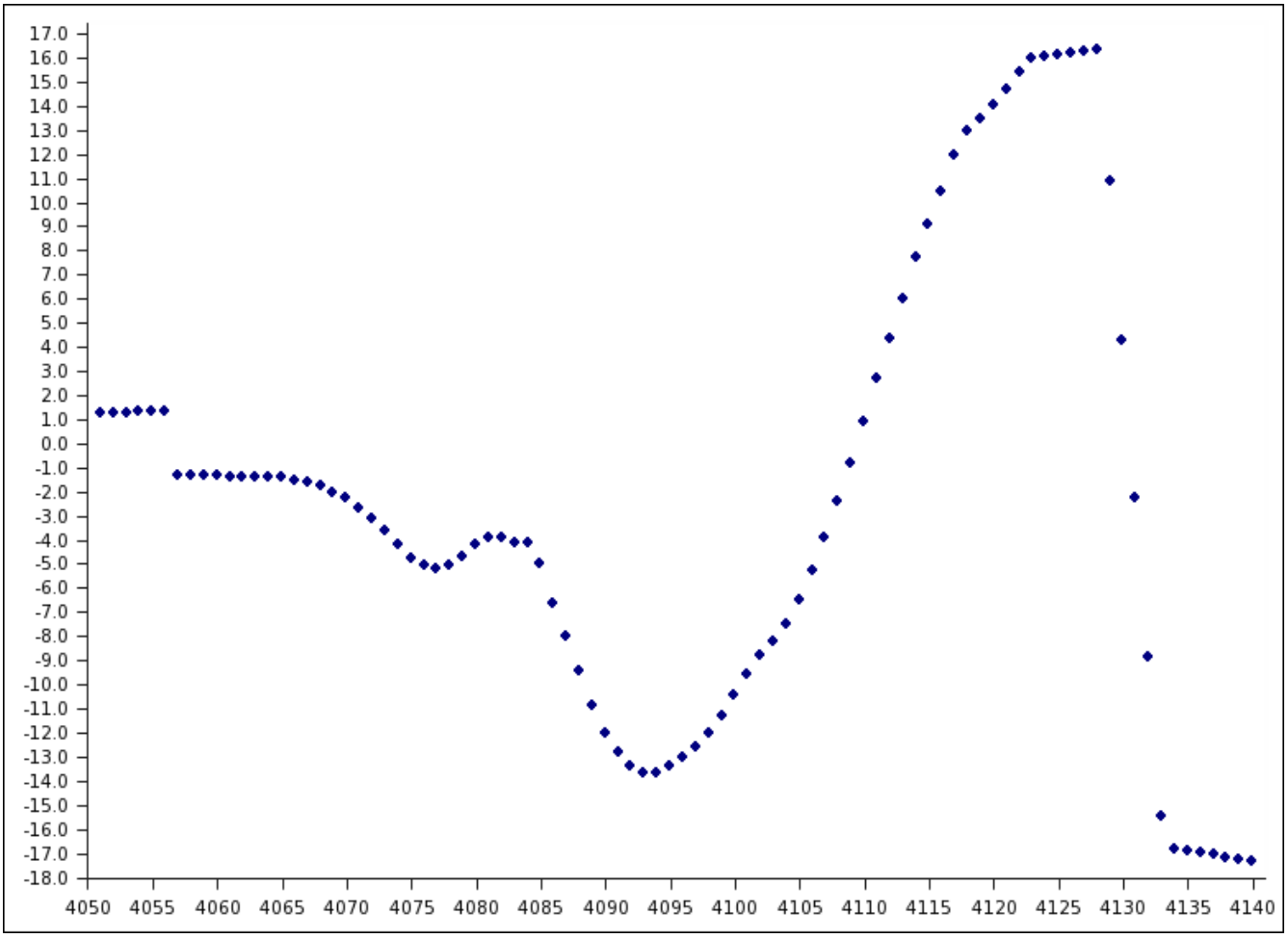


Figura 5.1.10B1. Gráfica de las diferencias de distancias numeradas 4051 a 4140, entre el modelo cabeza00_.pi y el modelo e00_01R_.pi, en el plano XY.

La sección final del renglón 45 es otro claro ejemplo de las diferencias de distancias obtenidas con outliers originales, en el modelo de cabeza: en la sección superior del modelo de cabeza (es decir, la sección cercana al casquete coronal) las diferencias de distancias son de relativamente poca magnitud, aunque al iniciar la comparación con secciones en donde existen puntos alejados del modelo de cabeza (en especial: nariz, orejas, mentón), las diferencias de distancias se disparan a magnitudes mucho mayores. Los cambios radicales de signo en las diferencias de distancias, indican la presencia de la superficie de los modelos.

✓ ***El análisis de error del plano XY se almacenó en el archivo: cabeza00_01_XY.gnumeric***

☞ ***Utilizando la hoja de cálculo preparada en Gnumeric 1.2.6, se analizó al error en las muestras del plano XZ.***

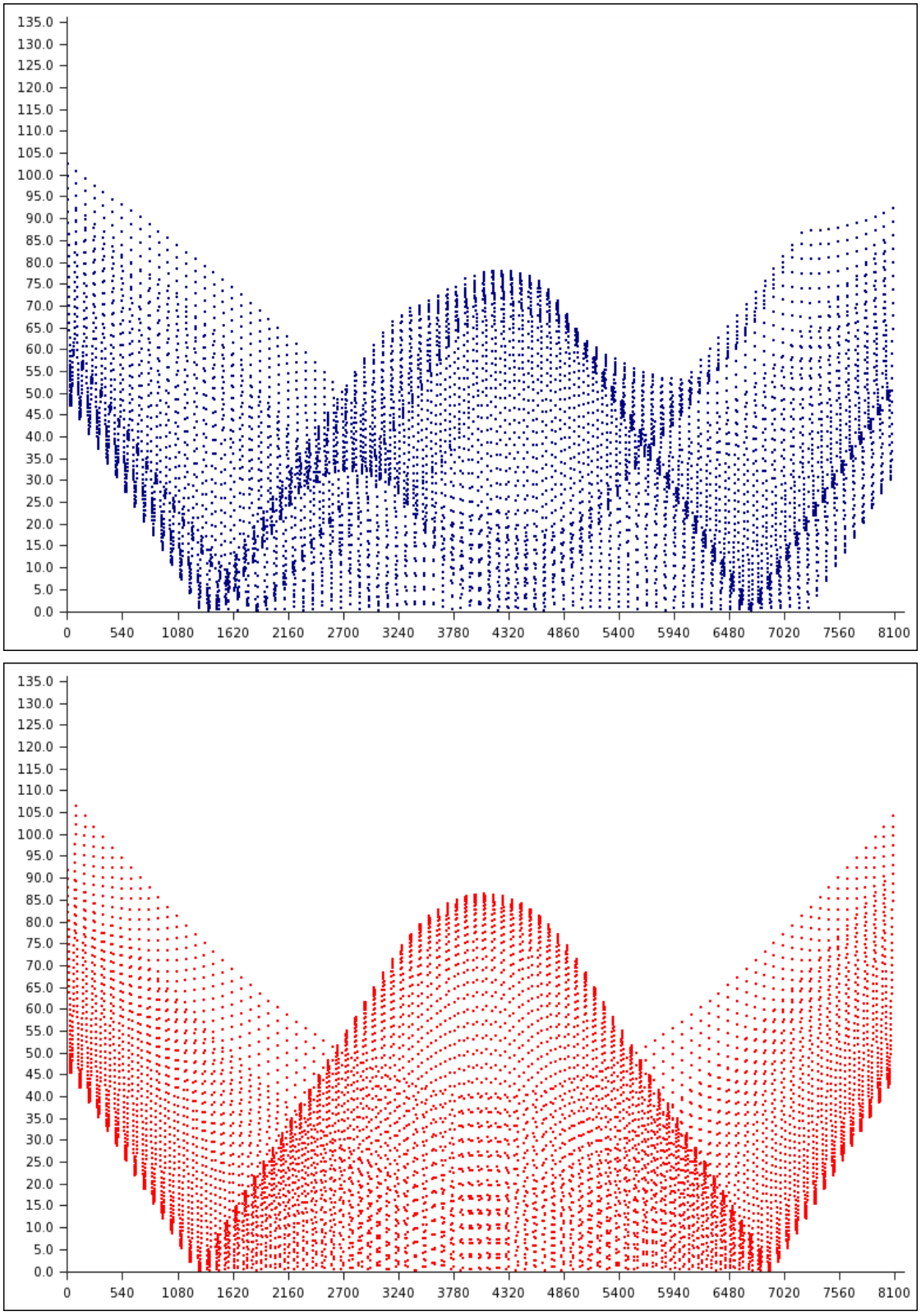


Figura 5.1.11A. Gráficas de las distancias en las muestras del plano XZ.
Superior: modelo cabeza00_.pi. Inferior: modelo e00_01R_.pi.
Eje X: Numeración de los 8100 puntos en el plano XY.
Eje Y: Valores de distancia (en milímetros).

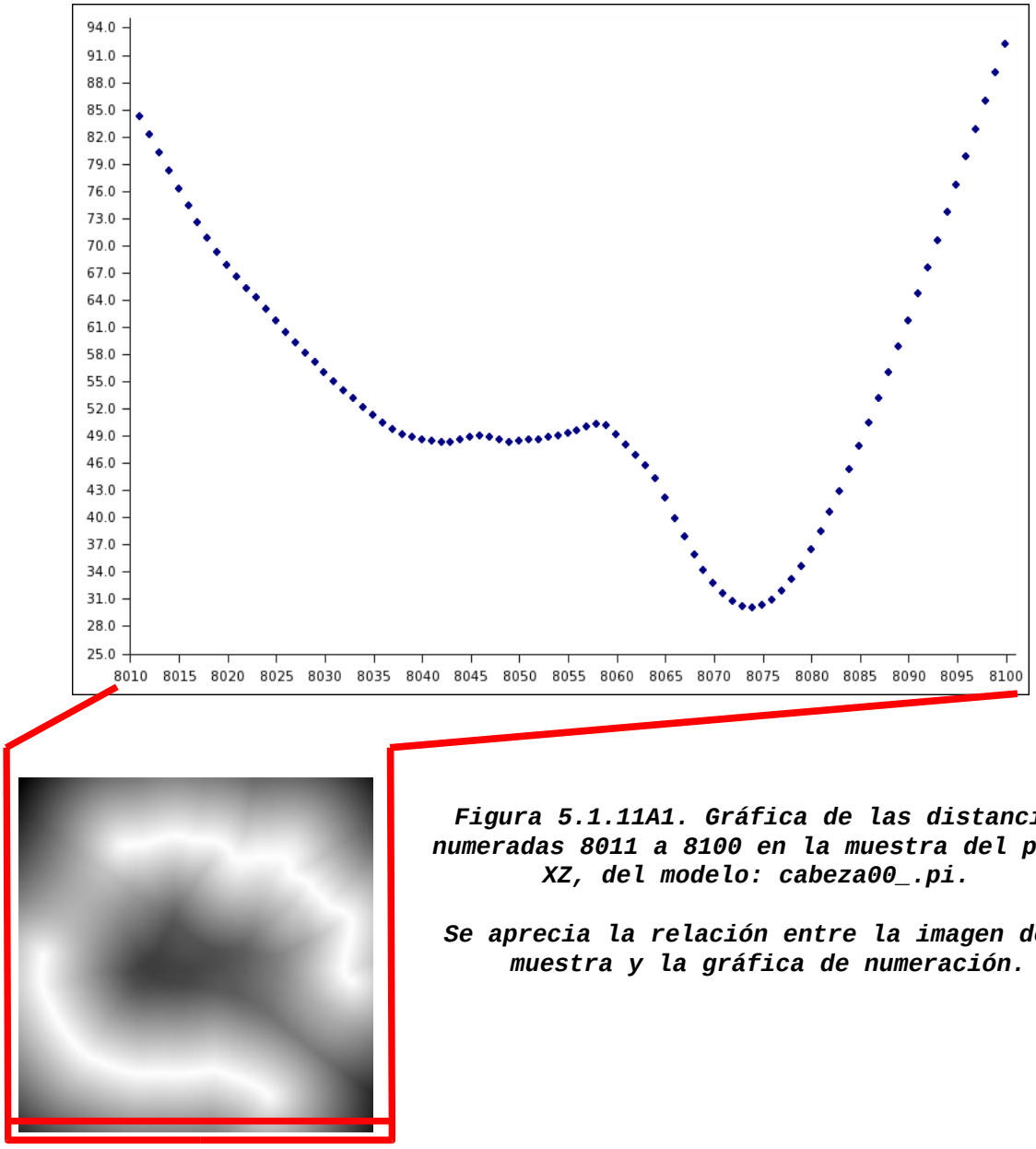


Figura 5.1.11A1. Gráfica de las distancias numeradas 8011 a 8100 en la muestra del plano XZ, del modelo: cabeza00_.pi.

Se aprecia la relación entre la imagen de la muestra y la gráfica de numeración.

El plano XZ fue el que mostró mayores valores de error en la comparación de distancias entre modelos de cabeza y modelos de elipsoide: la razón fundamental se debe a que los modelos de cabeza no son cerrados, generando valores de distancia de mayor desviación estándar. La *fig.5.1.11A1* indica que en el renglón 90, la presencia del cuello del modelo de cabeza reduce repentinamente la magnitud de la distancia, aunque al acercarse al corte del cuello, de forma similar se dispara el incremento de la magnitud de la distancia.

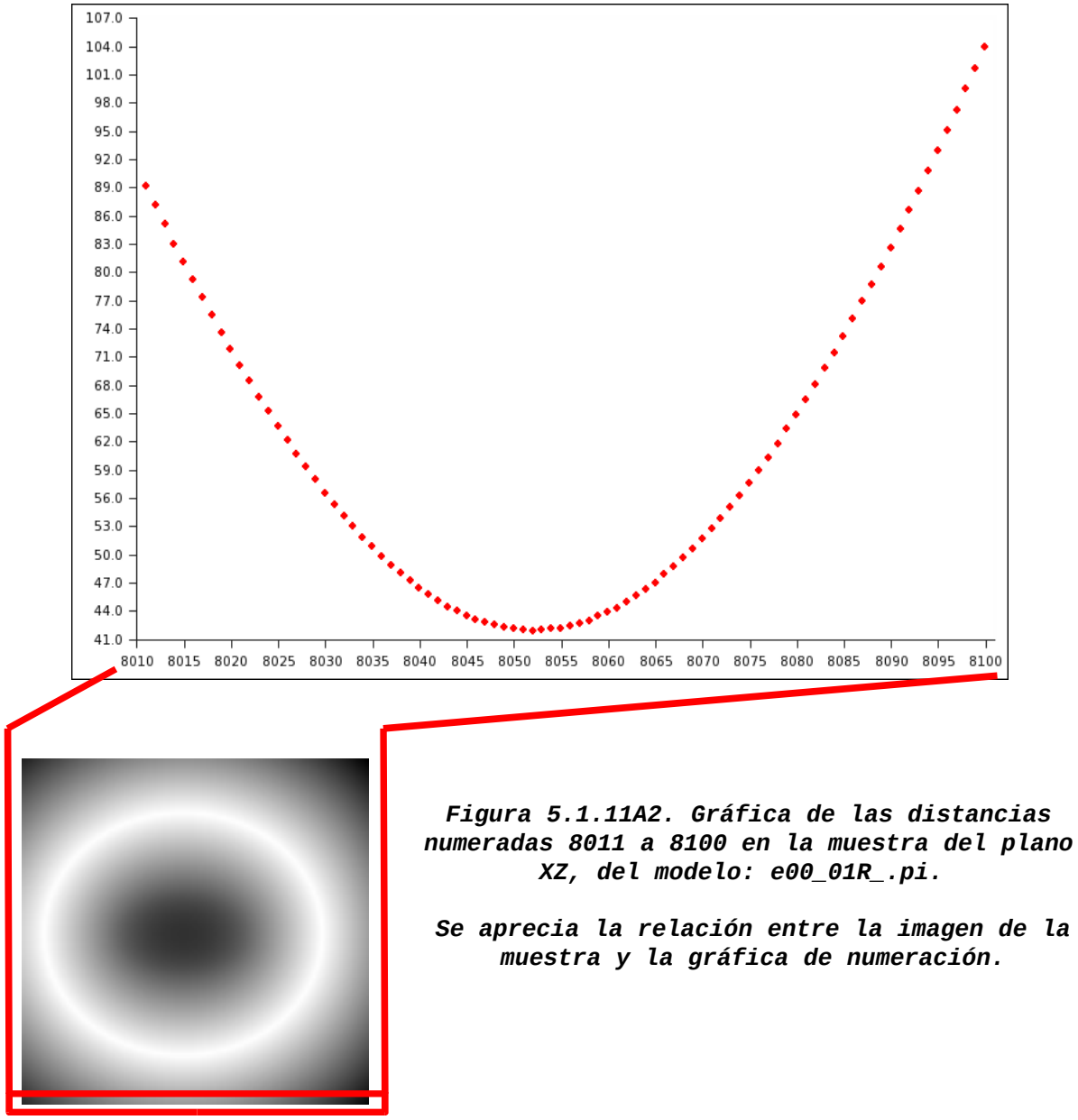


Figura 5.1.11A2. Gráfica de las distancias numeradas 8011 a 8100 en la muestra del plano XZ, del modelo: e00_01R_.pi.

Se aprecia la relación entre la imagen de la muestra y la gráfica de numeración.

El renglón 90 del modelo de elipsoide responde con distancias de magnitud simétrica, ya que nuevamente crea una parábola en la gráfica de distancias numeradas. De igual forma, también se aprecia a la traslación sobre el eje X.

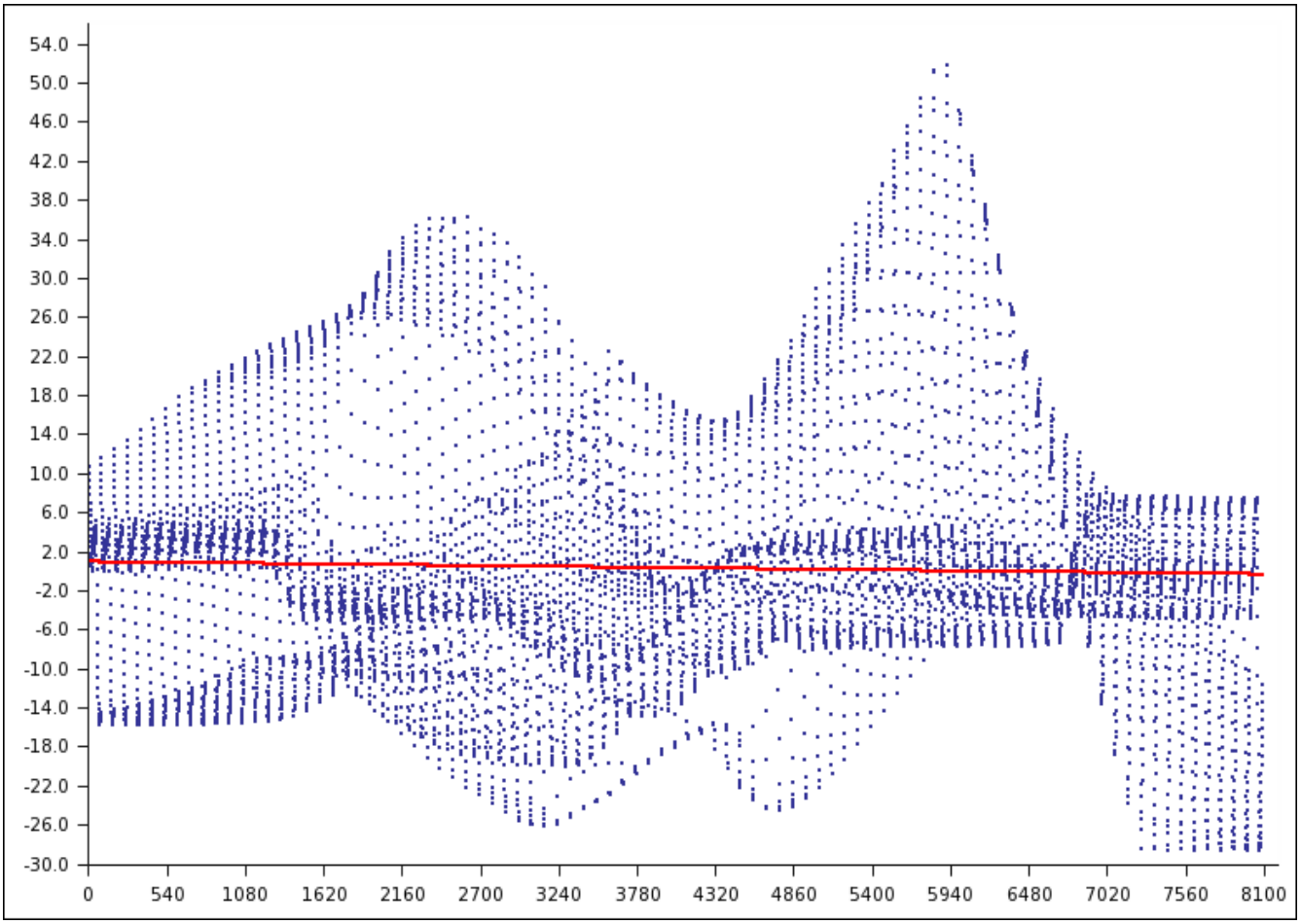


Figura 5.1.11B. Gráficas de las diferencias de distancias en el plano XZ.
Diferencias de distancia: cabeza00_ – e00_01R_ y recta de mínimos cuadrados.
Eje X: Numeración de los 8100 puntos en el plano XY.
Eje Y: Valores de diferencias de distancias (en milímetros)

➔➔➔ ***La ecuación de la recta de mínimos cuadrados XZ es la siguiente:***

$$y_{XZ} = -0.000157958207608361\,x + 0.8543446749$$

La ecuación representa al comportamiento del error en el plano XZ: la pendiente distinta de cero indica que existen diferencias de distancias con valores distintos, mientras que la ordenada al origen es una interpretación del promedio de las diferencias de distancias entre los modelos de cabeza y elipsoide. El valor del error en el plano XZ es la suma de diferencias cuadráticas de distancias:

$$E^2_{XZ} = 1246445.8969547060\ [\text{mm}^2], \qquad E_{XZ} = 1116.4434141302\ [\text{mm}].$$

En este plano (al ser el sagital) se refleja a mayor grado la influencia de los outliers (los puntos alejados, destacando nariz y mentón), ya que son los factores críticos que causan la obtención de los mayores valores de error, además del corte en el cuello y casquete coronal.

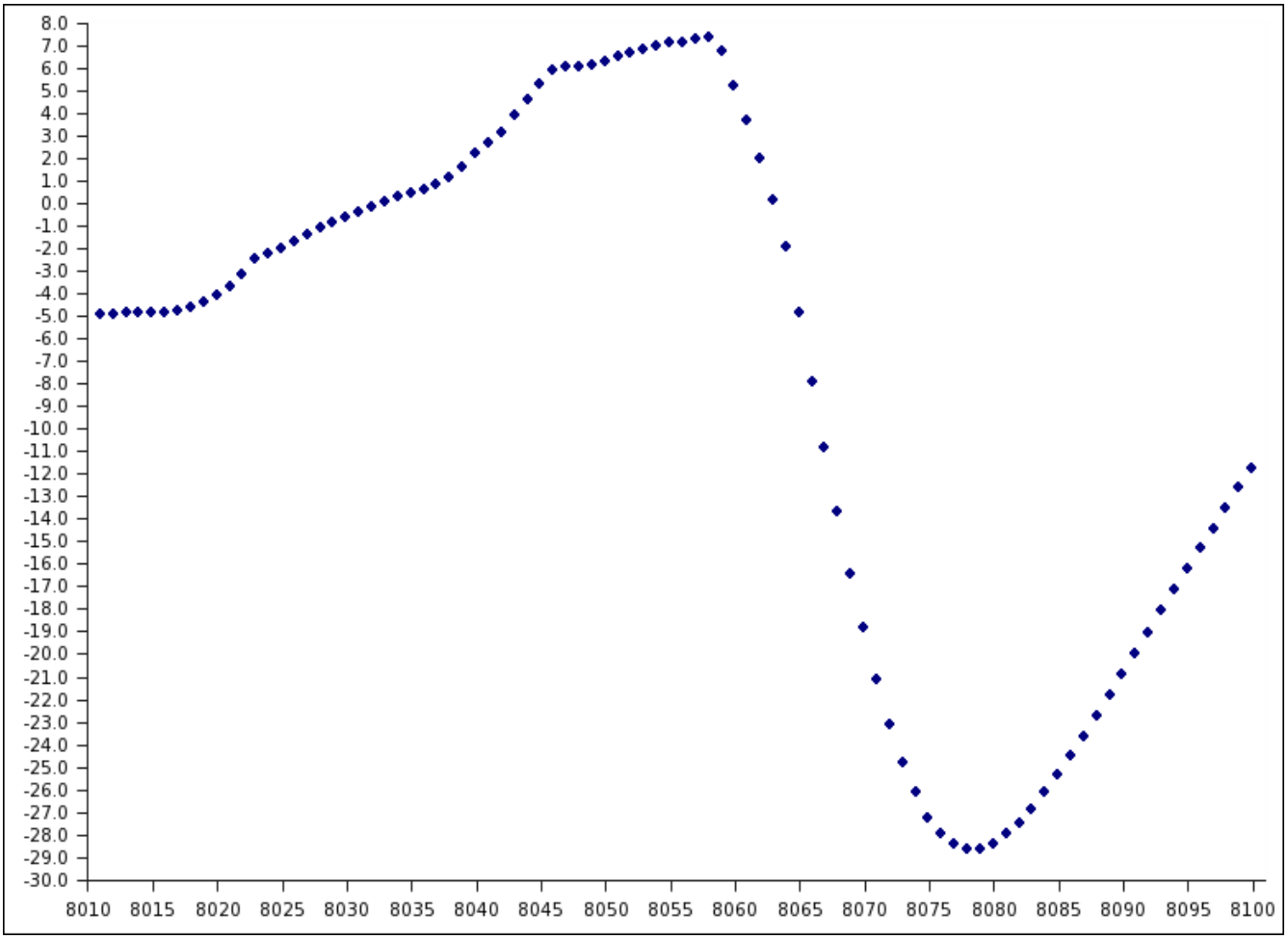


Figura 5.1.11B1. Gráfica de las diferencias de distancias numeradas 8011 a 8100, entre el modelo cabeza00_.pi y el modelo e00_01R_.pi, en el plano XZ.

En la *fig.5.1.11B1* se observa que la mayoría de los valores de diferencia de distancias es negativo, significando que en este renglón 90, es mayor la distancia reportada por el modelo de elipsoide. Lo anterior es otra forma de interpretar a un candidato de outlier: si un punto, ubicado en el exterior de ambos modelos, presenta mayor distancia en el modelo de elipsoide, entonces en el modelo de cabeza existen puntos candidatos a outlier al estar fuera del modelo de elipsoide (razón por la cual reportan menor distancia).

✓ ***El análisis de error del plano XZ se almacenó en el archivo: cabeza00_01_XZ.gnumeric***

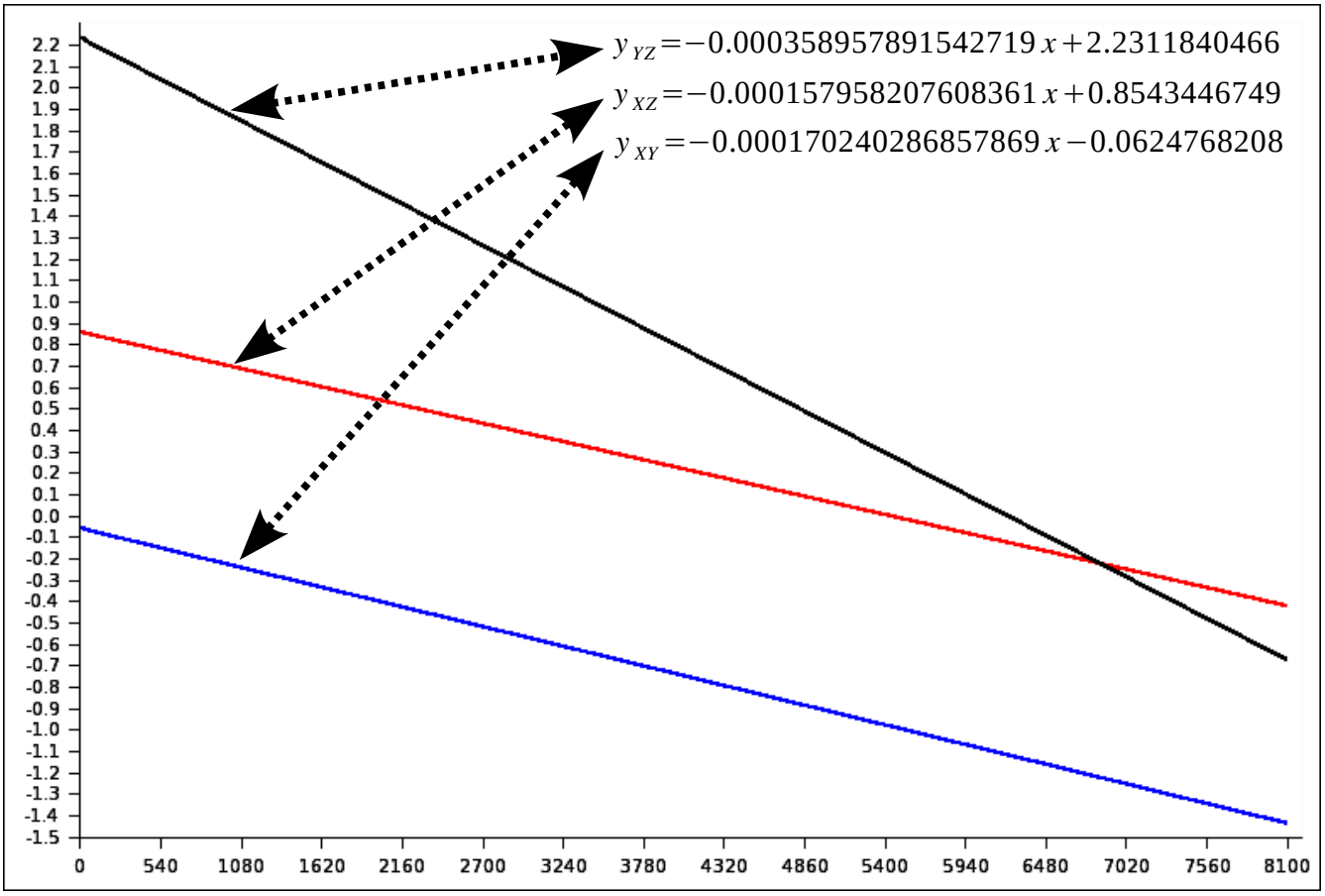


Figura 5.1.12. Ecuaciones de regresión lineal por mínimos cuadrados, para el criterio de similitud entre los modelos cabeza00_.pi y e00_01R_.pi.

	YZ	*XY*	*XZ*
Media [mm]	**0.7772251069**	**-0.7520351027**	**0.2145349550**
Mediana [mm]	**3.1525823384**	**-0.2133106737**	**-0.8195465532**
Desviación estándar [mm]	**6.4861688924**	**5.6953810076**	**12.4038372579**
Varianza [mm²]	**42.0703869005**	**32.4373648219**	**153.8551787202**

Valores estadísticos en la primera iteración del modelo cabeza00.pi Se observa que el plano sagital (XZ) reporta mayor desviación estándar aunque presentó la media con mayor aproximación al valor ideal: cero.

Ahora ya se cuenta con los valores de error correspondientes a la primera iteración,

Muestra del campo de distancia	Error cuadrático [mm²]	Error [mm]	Error [mm] (iteración previa)
YZ	*345621.1023283845*	***587.895485888763***	*NA*
XY	*267291.2277374494*	***517.002154480471***	*NA*
XZ	*1246445.8969547060*	***1116.4434141302***	*NA*

Tabla 5.1.3. Cantidades de error en la primera iteración.

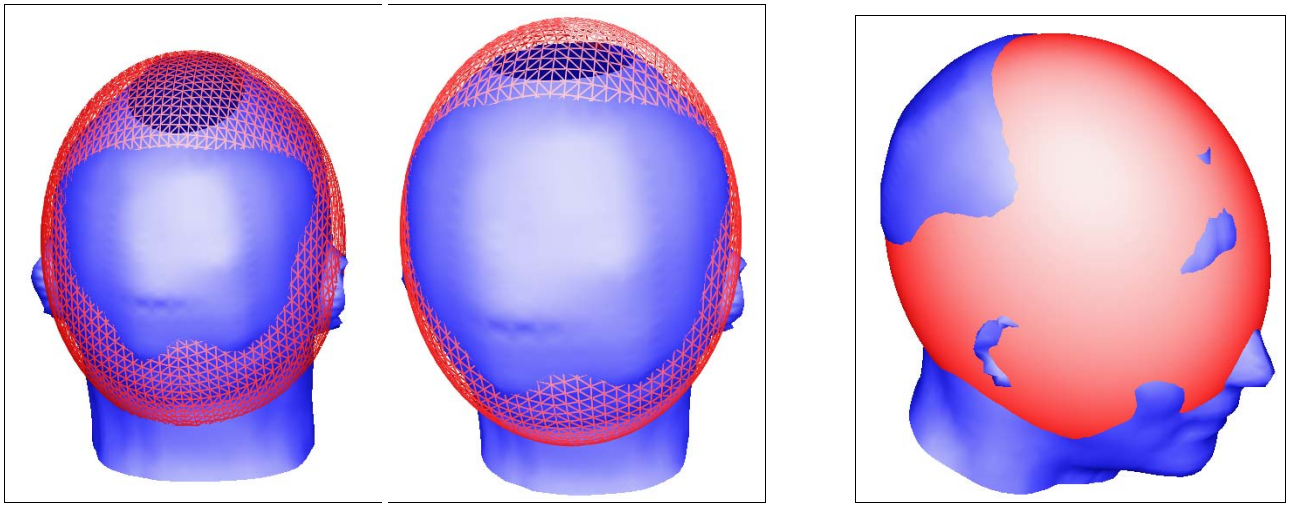


Figura 5.1.13A. Visualización de los modelos cabeza00_.pi y e00_01R_.pi. Se observa que la elipsoide equivalente tiende a completar al casquete del modelo de cabeza. Para mejorar la aproximación (y preparar el espacio de trabajo de la siguiente iteración) se debe aplicar la supresión de outliers. Izquierda: Inspección posterior con proyección ortogonal y proyección Frustum. Derecha: Inspección lateral con proyección ortogonal. Todas las proyecciones tienen normales Gouraud.

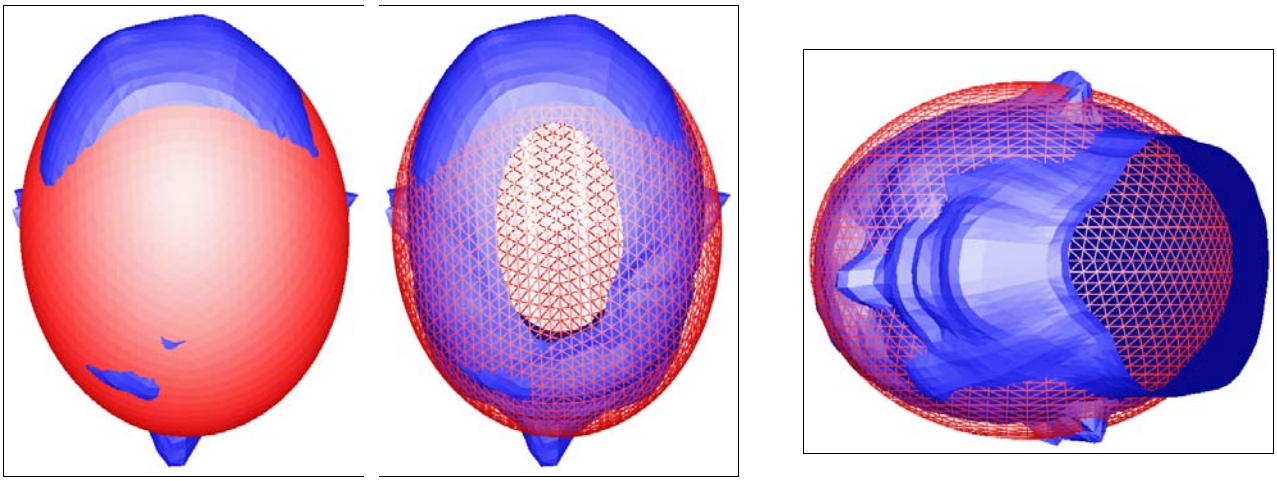


Figura 5.1.13B. Visualización de los modelos cabeza00_.pi y e00_01R_.pi. Izquierda: Inspección superior. Derecha: Inspección inferior. Proyecciones ortogonales y normales planas.

Paso opcional ----------

De forma auxiliar, para visualizar y definir qué puntos del modelo de cabeza son considerados como posibles outliers en la iteración, se puede emplear al plano auxiliar en YZ (plano.pi). Para ello se debe editar al modelo plano.pi para que el valor X de sus cuatro puntos sea el valor X del centroide de la elipsoide equivalente trasladada (en este caso -12.9551202384), ya que como se mencionó previamente (sección 4.6), la supresión de outliers consiste en su traslación y se pretende alterar de forma extraordinaria a los puntos del modelo de cabeza ubicados en las secciones frontal, parietal y occipital del cráneo. En caso de ser necesario, se pueden aplicar traslaciones y rotaciones adicionales al modelo plano.pi, con el fin de mejorar al criterio de supresión de outliers.

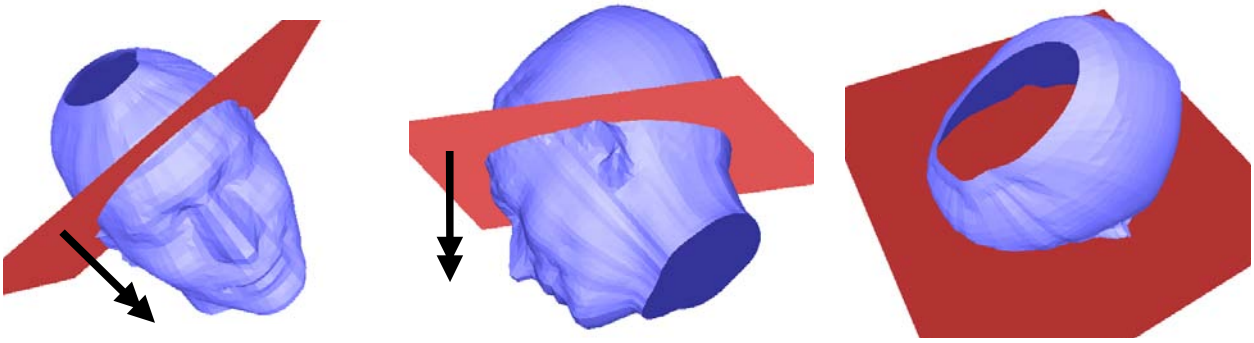


Figura 5.1.14. Posibles outliers en el modelo cabeza00_.pi, indicados con la flecha que señala el corte del plano YZ, cuyo valor X es el mismo valor que tiene la coordenada X del centroide de la elipsoide equivalente. Izquierda: Inspección frontal con proyección Frustum. Centro: Inspección lateral con proyección ortogonal. Derecha: Inspección superior (puntos que no son posibles outliers) con proyección Frustum. Normales planas en los tres casos.

Paso opcional ----------

☞ ***El siguiente paso es aplicar supresión de outliers al modelo de cabeza. Desde consola, en la carpeta de programas se ejecuta al shell sup_out.sh***

./sup_out.sh 00 01

➔➔➔ *Respaldo de algunos archivos resultantes de la primera iteración*

Backup of cabeza00.pi iteration's files...

mv ./00/cabeza00.pi ./00/cNN/cabeza00_01.pi
mv ./00/cabeza00EHR_.pi ./00/eNN/cabeza00EHR_01.pi
mv ./00/cabeza00EHR_.xml ./00/eNN/cabeza00EHR_01.xml
mv ./00/cabeza00EHR.pi ./00/eNN/cabeza00EHR01.pi
mv ./00/e00_01.pi ./00/eNN/e00_01.pi
mv ./00/e00_01R.pi ./00/eNN/e00_01R.pi
mv ./00/e00_01R_.pi ./00/eNN/e00_01R_.pi
mv ./00/e00_01R_.xml ./00/eNN/e00_01R_.xml

➔➔➔ *Instrucciones para aplicar la supresión de outliers*

Use the abc and traslation values to apply outliers criteria with the next program:

./aproximaElipsoideX0Y0Z0ConPlano
ERROR1, Bad number of parameters...

Sintaxis:
./aproximaElipsoideX0Y0Z0ConPlano File1 File2 x1 y1 z1 x2 y2 z2 x3 y3 z3 a b c X0 Y0 Z0
Where:
File1 - original PIF model's file.
File2 - resulting PIF model's file after the adaptation defined by the Plane and Ellipsoid.
x1, y1, z1 - Point1 in the plane.
x2, y2, z2 - Point2 in the plane.
x3, y3, z3 - Point3 in the plane.
a, b, c - Ellipsoid's axes values
X0, Y0, Z0 - Ellipsoid's origin

Waldo Ramirez Montano. Thesis 2005.

✓ ***Es posible aplicar la supresión de outliers en el modelo de cabeza al conocer los parámetros indicados.***

☞ ***Desde consola, en la carpeta de programas se ejecuta:***

```
./aproximaElipsoideX0Y0Z0ConPlano ./00/cabeza00_.pi     ➔➔➔ Modelo origen
                        ./00/cabeza00.pi            ➔➔➔ Modelo resultante
                        -12.9551202384 0.0 0.0      ➔➔➔ Primer punto del plano
                        -12.9551202384 -1.0 0.0     ➔➔➔ Segundo punto del plano
                        -12.9551202384 0.0 1.0      ➔➔➔ Tercer punto del plano
                        120.1796439751   ➔➔➔ a de la elipsoide equivalente
                        86.2122665308    ➔➔➔ b de la elipsoide equivalente
                        104.8086303998   ➔➔➔ c de la elipsoide equivalente
                        -12.9551202384   ➔➔➔ X0 de la elipsoide equivalente
                        0.0              ➔➔➔ Y0 de la elipsoide equivalente
                        0.0              ➔➔➔ Z0 de la elipsoide equivalente
```

Creating the output file: "./00/cabeza00.pi"...
INFO-pif: Succesfull file opening, obtaining model's name... "./00/cabeza00__", done.
INFO-pif: Max. file's line length: 46
INFO-pif: Getting points from 3D model... done.
INFO-pif: Getting indexes from 3D model... done.
WARNING-pif: Valid dynamic values of the model had not been calculated.
INFO-pif: Closing model's PIF file and freeing memory...

➔➔➔ ***Definición de la ecuacion3D del plano especificado***

The adaptation criteria for the model is defined by the plane:
(1.000000x^1.000000)+(-0.000000y^1.000000)+(0.000000z^1.000000)+(12.955120)=0.0

➔➔➔ ***Criterio de supresión de outliers de acuerdo a la ecuacion3D de la elipsoide equivalente. Aplica a todo outlier que no es justificado por el plano: se le hace traslación que lo aproxima a la elipsoide***

Adapting model according to the defined ellipsoid...
(0.000069x^2.000000)+(0.000135y^2.000000)+(0.000091z^2.000000)+(-1.000000)=0.0
With center in (-12.9551202384, 0.0000000000, 0.0000000000).
1457 accepted points, **1844 adapted points** with plane-ellipsoid criteria; starting polygons (triangles) creation...
Finishing: closing files and freeing memory.

✓ ***Se aplicó supresión de outliers (véase fig.5.1.15) en el modelo de cabeza, con lo cual se creó la versión del modelo de cabeza apta para ser el punto de partida de la siguiente iteración.***

Analíticamente, se considera a un punto del modelo de cabeza como outlier (punto alejado) a aquel que: (1) se encuentra fuera de la elipsoide equivalente (es decir, su evaluación en la ecuación3D de la elipsoide es mayor que cero) y (2) no es justificado por el plano (es decir, su evaluación en la ecuación3D del plano es mayor que cero). De esta manera se obtiene un control adaptable de los puntos candidatos a ser outliers en cada modelo de cabeza en cualquier iteración, ya que en caso de ser necesario, el aplicar transformaciones geométricas adicionales al plano o elipsoide permiten acoplar la elección de outliers de acuerdo al estado del modelo de cabeza.
Una propuesta de mejora para este criterio, es hacer uso de un modelo de justificación más complejo que el plano (por ejemplo, un paraboloide u otra elipsoide).

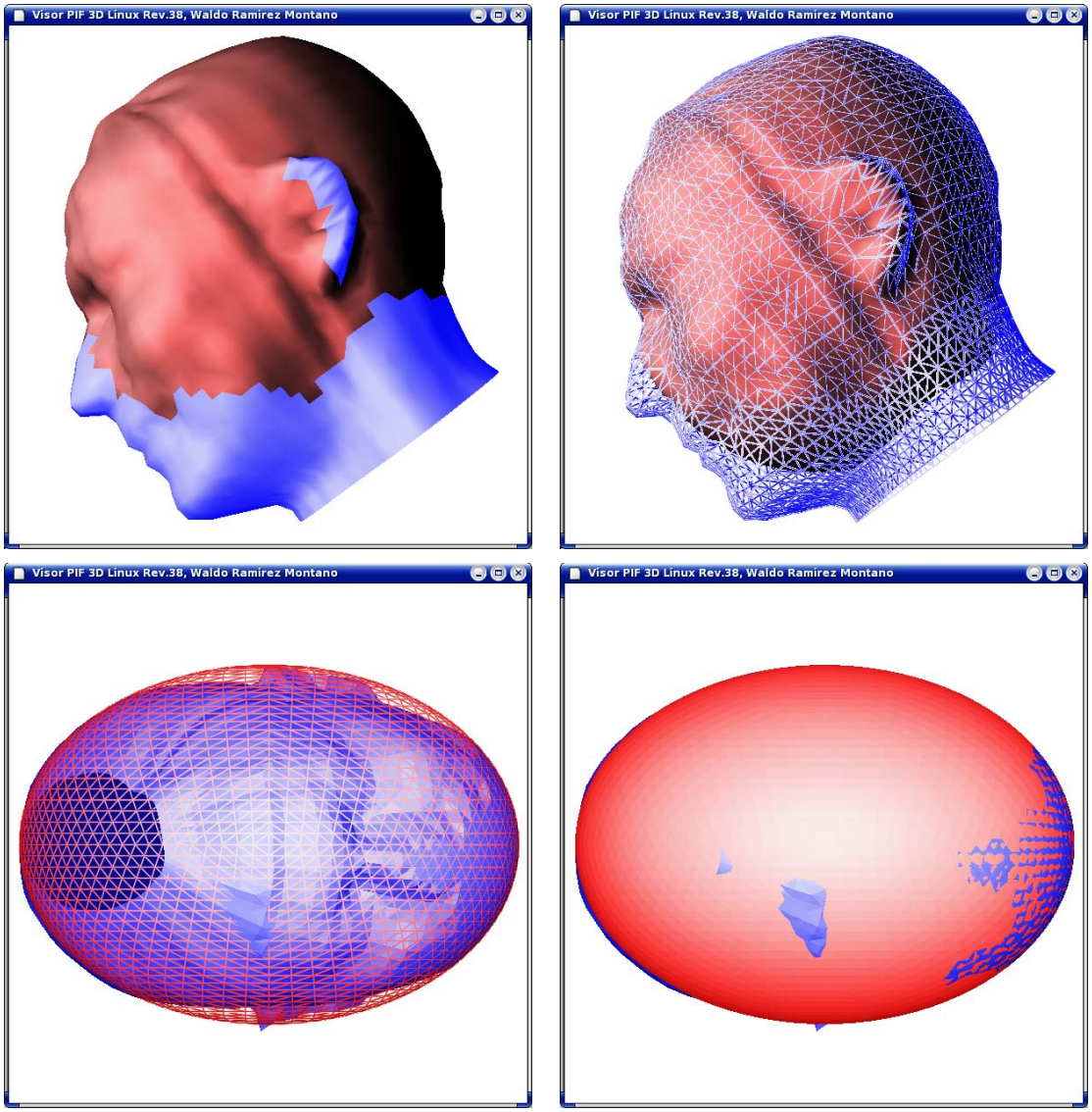


Figura 5.1.15. Verificación de la supresión de outliers en el modelo cabeza00_.pi.
Primera fila: Comparación en el modelo de cabeza: antes y después de la supresión. Proyección Frustum y normales Gouraud.
Segunda fila: Nuevo modelo de cabeza y la elipsoide equivalente. Proyección ortogonal y normales planas.

Para finalizar la iteración, sólo falta hacer el respaldo del resto de los archivos resultantes de la primera iteración:

☞ ***Desde consola, en la carpeta de programas se ejecuta al shell end.sh***
./end.sh 00 01
➔➔➔ ***Respaldo del resto de archivos resultantes de la primera iteración***
Backup of cabeza00_.pi iteration's files...
mv ./00/cabeza00_.pi ./00/cNN/cabeza00_01A.pi
mv ./00/cabeza00_.xml ./00/cNN/cabeza00_01A.xml
End of iteration 01, for model cabeza00.pi...

✓ ***A continuación se presenta la segunda iteración. Se omiten bitácoras de la consola, presentando imágenes y resultados de la iteración.***

☞ ***Desde consola, en la carpeta de programas se ejecuta al shell abc.sh***

```
./abc.sh 00
```

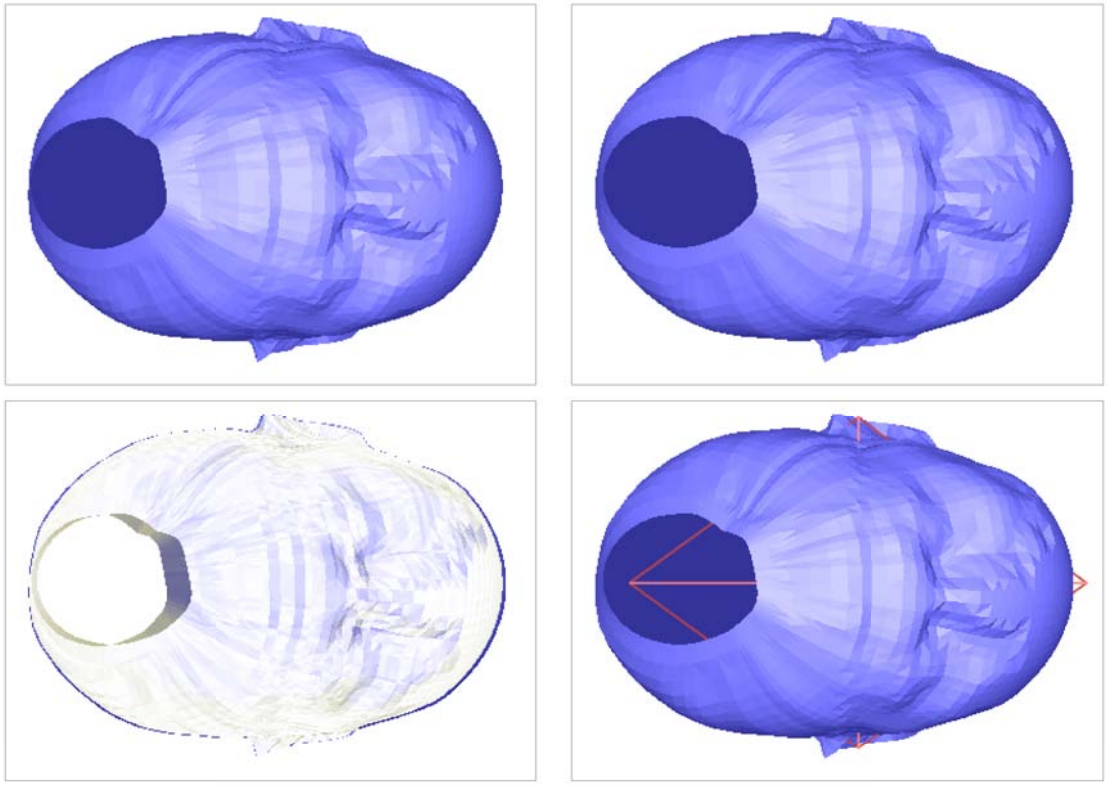


Figura 5.2.1. Alineación del modelo de cabeza y creación de elipsoide equivalente. Primera fila. Izquierda: cabeza00.pi no alineado. Derecha: cabeza00.pi alineado. Segunda fila. Izquierda: Diferencia de las imágenes izquierda y derecha de la primera fila (mediante GIMP). Derecha: Modelo cabeza00.pi junto con su elipsoide equivalente nivel 1 (cabeza00EHR_.pi) de la segunda iteración.

Para poder apreciar a detalle la variación del modelo de cabeza no alineado con su nueva versión alineada, mediante GIMP se creó una imagen de dos capas: la capa base corresponde al modelo de cabeza no alineado, mientras que la capa superior corresponde al modelo de cabeza alineado. Finalmente, se aplicó el modo de "Diferencia" en la capa superior, con lo cual se destacan los cambios hechos por las transformaciones de alineación.

☞ ***A continuación se obtiene la elipsoide equivalente nivel 25, 2da. Iteración. Desde consola, en la carpeta de programas se ejecuta al shell e25.sh***

```
./e25.sh 00 02
```

Elipsoide equivalente	Valor de iteración actual [mm]	Valor de iteración previa [mm]
A	***115.4818418142***	120.1796439751
B	***84.0051287103***	86.2122665308
C	***101.5742981853***	104.8086303998

Tabla 5.2.1. Obtención de elipsoide equivalente nivel 25. Los nuevos valores tienden a ser menores gracias a la supresión de outliers.

☞ ***A continuación se traslada al modelo de elipsoide para buscar completar al casquete. Desde consola, en la carpeta de programas se ejecuta al shell max_ev.sh***

```
./max_ev.sh 00 02
```

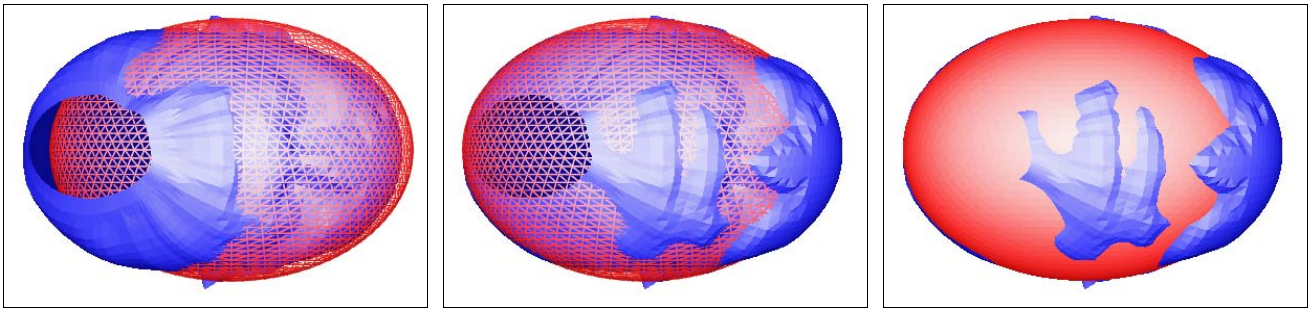


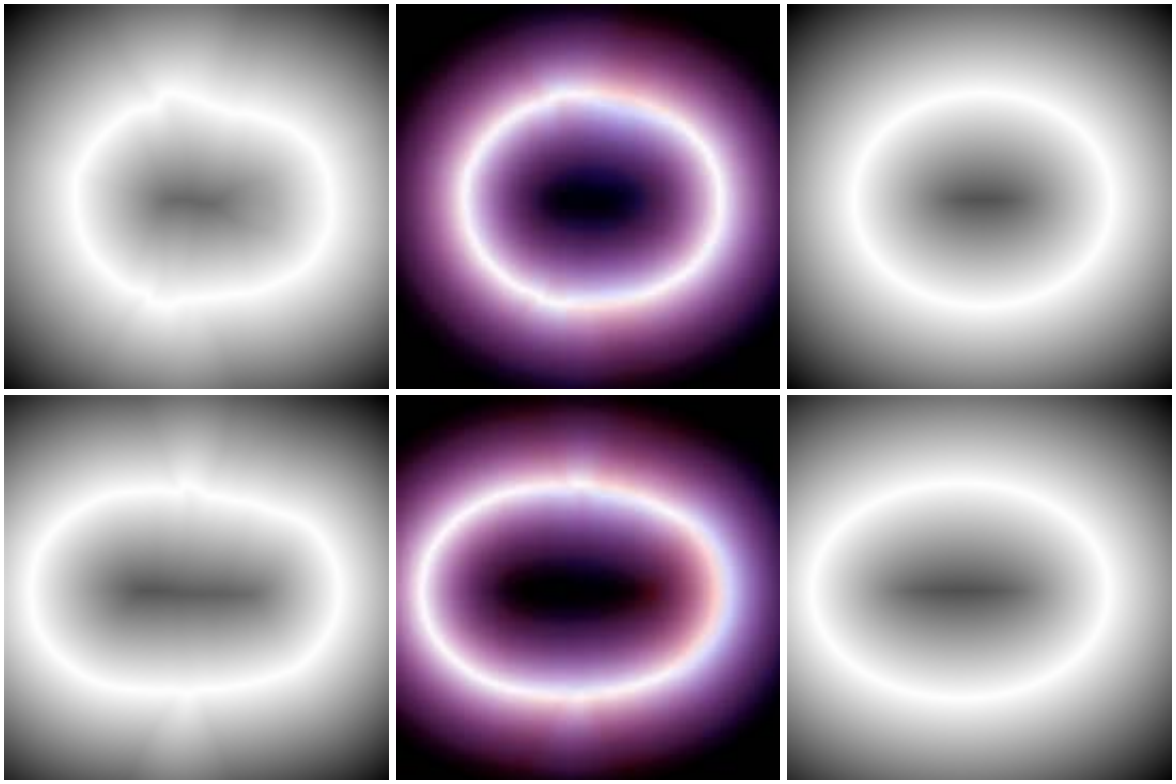
Figura 5.2.2. Traslación de la elipsoide equivalente al máximo punto del casquete. Izquierda: Antes de la traslación. Centro: Después de la traslación. Derecha: Después de la traslación (ambos modelos en modo sólido). Proyecciones ortogonales y normales planas.

Centroide de la elipsoide equivalente trasladada	Valores de iteración actual [mm]	Valores de iteración previa [mm]
X	***-17.4382861611***	-12.9551202384
Y	***0.0***	0.0
Z	***0.0***	0.0

Tabla 5.2.2. Traslación de la elipsoide equivalente.

☞ ***Desde consola, en la carpeta de programas***

```
./distanceField.sh 00 02
```



(fig. continua ...)

Figura 5.2.3. Imágenes representativas de las seis muestras del campo de distancia de los modelos cabeza00_.pi y e00_01R_.pi (segunda iteración). Primera fila: Plano YZ (axial). Segunda fila: Plano XY (coronal). Tercera fila: Plano XZ (sagital). La primera columna corresponde al modelo de cabeza, la tercera columna corresponde al modelo de elipsoide, mientras que la segunda columna corresponde a la comparación entre muestras de cabeza y elipsoide, hechas mediante GIMP. El tamaño de las seis imágenes es de 90X90 [pixels].

☞ ***Utilizando la hoja de cálculo preparada en Gnumeric 1.2.6***

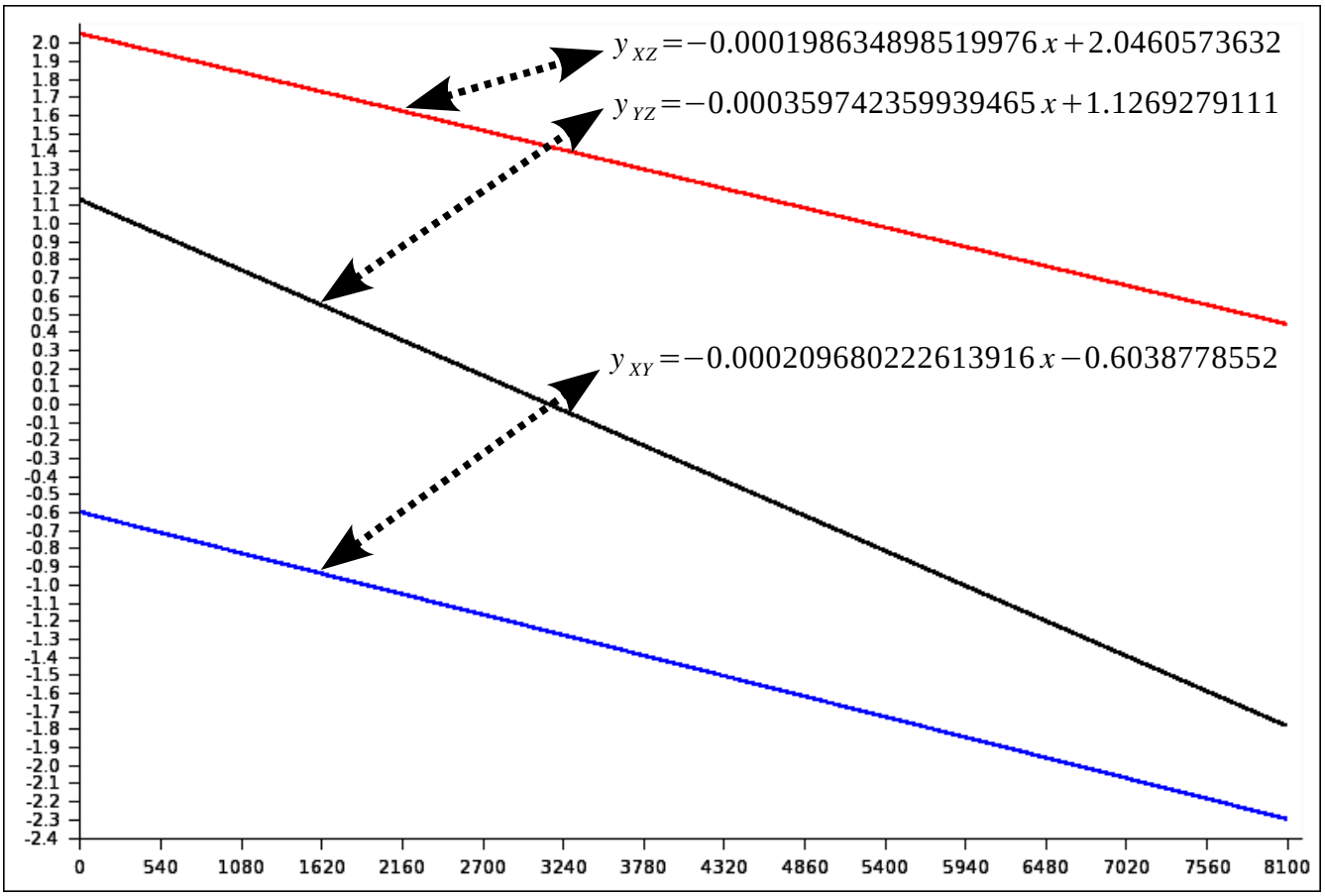


Figura 5.2.4. Ecuaciones de regresión lineal por mínimos cuadrados, para el criterio de similitud entre los modelos cabeza00_.pi y e00_02R_.pi.

	YZ	*XY*	*XZ*
Media [mm]	***-0.3302085178***	***-1.4531875969***	***1.2414867067***
Mediana [mm]	***-0.1426627721***	***-0.8062278171***	***-0.6115212671***
Desviación estándar [mm]	***3.8376232645***	***4.7805254025***	***9.0449059872***
Varianza [mm²]	***14.7273523200***	***22.8534231241***	***81.8103243176***

Valores estadísticos en la segunda iteración del modelo cabeza00.pi.

Muestra del campo de distancia	Error cuadrático [mm²]	Error [mm]	Error [mm] (iteración previa)
YZ	*120160.0315278027*	***346.641070168846***	*587.895485888763*
XY	*202195.0828344719*	***449.661075516296***	*517.002154480471*
XZ	*675066.2595164579*	***821.624159525788***	*1116.4434141302*

Tabla 5.2.3. Cantidades de error en la segunda iteración.

✓ ***Se presentó menor error en la segunda iteración.***

☞ ***Desde consola, para preparar a la supresión de outliers, en la carpeta de programas se ejecuta al shell sup_out.sh***

```
./sup_out.sh 00 02
```

➔➔➔ ***Respaldo de algunos archivos resultantes de la segunda iteración***

Backup of cabeza00.pi iteration's files...

☞ ***Desde consola, en la carpeta de programas***

```
./aproximaElipsoideX0Y0Z0ConPlano ./00/cabeza00_.pi ./00/cabeza00.pi
                                  -17.4382861611 0.0 0.0
                                  -17.4382861611 -1.0 0.0
                                  -17.4382861611 0.0 1.0
                                  115.4818418142 84.0051287103 101.5742981853
                                  -17.4382861611 0.0 0.0
```

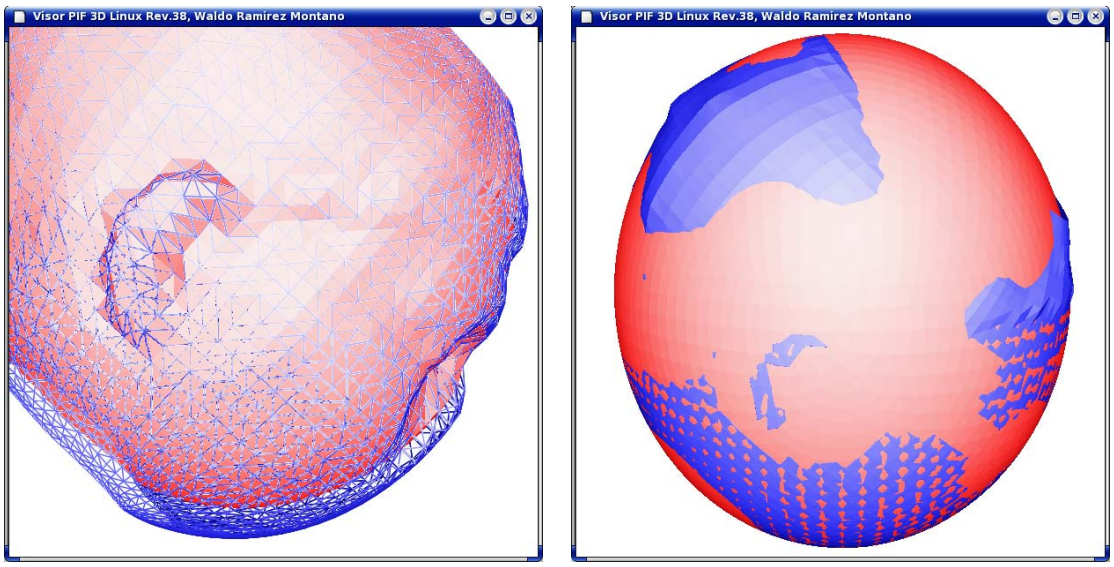


Figura 5.2.5. Inspección de la supresión de outliers, segunda iteración. Izquierda: Mayor cantidad de outliers en nariz y mentón. Derecha: Se observa que también se trasladó a una porción de la frente. Proyecciones ortogonales y normales planas.

☞ ***Para finalizar iteración, en la carpeta de programas se ejecuta al shell end.sh***
./end.sh 00 02
➔➔➔ *Respaldo del resto de archivos resultantes de la segunda iteración*
Backup of cabeza00_.pi iteration's files...

Debido al decremento de error obtenido en la segunda iteración (ver tabla 5.2.3), es viable continuar con la tercera iteración.

☞ ***Desde consola, en la carpeta de programas se ejecuta al shell abc.sh***
./abc.sh 00

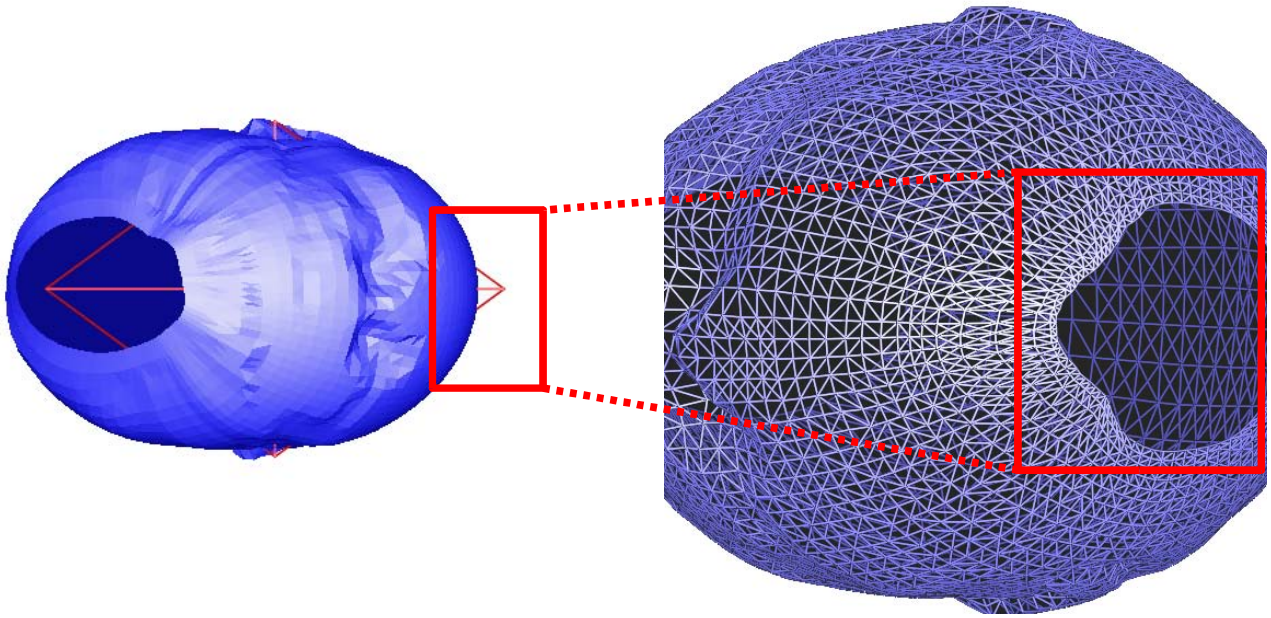


Figura 5.3.1. Alineación del modelo de cabeza y creación de elipsoide equivalente. En este iteración se refleja la sensibilidad del tensor de inercia: debido a la supresión de outliers, diversos puntos generan mayor densidad de masa en la zona inferior del modelo de cabeza, con lo cual la elipsoide equivalente le compensa con la extensión geométrica de un eje principal.

☞ ***Desde consola, en la carpeta de programas se ejecuta al shell e25.sh***
./e25.sh 00 03

Elipsoide equivalente	Valor de iteración actual [mm]	Valor de iteración previa [mm]
A	***112.1432496684***	115.4818418142
B	***82.0785744462***	84.0051287103
C	***99.4812637698***	101.5742981853

Tabla 5.3.1. Obtención de elipsoide equivalente nivel 25. Los nuevos valores tienden a ser menores gracias a la supresión de outliers.

☞ ***Desde consola, en la carpeta de programas se ejecuta al shell max_ev.sh***
./max_ev.sh 00 03

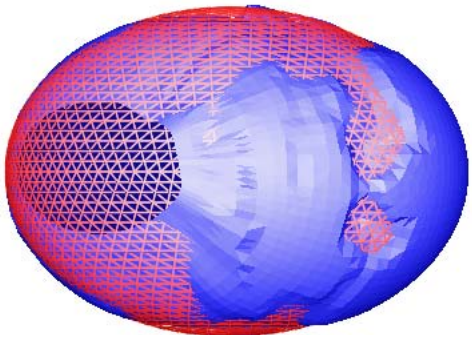


Figura 5.3.2. Traslación de la elipsoide equivalente a la región del casquete del modelo de cabeza, en la tercera iteración. Proyecciones ortogonales y normales planas.

Centroide de la elipsoide equivalente trasladada	Valores de iteración actual [mm]	Valores de iteración previa [mm]
X	***-19.6801026902***	-17.4382861611
Y	***0.0***	0.0
Z	***0.0***	0.0

Tabla 5.3.2. Traslación de la elipsoide equivalente.

☞ ***Desde consola, en la carpeta de programas se ejecuta al shell distanceField.sh***

```
./distanceField.sh 00 03
```

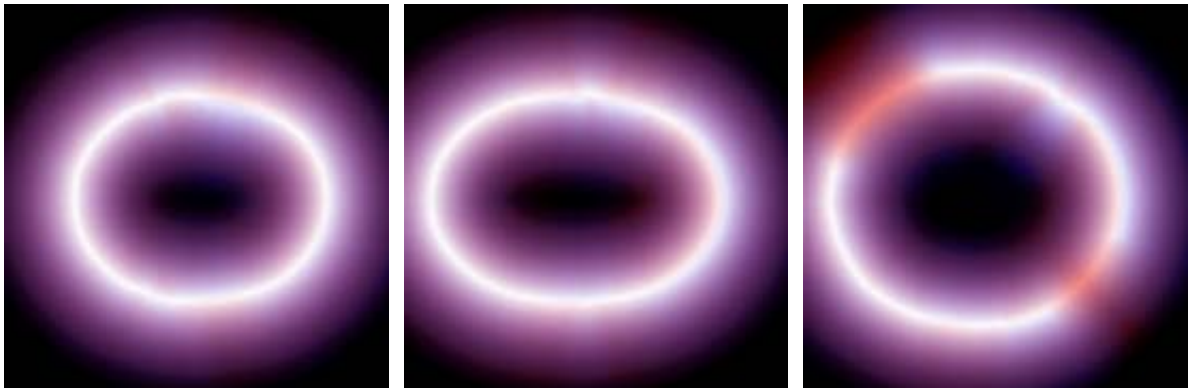


Figura 5.3.3. Muestras del campo de distancia para el criterio de similitud entre los modelos cabeza00_.pi y e00_03R_.pi. Izquierda: plano YZ, Centro: plano XY, Derecha: plano XZ.

$$y_{YZ} = -0.000282387160509217\,x + 0.3576642401$$

$$y_{XY} = -0.000117236215432878\,x - 0.8363753117$$

$$y_{XZ} = -0.000256998184331835\,x + 2.6014494892$$

	YZ	*XY*	*XZ*
Media [mm]	***-0.7861449535***	***-1.3112406023***	***1.5604783436***
Mediana [mm]	***-0.7988134834***	***-0.9733155811***	***-0.5949545967***
Desviación estándar [mm]	***2.5448529506***	***3.3470815216***	***7.8907846336***
Varianza [mm^2]	***6.4762765401***	***11.2029547122***	***62.2644821337***

Valores estadísticos en la tercera iteración del modelo cabeza00.pi.

Muestra del campo de distancia	Error cuadrático [mm^2]	Error [mm]	Error [mm] (iteración previa)
YZ	*57457.3571910339*	***239.702643270853***	*346.641070168846*
XY	*104659.4807423333*	***323.511175606552***	*449.661075516296*
XZ	*524004.2913543154*	***723.881406968238***	*821.624159525788*
Tabla 5.3.3. Cantidades de error en la tercera iteración.			

✓ ***Se presentó menor error en la tercera iteración.***

☞ ***Desde consola, en la carpeta de programas se ejecuta al shell sup_out.sh***

```
./sup_out.sh 00 03
```

Backup of cabeza00.pi iteration's files...

☞ ***Desde consola, en la carpeta de programas***

```
./aproximaElipsoideX0Y0Z0ConPlano ./00/cabeza00_.pi
                                  ./00/cabeza00.pi
                                  -19.6801026902 0.0 0.0
                                  -19.6801026902 -1.0 0.0
                                  -19.6801026902 0.0 1.0
                                  112.1432496684 82.0785744462 99.4812637698
                                  -19.6801026902 0.0 0.0
```

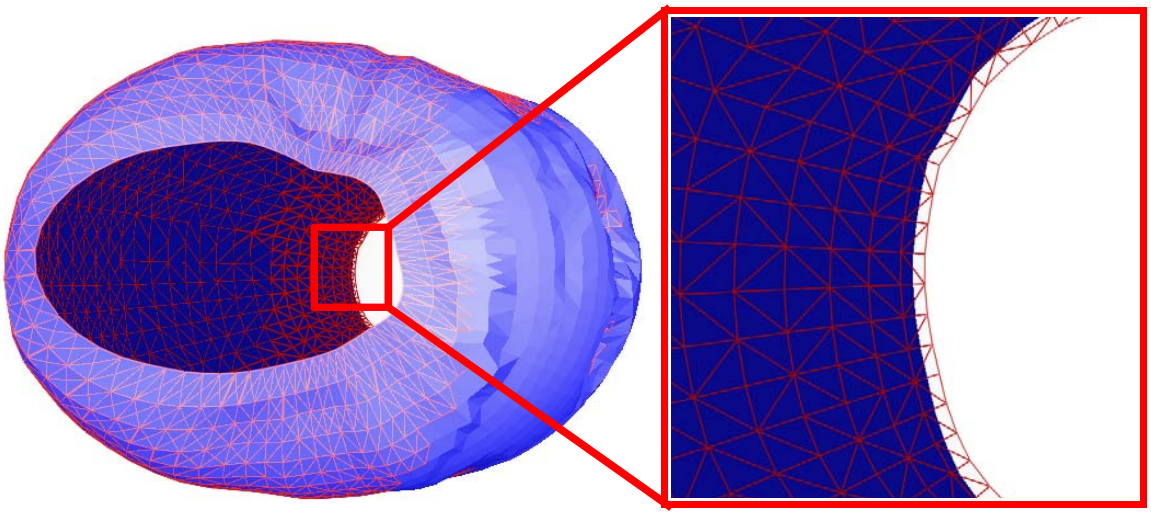


Figura 5.3.4. Supresión de outliers. Se observa que gracias a la sensibilidad del tensor de inercia, el ajuste en la parte inferior del modelo de cabeza00_.pi es menor, y se incrementa el ajuste en la sección frontal del modelo.

☞ ***Para finalizar iteración, en la carpeta de programas se ejecuta al shell end.sh***

```
./end.sh 00 03
```

➔➔➔ ***Respaldo del resto de archivos resultantes de la tercera iteración***

Backup of cabeza00_.pi iteration's files...

✓ ***A continuación se presentan los resultados de la cuarta iteración.***

☞ ***Desde consola, en la carpeta de programas se ejecuta al shell abc.sh***
`./abc.sh 00`

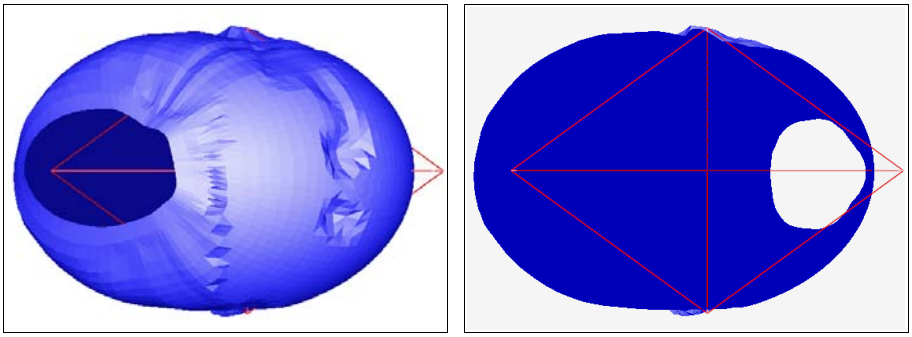


Figura 5.4.1. Alineación del modelo de cabeza y obtención de su elipsoide equivalente, mediante el programa Visor3D. Izquierda: Modelo cabeza00.pi alineado junto con su elipsoide equivalente nivel 1. Derecha: Corte visual sobre el plano XY. Proyecciones ortogonales y normales planas.

☞ ***Desde consola, en la carpeta de programas se ejecuta al shell e25.sh***
`./e25.sh 00 04`

Elipsoide equivalente	Valor de iteración actual [mm]	Valor de iteración previa [mm]
A	***109.8367643760***	112.1432496684
B	***80.2783416933***	82.0785744462
C	***97.5761119669***	99.4812637698
Tabla 5.4.1. Obtención de elipsoide equivalente nivel 25.		

☞ ***Desde consola, en la carpeta de programas se ejecuta al shell max_ev.sh***
`./max_ev.sh 00 04`

Centroide de la elipsoide equivalente trasladada	Valores de iteración actual [mm]	Valores de iteración previa [mm]
X	***-20.9976596937***	-19.6801026902
Y	***0.0***	0.0
Z	***0.0***	0.0
Tabla 5.4.2. Traslación de la elipsoide equivalente.		

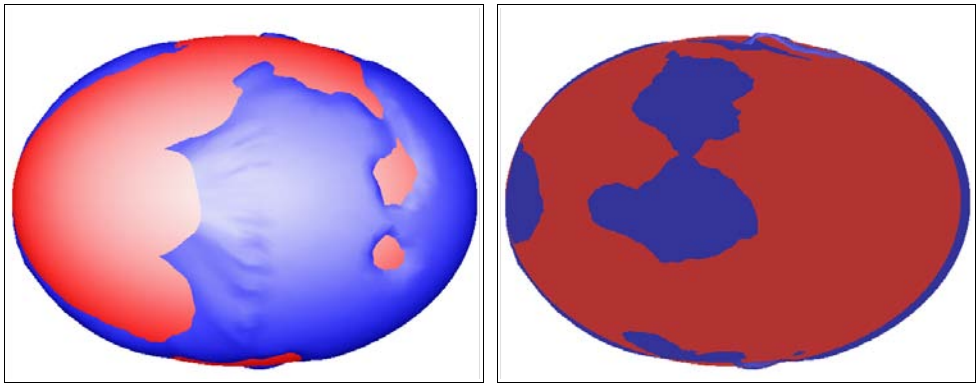


Figura 5.4.2. Traslación de la elipsoide equivalente mediante el programa Visor3D.

Izquierda: Después de la traslación. Derecha: Corte visual en el plano XY. Proyecciones ortogonales y normales Gouraud.

☞ ***Desde consola, en la carpeta de programas se ejecuta al shell distanceField.sh***
`./distanceField.sh 00 04`

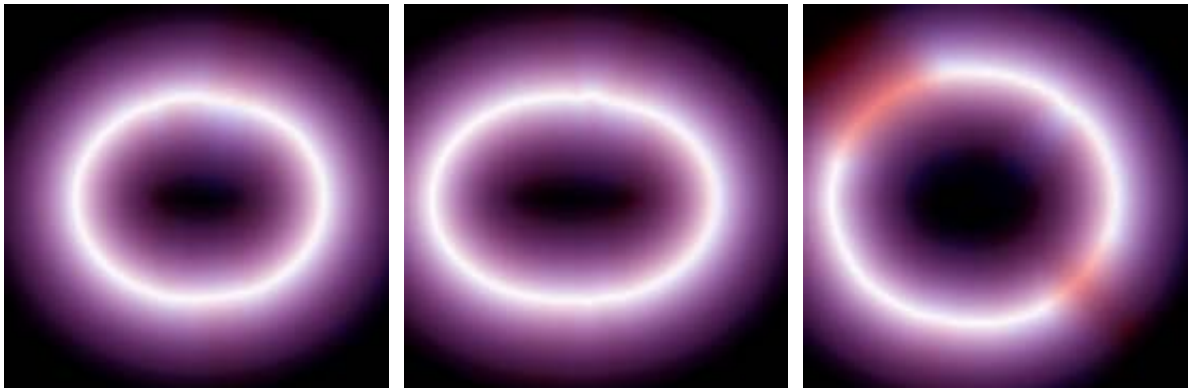


Figura 5.4.3. Muestras del campo de distancia para el criterio de similitud entre los modelos cabeza00_.pi y e00_04R_.pi. Izquierda: Plano YZ, Centro: Plano XY, Derecha: Plano XZ.

$$y_{YZ} = -0.000175454030315533\,x - 0.3767744301$$

$$y_{XY} = -0.000056567535329936 2\,x - 0.9564324407$$

$$y_{XZ} = -0.000307951618268847\,x + 3.0470572387$$

	YZ	*XY*	*XZ*
Media [mm]	***-1.0874509799***	***-1.1855592426***	***1.7996992089***
Mediana [mm]	***-1.6877051243***	***-1.6070902421***	***-0.2703255625***
Desviación estándar [mm]	***2.0891238674***	***2.3619404074***	***7.2600605734***
Varianza [mm^2]	***4.3644385334***	***5.5787624880***	***52.7084795298***

Valores estadísticos en la cuarta iteración del modelo cabeza00.pi.

Muestra del campo de distancia	Error cuadrático [mm^2]	Error [mm]	Error [mm] (iteración previa)
YZ	*44926.2397155207*	***211.958108397675***	*239.702643270853*
XY	*56567.3582037719*	***237.838933322053***	*323.511175606552*
XZ	*453121.2053777508*	***673.142782311265***	*723.881406968238*

Tabla 5.3.3. Cantidades de error en la cuarta iteración.

✓ ***Se presentó menor error en la cuarta iteración.***

☞ ***Desde consola, en la carpeta de programas se ejecuta al shell sup_out.sh***
`./sup_out.sh 00 04`
Backup of cabeza00.pi iteration's files...

☞ ***Desde consola, en la carpeta de programas***

```
./aproximaElipsoideX0Y0Z0ConPlano ./00/cabeza00_.pi
                                  ./00/cabeza00.pi
                                  -20.9976596937 0.0 0.0
                                  -20.9976596937 -1.0 0.0
                                  -20.9976596937 0.0 1.0
                                  109.8367643760 80.2783416933 97.5761119669
                                  -20.9976596937 0.0 0.0
```

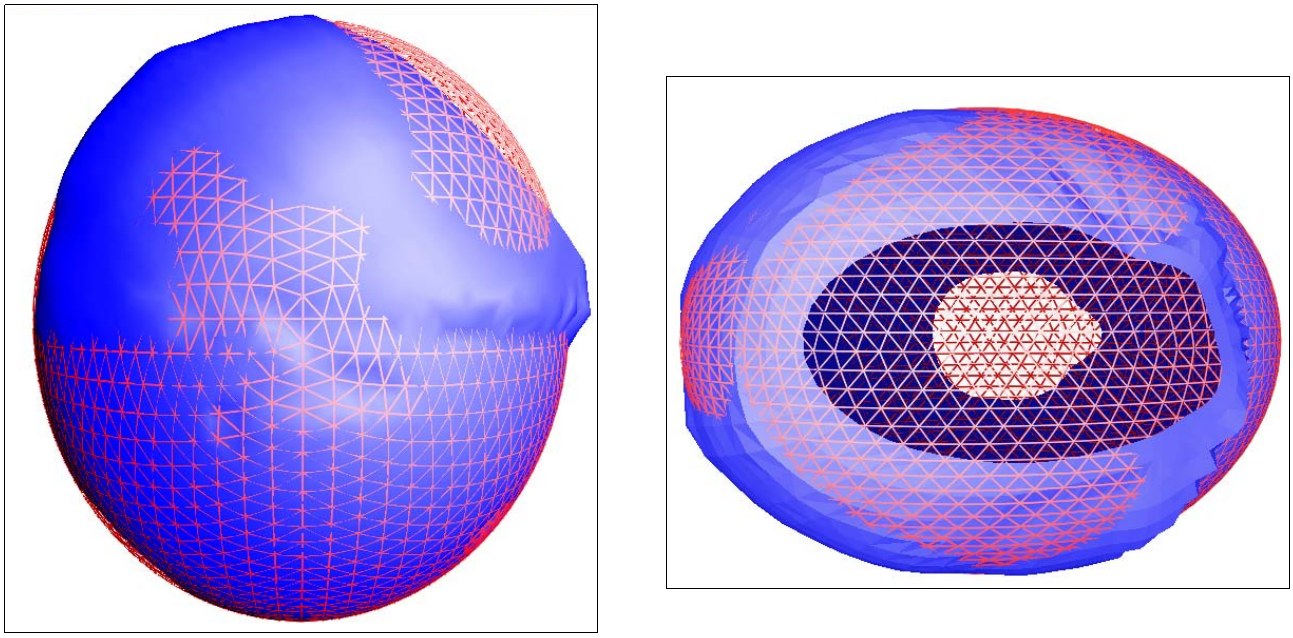


Figura 5.4.4. Resultado de la supresión de outliers en la cuarta iteración. Izquierda: Perspectiva lateral. Derecha: Perspectiva superior. Se aprecia que para completar el casquete de forma ideal, se requiere que el corte superior en el modelo de cabeza fuese elíptico; en este caso el corte tiende a ser elíptico, y sus variaciones causan divergencia con el modelo de elipsoide. Proyecciones Frustum con normales Gouraud (izquierda) y planas (derecha).

☞ ***Para finalizar iteración, en la carpeta de programas se ejecuta al shell end.sh***

```
./end.sh 00 04
```

➔➔➔ ***Respaldo del resto de archivos resultantes de la cuarta iteración***

Backup of cabeza00_.pi iteration's files...

Se refleja que en los modelos de ajuste robusto su máxima variación morfológica es limitada por el segundo grado de su ecuación analítica: tanto para esferas ó elipsoides. Esto abre la posibilidad de considerar modelos de mayor grado ó de perfil discreto, para obtener un mejor ajuste morfológico.

✓ ***A continuación se presentan los resultados de la quinta iteración.***

☞ ***Desde consola, en la carpeta de programas se ejecuta al shell abc.sh***
./abc.sh 00

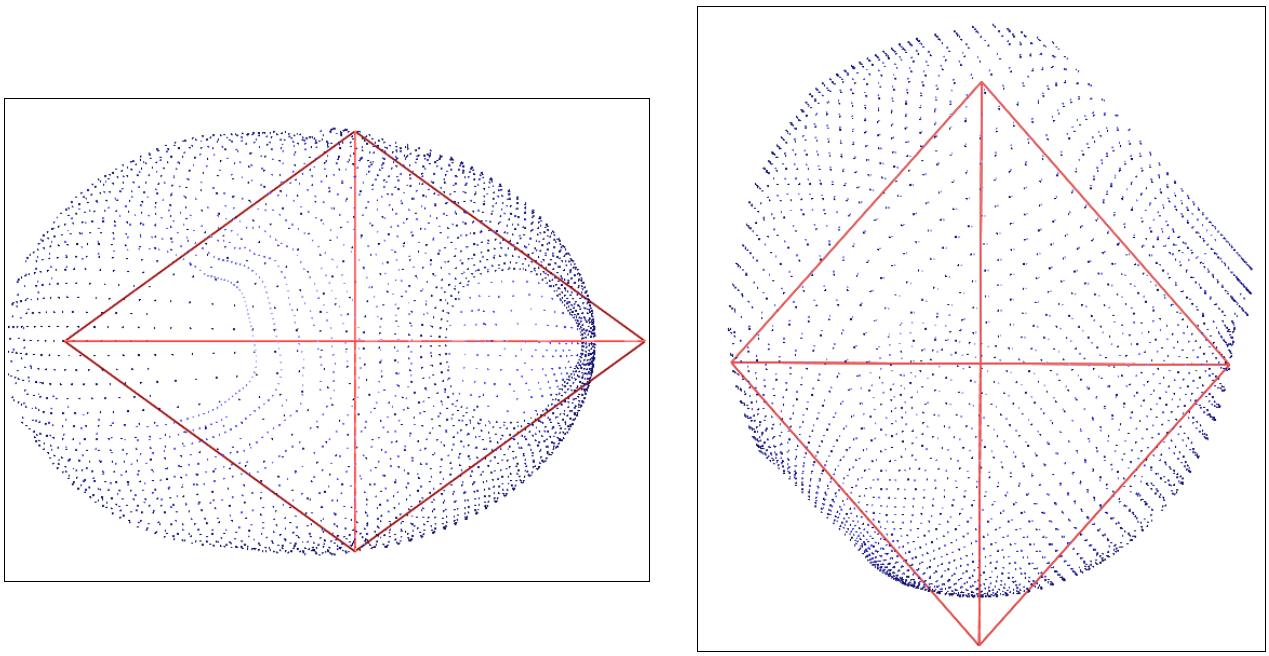


Figura 5.5.1. Alineación del modelo de cabeza y obtención de su elipsoide equivalente, mediante el programa Visor3D. Izquierda: Modelo cabeza00.pi alineado junto con su elipsoide equivalente nivel 1. Derecha: Perfil lateral. Proyecciones ortogonales y normales planas.

☞ ***Desde consola, en la carpeta de programas se ejecuta al shell e25.sh***
./e25.sh 00 05

Elipsoide equivalente	Valor de iteración actual [mm]	Valor de iteración previa [mm]
A	***108.2208235867***	109.8367643760
B	***78.6331227497***	80.2783416933
C	***95.8336514619***	97.5761119669

Tabla 5.5.1. Obtención de elipsoide equivalente nivel 25.

☞ ***Desde consola, en la carpeta de programas se ejecuta al shell max_ev.sh***
./max_ev.sh 00 05

Centroide de la elipsoide equivalente trasladada	Valores de iteración actual [mm]	Valores de iteración previa [mm]
X	***-21.8458870517***	-20.9976596937
Y	***0.0***	0.0
Z	***0.0***	0.0

Tabla 5.5.2. Traslación de la elipsoide equivalente.

☞ ***Desde consola, en la carpeta de programas se ejecuta al shell distanceField.sh***
`./distanceField.sh 00 05`

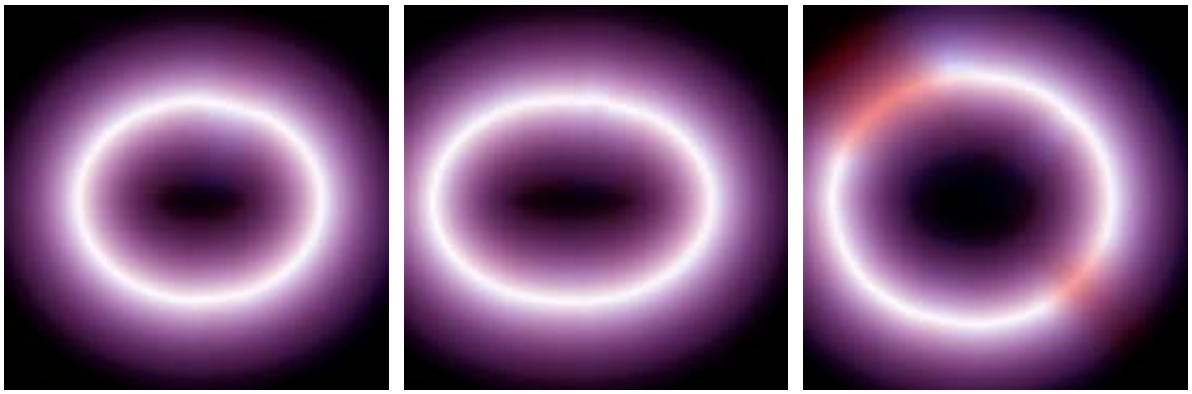


Figura 5.5.2. Muestras del campo de distancia para el criterio de similitud entre los modelos cabeza00_.pi y e00_05R_.pi. Izquierda: Plano YZ, Centro: Plano XY, Derecha: Plano XZ.

$$y_{YZ} = -0.000130452574572563\,x - 0.6332168342$$

$$y_{XY} = -0.0000294129469395813\,x - 0.9673844663$$

$$y_{XZ} = -0.000345978208740085\,x + 3.3479521635$$

	YZ	*XY*	*XZ*
Media [mm]	***-1.1616149875***	***-1.0865216079***	***1.9465674290***
Mediana [mm]	***-1.7078367444***	***-1.9169497133***	***-0.0831706784***
Desviación estándar [mm]	***1.9070424996***	***1.9236708942***	***6.8699247366***
Varianza [mm²]	***3.6368110955***	***3.7005097093***	***47.1958658865***

Valores estadísticos en la quinta iteración del modelo cabeza00.pi.

Muestra del campo de distancia	Error cuadrático [mm²]	Error [mm]	Error [mm] (iteración previa)
YZ	*40384.2630343051*	***200.958361444119***	*211.958108397675*
XY	*39532.7146905560*	***198.82835484547***	*237.838933322053*
XZ	*412931.2283351593*	***642.597252044513***	*673.142782311265*

Tabla 5.5.3. Cantidades de error en la quinta iteración.

✓ ***Se presentó menor error en la quinta iteración.***

☞ ***Desde consola, en la carpeta de programas se ejecuta al shell sup_out.sh***
`./sup_out.sh 00 05`
Backup of cabeza00.pi iteration's files...

☞ ***Desde consola, en la carpeta de programas***

```
./aproximaElipsoideX0Y0Z0ConPlano ./00/cabeza00_.pi
                                  ./00/cabeza00.pi
                                  -21.8458870517 0.0 0.0
                                  -21.8458870517 -1.0 0.0
                                  -21.8458870517 0.0 1.0
                                  108.2208235867 78.6331227497 95.8336514619
                                  -21.8458870517 0.0 0.0
```

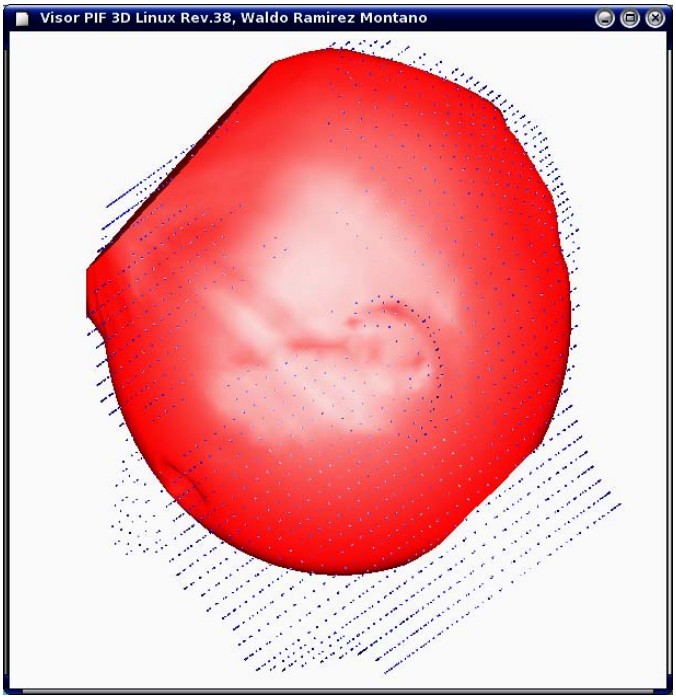


Figura 5.5.3. Comparación del modelo de cabeza resultado de la supresión de outliers y el modelo de cabeza original alineado. Proyección ortogonal y normales Gouraud.

☞ ***Para finalizar iteración, en la carpeta de programas se ejecuta al shell end.sh***

```
./end.sh 00 05
```

➔➔➔ ***Respaldo del resto de archivos resultantes de la quinta iteración***

Backup of cabeza00_.pi iteration's files...

✓ ***A continuación se presentan los resultados de la sexta iteración.***

Para este modelo, la sexta iteración fue el inicio de la convergencia del proceso, ya que dos sumas de diferencias cuadráticas ($E^2_{YZ.}, E^2_{XY.}$) presentaron incremento en lugar de decremento (como se detalla en la tabla 5.6.3). A continuación se resumen los resultados de la sexta iteración con el fin de apreciar la convergencia (y por ende, fin del proceso iterativo).

☞ ***Desde consola, en la carpeta de programas se ejecutan los shells***

```
./abc.sh 00
./e25.sh 00 06
./max_ev.sh 00 06
./distanceField.sh 00 06
```

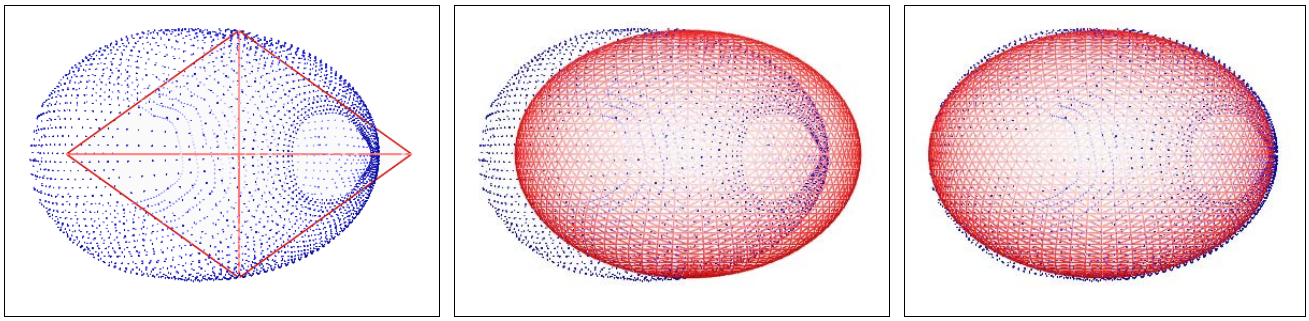


Figura 5.6.1. Alineación del modelo de cabeza y su elipsoide equivalente. Izquierda: Modelo de cabeza alineado junto con su elipsoide equivalente nivel 1. Centro: Modelo de cabeza alineado junto con su elipsoide equivalente nivel 25. Derecha: Traslación del modelo de elipsoide equivalente nivel 25. Proyecciones ortogonales y normales planas.

Elipsoide equivalente	Valor de iteración actual [mm]	Valor de iteración previa [mm]
A	***107.0617011006***	108.2208235867
B	***77.1799887473***	78.6331227497
C	***94.2737745964***	95.8336514619
Tabla 5.6.1. Obtención de elipsoide equivalente nivel 25.		

Centroide de la elipsoide equivalente trasladada	Valores de iteración actual [mm]	Valores de iteración previa [mm]
X	***-22.4353921399***	-21.8458870517
Y	***0.0***	0.0
Z	***0.0***	0.0
Tabla 5.6.2. Traslación de la elipsoide equivalente.		

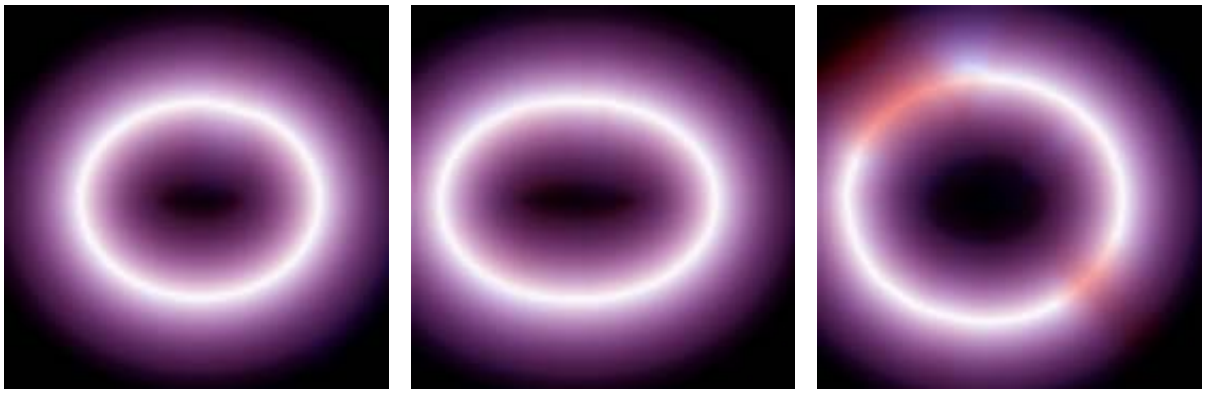


Figura 5.6.2. Muestras del campo de distancia para el criterio de similitud entre

los modelos cabeza00_.pi y e00_06R_.pi. Izquierda: Plano YZ, Centro: Plano XY, Derecha: Plano XZ.

$$y_{YZ} = -0.000189499117542282\,x - 0.4534365572$$

$$y_{XY} = -0.0000161142806225244\,x - 0.9633514208$$

$$y_{XZ} = -0.000363736324403195\,x + 3.4911230362$$

	YZ	*XY*	*XZ*
Media [mm]	***-1.2210027328***	***-1.0286223145***	***2.0178090542***
Mediana [mm]	***-1.5124662126***	***-1.7726025403***	***-0.2160852271***
Desviación estándar [mm]	***1.9583561471***	***1.9801842614***	***6.6022607108***
Varianza [mm²]	***3.8351587989***	***3.9211297091***	***43.5898464939***

Valores estadísticos en la sexta iteración del modelo cabeza00.pi.

Muestra del campo de distancia	**Error cuadrático [mm²]**	**Error [mm]**	**Error [mm] (iteración previa)**
YZ	*43136.8172682242*	***207.694047262371***	*200.958361444119*
XY	*40327.5468272316*	***200.81719753853***	*198.82835484547*
XZ	*386013.749126130*	***621.30004758259***	*642.597252044513*

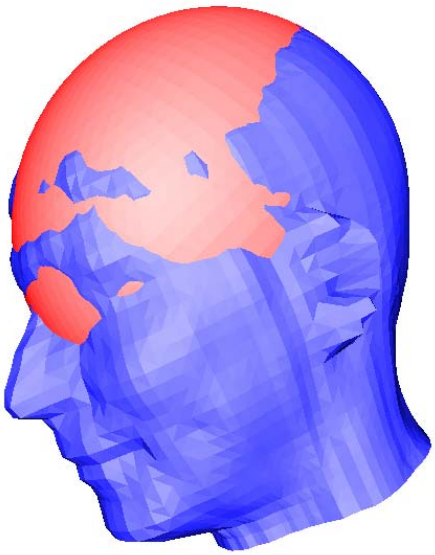
Tabla 5.6.3. Cantidades de error en la sexta iteración.

x No se presentó menor error en todas las muestras de la sexta iteración.

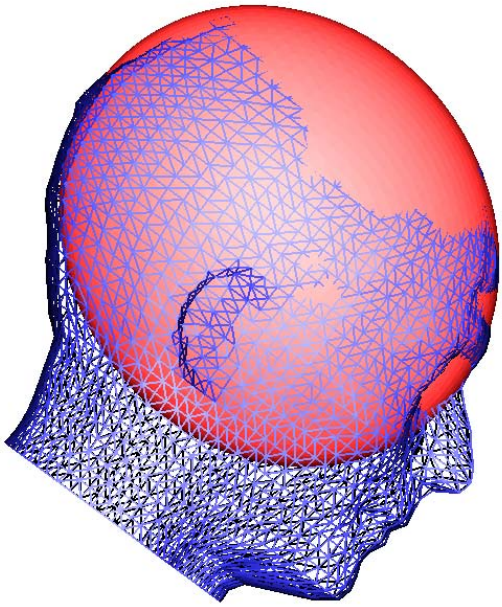
Se llega al fin del proceso iterativo al obtener mayor error en dos de las tres muestras del campo de distancia (es decir, en los planos YZ y XY). La sexta iteración es el mejor resultado de éste proceso de ajuste robusto para el modelo cabeza00.pi, ya que a pesar de presentar la convergencia del proceso iterativo, la suma de los tres valores de error fue menor con respecto a su iteración previa.

A partir del modelo elipsoide de la sexta iteración se puede emplear al visor para aplicar transformaciones geométricas de traslación (sin aplicar cambios morfológicos, sólo cambios de posición), que tienen la finalidad de completar al casquete del modelo de cabeza.

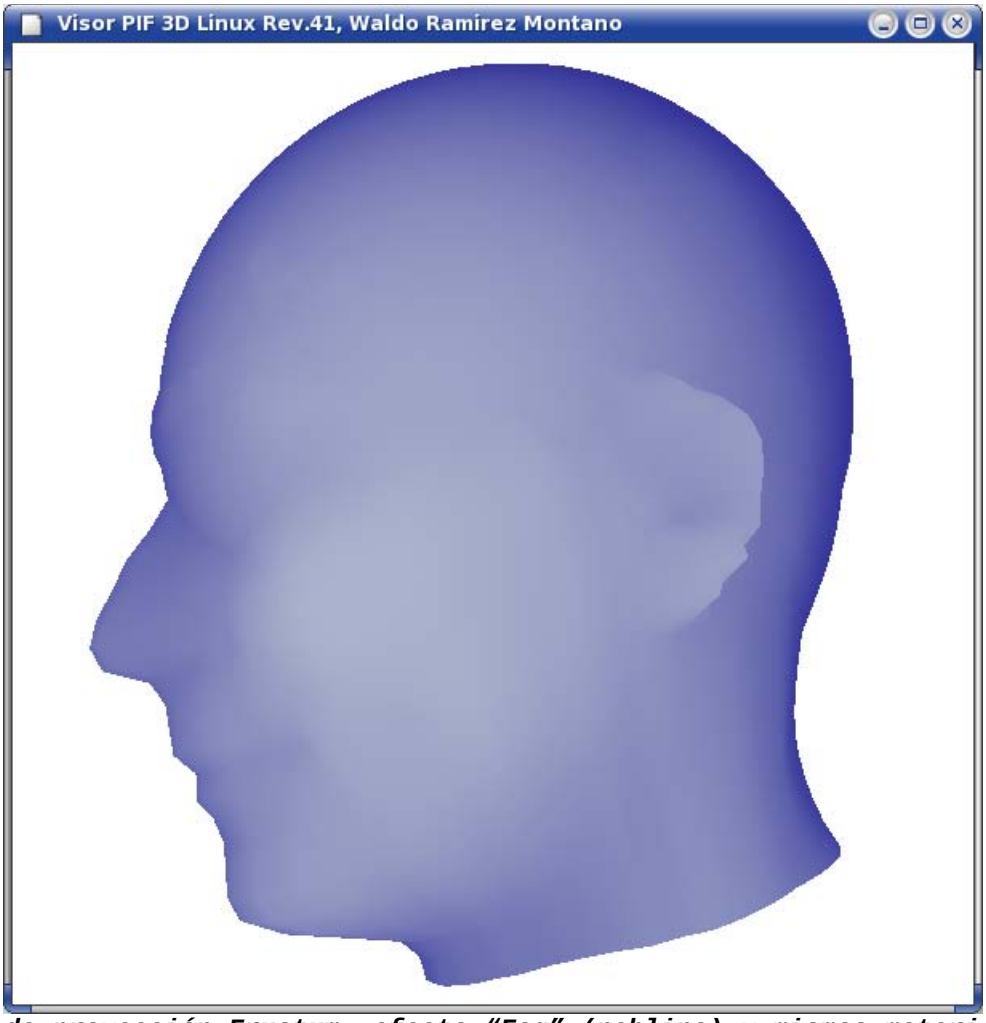
A continuación se muestran ejemplos del ajuste práctico del modelo elipsoide de la sexta iteración (*e00_06R_.pi,* ubicado en la carpeta eNN) del proceso y el modelo de *cabeza00.pi* alineado en la primera iteración (*cabeza00_01A.pi,* ubicado en la carpeta cNN):



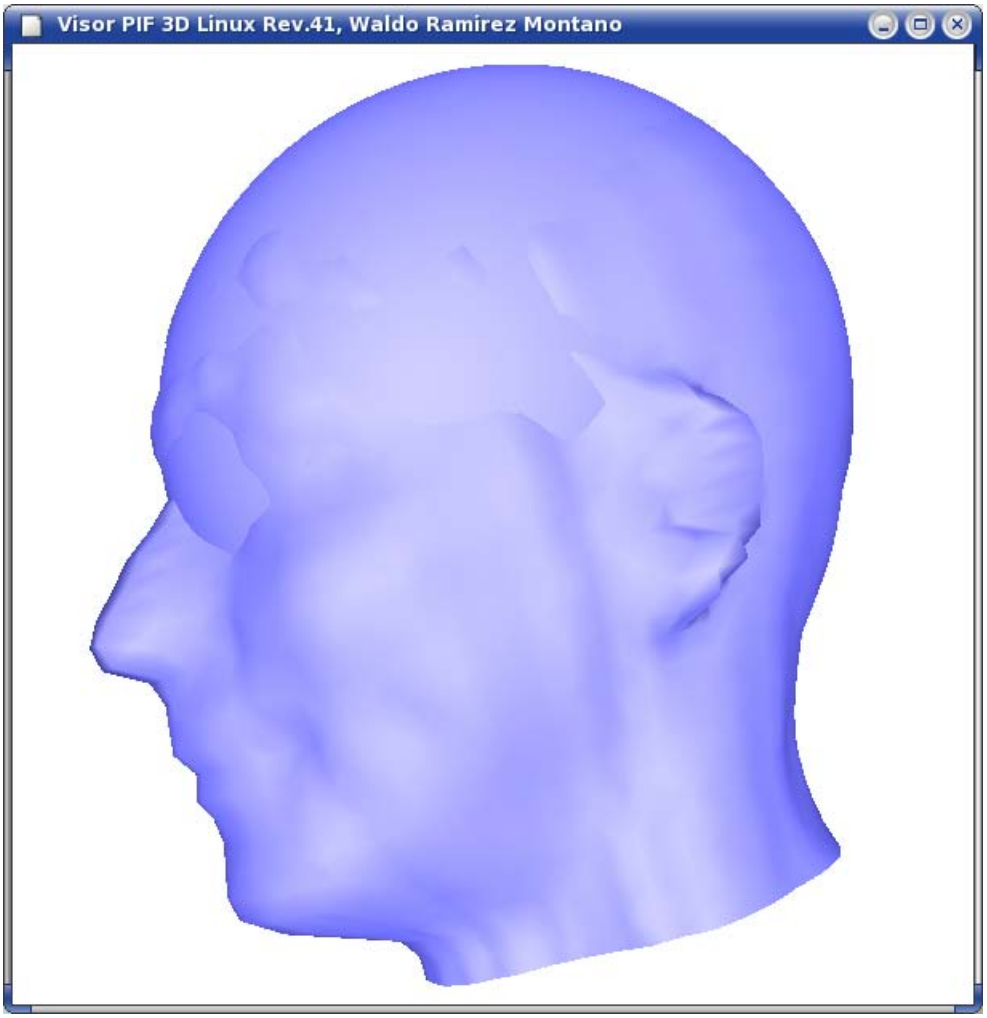
Uso de proyección Frustum



Uso de proyección Ortogonal



Uso de proyección Frustum, efecto "Fog" (neblina) y mismos materiales



Uso de proyección Frustum, normales Gouraud y mismos materiales

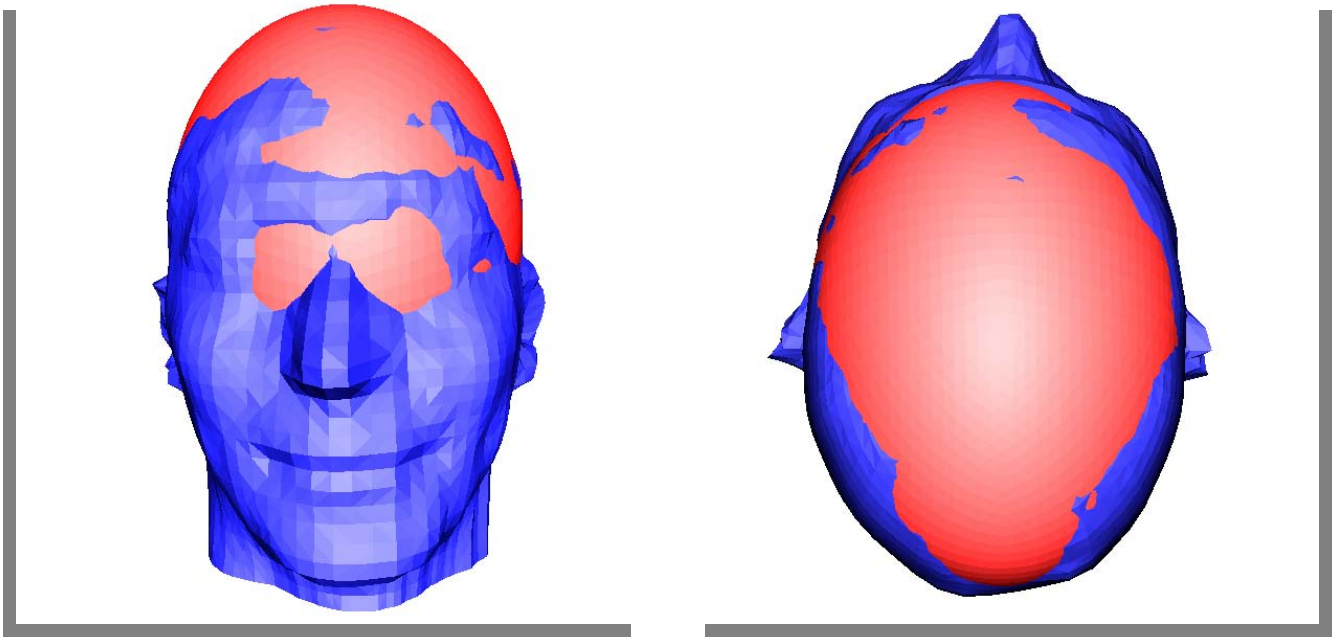


Figura 5.7. Aproximación de completar al casquete con el modelo elipsoide.

5.3 Resumen de resultados y conclusiones

La propuesta del ajuste robusto provocó resultados satisfactorios que cubren a los objetivos mencionados en el Capítulo I, sección 1.4.

- La compatibilidad de los programas, con diversas distribuciones del sistema operativo Linux, fue satisfactoria ya que no se presentaron problemas de funcionamiento ni la necesidad de su recompilación. Incluso la compatibilidad se extendió a distribuciones *Linux LiveCD*.
- La alineación de los modelos de cabeza destacó a los puntos alejados, en especial a los puntos de la nariz, mentón y orejas.
- Para todo modelo de cabeza, su elipsoide representativa corresponde a la elipsoide equivalente de la iteración de mejor ajuste. Con ello se demostró que la traslación de los puntos alejados, mejora a la elipsoide representativa, con justificación de la medición de error con los campos de distancia.
- Tomando en cuenta a la elipsoide equivalente de mejor ajuste en cada modelo de cabeza, fue posible generar a la elipsoide representativa de la población de modelos, mediante un promedio estándar. Presenta los valores siguientes:

Eje de elipsoide	Magnitud [mm]
a	*103.8309982335*
b	*79.8284938045*
c	*87.2770342150*

Tabla 5.7.1. Elipsoide representativa de la población de modelos de cabeza.

- Como modelo adicional, a partir de la elipsoide representativa de la población, la esfera representativa de la población (con fines ilustrativos) consiste en el promedio de los valores a, b, c, de la tabla 5.7.1,

Radio	Magnitud [mm]
r	*90.3121754177*

Tabla 5.7.2. Elipsoide representativa de la población de modelos de cabeza.

- El ajuste del casquete de los modelos de cabeza fue posible, aunque mediante un método práctico de traslaciones. La ventaja fundamental es que puede ser obtenida una representación analítica de la elipsoide (su ecuación), ya que mediante sus ejes (a, b, c) son previamente conocidos y mediante el Visor3D se puede obtener a su centroide geométrico (el cual representa al historial de sus traslaciones geométricas).

A continuación se presentan las tablas de resultados correspondientes a la población de 35[18] modelos de cabeza; posteriormente se destacan características de los datos resultantes y se especifican las conclusiones y propuestas de mejora.

18 En las tablas de resultados, la numeración tiene la ausencia de los modelos 02, 14, 25, 32, 36 y 37; se debe a su inexistencia en el proyecto. La numeración se mantiene con fines de referencia para proyectos relacionados.

Elipsoides representativas			
Cabeza	***a* [mm]**	***b* [mm]**	***c* [mm]**
00	107.0617011006	77.1799887473	94.2737745964
01	106.3288189544	70.8455131848	83.2662428638
03	106.3166072534	86.2165754192	95.8960795910
04	109.2656579666	80.6830476068	92.7146241819
05	96.7252938317	78.8243141137	90.3402681113
06	100.7829190678	85.0975138963	91.8596814239
07	98.9387831699	84.7146870409	92.2975847696
08	99.4888511433	83.3592522749	93.2666750009
09	105.4924812392	75.9205158499	80.0948366190
10	104.2251261400	80.6983623137	89.6199700553
11	98.7459901176	84.9074158053	91.6498859466
12	96.3245138288	80.7164409661	92.8676160130
13	100.5641320755	74.2564499948	91.2790433461
15	109.4633313589	76.0158006772	79.7402691842
16	99.9773003609	74.4896927121	84.8629748862
17	100.9145975956	83.5639387824	90.8254802134
18	107.2438146624	82.9300232383	85.2303468732
19	102.4734935456	77.8002066507	87.4104164085
20	116.5649171421	76.3069211301	77.7248357588
21	107.2100199283	73.9776451470	90.2865655287
22	106.1210943447	76.5177003740	77.4567851824
23	107.2096661660	73.7916842913	78.0941213717
24	104.0612416426	87.2384070049	92.9926339939
26	92.9667925325	78.0143188705	79.4542239243
27	94.8320988241	77.0028763012	86.5063736542
28	105.1925903120	78.5040235816	83.0803976950
29	107.0138472151	76.3593833205	81.4865748137
30	108.1401473972	74.8086933319	79.6835224395
31	108.7681044910	81.6825187130	90.7414536611
33	100.5091371193	82.3567965933	84.4061461748
34	105.4155063133	84.4165379948	89.3965732916
35	97.5481875084	80.9627845559	87.2306108327
38	110.7462535139	86.0299459506	88.4644546406
39	106.1768912837	85.0457983882	91.3695576069
40	105.2750290275	82.7615083342	88.8255968704

Tabla 5.7.3. Ejes de cada elipsoide representativa resultante.

Centroide de cada Elipsoide representativa			
Cabeza	***cgx* [mm]**	***cgy* [mm]**	***cgz* [mm]**
00	-22.4353921399	0.0000000000	0.0000000000
01	-25.6879036320	0.0000000000	10.4500755535
03	-8.8230641692	0.0000000000	-8.0658736707
04	-6.5009321756	0.0000000000	2.0874343969
05	-21.5878222625	0.0000000000	25.0190191244
06	-25.8239459033	-2.2183389175	0.0000000000
07	-27.6977125984	0.0000000000	0.0000000000
08	-16.4844379165	0.9046121675	0.0000000000
09	-23.3915262459	0.0000000000	18.6948382630
10	-4.7936194005	0.0000000000	32.7468754950
11	-27.6768451086	-2.0000000000	0.0000000000
12	-22.3152621706	0.0000000000	8.3050014952
13	-21.3552618015	0.0000000000	6.3407162997
15	-18.7997664118	0.0000000000	10.0392442045
16	-21.6207992000	0.0000000000	19.0979900726
17	-18.5694157015	-4.0431842822	15.0741441378
18	-18.7468291719	5.8713505505	18.5823036194
19	-11.4856488229	9.8567149087	16.8166330378
20	-24.0869824386	17.3364550174	0.0000000000
21	-12.8471152449	4.6934695950	0.0000000000
22	-17.0658738539	0.0000000000	12.4920982287
23	-9.9158220562	0.0000000000	8.2542814263
24	-20.7208971515	-7.1835562786	16.9757756346
26	-17.6824789585	0.0000000000	19.1983319107
27	-9.0170278837	0.0000000000	20.5921965240
28	-14.0603250018	10.5657629351	0.0000000000
29	-13.8438213872	-15.9107211907	0.0000000000
30	-12.8116397385	15.6436162487	0.0000000000
31	-16.7340445500	0.0000000000	6.6683778206
33	-22.5781436465	0.0000000000	22.4182767620
34	-5.3388586492	-9.1535745789	25.5372648135
35	-9.5805270575	12.4874751706	17.7622182993
38	1.7099273479	-4.1897522532	26.5524495934
39	-10.9760198308	0.0000000000	22.8815111743
40	-3.7949265741	-4.3596343669	23.5717891240

Tabla 5.7.4. Centroide de cada elipsoide representativa resultante.

Decremento de error en la última iteración		
Cabeza	***# de la iteración de convergencia***	***Error* [mm]**
00	6	-12.5726759506
01	4	-68.6079378980
03	5	-12.3620367797
04	10	-15.2705134833
05	3	-495.5514409914
06	3	-454.6591184512
07	4	-195.0106092576
08	5	-151.0368345704
09	3	-120.3626885572
10	3	-521.1920658625
11	4	-132.7463863695
12	5	-100.6631612106
13	3	-152.8232225896
15	6	-20.0446508121
16	4	-75.3007890667
17	4	-149.7282820255
18	4	-81.2679727479
19	5	-59.3502613039
20	4	-32.4389813267
21	10	-23.7567808122
22	6	-3.1318495570
23	5	-31.8005389672
24	2	-1186.4241983892
26	4	-56.8766870474
27	3	-143.9044795368
28	7	-27.3448314401
29	4	-169.8651140912
30	4	-57.4559066386
31	3	-79.7712138408
33	4	-275.6743971037
34	3	-129.5450084174
35	4	-1.0392096081
38	2	-732.1783545631
39	3	-255.0590724643
40	3	-196.9497647112

Tabla 5.7.5. Decremento de error obtenido en la última iteración.

Rectas de regresión lineal en la última iteración						
Cabeza	***m (YZ)***	***b (YZ)***	***m (XY)***	***b (XY)***	***m (XZ)***	***b (XZ)***
00	-0.0001894991	-0.4534365572	-0.0000161143	-0.9633514208	-0.0003637363	3.4911230362
01	-0.0006604324	0.4037827173	-0.0000775353	-2.0707093473	-0.0016767074	8.1715998939
03	0.0002780076	-1.2886287586	-0.0000519277	0.1019520498	0.0004117819	-0.2904490473
04	-0.0000209436	-0.4620329231	0.0000339753	-0.2755186765	0.0000372723	1.1694882959
05	0.0008690409	-6.9212895283	0.0011365426	-8.1578164576	-0.0028353917	8.3649202241
06	-0.0002166018	-0.0146391850	-0.0001133371	-1.6271280000	-0.0007726019	4.3284327954
07	0.0002548385	-2.3105038140	0.0001788821	-2.6912241892	-0.0006505694	4.4076365496
08	0.0002876682	-1.8312428906	0.0001614894	-1.2662912425	-0.0001232510	1.5422241356
09	0.0001935833	-4.3625915583	0.0000248797	-1.9982882481	-0.0032215592	14.3571613496
10	0.0000661798	-4.1402159427	-0.0004678619	-0.7949390733	-0.0030141642	10.7351666209
11	-0.0001765761	-1.6534467567	0.0000599631	-1.9637913135	-0.0004902878	3.4560812751
12	-0.0001588981	-0.9499255758	-0.0001156779	-1.4311563314	-0.0014451929	6.5575255195
13	0.0000616309	-1.9054432737	-0.0000305698	-1.6708918189	-0.0015547961	6.9775576811
15	-0.0002311394	0.2229642945	0.0001256082	-2.3510777457	-0.0009359331	5.8139090483
16	0.0004348861	-3.3733	0.0004488901	-3.9696	-0.0019001570	8.2301
17	-0.0001800385	-0.568	0.0001345007	-1.9155	-0.0019714380	8.57957
18	-0.0007526136	1.07648	-0.0009574073	2.56016	-0.0023703484	10.4912
19	-0.0002862418	-0.8579	0.0002136267	-3.0648	-0.0015837900	7.1072
20	-0.0017289251	5.94243	-0.0026934048	11.035	-0.0005334092	1.72326
21	0.0007756052	-0.9120939637	0.0006194655	1.7525837452	0.0001951511	-0.6225857163
22	0.0002449454	-1.7319428178	0.0003099458	-3.1625225075	-0.0014273703	8.5543955419
23	-0.0002184685	-0.2475	-0.0003480163	-0.5443	-0.0009303622	5.73783
24	0.0001073839	-1.4567	-0.0000436319	-2.0337	-0.0023600835	9.78547
26	-0.0000056482	-0.8255	-0.0001639021	-1.7267	-0.0009475719	1.54409
27	0.0003710375	-2.4531	0.0006776703	-4.5648	-0.0023931507	9.91558
28	0.0001123044	0.5302	-0.0002694825	2.97088	-0.0004315040	0.83694
29	0.0012263421	-5.7445905342	0.0016488837	-6.9162430085	-0.0001322982	-0.5827690451
30	-0.0007184540	2.4569174442	-0.0015684075	8.5980229358	-0.0000867644	-0.7569107717
31	-0.0008750636	2.52519	-0.0009596883	2.62101	-0.0008760318	5.43596
33	0.0005007100	-3.9917836582	0.0004499237	-3.2396340963	-0.0018513357	6.1423591642
34	0.0006737122	-5.1209	0.0003398033	-3.6193	-0.0026023113	11.619
35	-0.0004359963	0.39971	-0.0003340161	-0.5337	-0.0022192154	9.70827
38	0.0003707517	-1.9289	0.0000733196	-1.4817	-0.0015594354	3.91658
39	-0.0005356418	1.17935	-0.0010937753	3.3732	-0.0023243776	8.98726
40	0.0002624813	-2.8482	0.0001008722	-2.2231	-0.0021163108	8.76457

Tabla 5.7.6. Pendientes y ordenadas al origen de las rectas de regresión lineal para cada modelo en los tres planos analizados.

Se observa que la ordenada al origen en el plano XZ tendió a ser el valor de mayor magnitud: es muestra de la presencia de los cortes en el cuello y el casquete de los modelos.

Al apreciar las tablas de resultados, el análisis de resultados refleja el comportamiento homogéneo del proceso iterativo, como se ilustra en la tabla siguiente,

	Mínimo	Máximo	Desv. Estándar.	Media
a [mm]	92.9667925325	116.5649171421	*5.0642167527*	***103.8309982335***
b [mm]	70.8455131848	87.2384070049	*4.3534995049*	***79.8284938045***
c [mm]	77.4567851824	95.8960795910	*5.4039145151*	***87.2770342150***
cgx [mm]	-27.6977125984	1.7099273479	*7.4618549605*	***-16.0897360431***
cgy [mm]	-15.9107211907	17.3364550174	*6.4049840778*	***0.8085912779***
cgz [mm]	-8.0658736707	32.7468754950	*10.4774382234*	***11.3740849526***
Iteración [#]	2	10	*1.8140441843*	***4.3428571429***
|Error| [mm]	1.0392096081	1186.4241983892	*242.1947410012*	***-177.7647724698***

Tabla 5.8.1. Resumen del análisis de resultados I.

Se aprecia que la desviación estándar de los valores propios de las elipsoides representativas (sus ejes y centroides geométricos) no supera los 10.5 [mm] de magnitud; de hecho, el único valor que se aproxima a los 10.5 [mm] es la coordenada en Z del centroide geométrico (el resto de los valores mencionados no pasa de 7.5 [mm]).

En cuanto a la cantidad de iteraciones de cada modelo en la población, el valor máximo y el valor mínimo aparentan una amplia variación, aunque la media y desviación estándar correspondientes, aclaran que los valores extremos pertenecen a muy pocos modelos (de hecho, el valor mínimo es de 2 modelos y el máximo es de 2 modelos).
El decremento del error establece un comportamiento similar al citado en la cantidad de iteraciones: a pesar de existir extremos de amplia diferencia, no simboliza a un comportamiento inestable. Lo que indica es que en las iteraciones iniciales de cada modelo, el decremento de error es amplio, y tiende a disminuir en las iteraciones siguientes (en caso de ser necesarias).

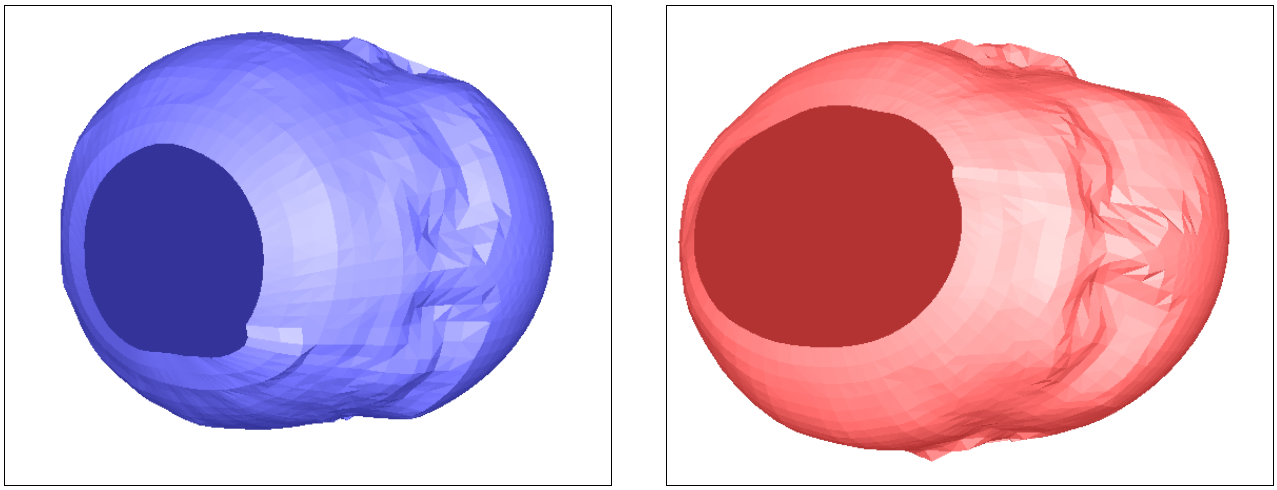


Figura 5.8. Estado físico de los modelos con menor decremento de error (izquierda, cabeza34_04A.pi) y mayor decremento de error (derecha, cabeza24_02A.pi) en su última iteración. Proyecciones ortogonales y normales planas.

Como se detalla en la sección *5.2 (Ejemplo del proceso iterativo),* el uso de medidas estadísticas apoya a la interpretación del ajuste robusto del modelo; sin embargo, como se especificó en el Capítulo IV *(4.7 Estimación de errores morfológicos),* proponemos una interpretación particular de los campos de distancia para la medida de error: la regresión lineal de las diferencias cuadráticas de distancias. En el punto 5.2 se especificaron a las medidas estadísticas con el fin de compararles con la propuesta de regresión lineal: la pendiente y ordenada al origen de la recta de regresión lineal también representan al comportamiento de la diferencia (cabeza – elipsoide) entre cada muestra de los campos de distancia (ya sea el plano XY, YZ, ó XZ). La ordenada al origen simboliza el *grado de dispersión* de las diferencias, mientras que la pendiente simboliza el *grado de variación* de las diferencias.

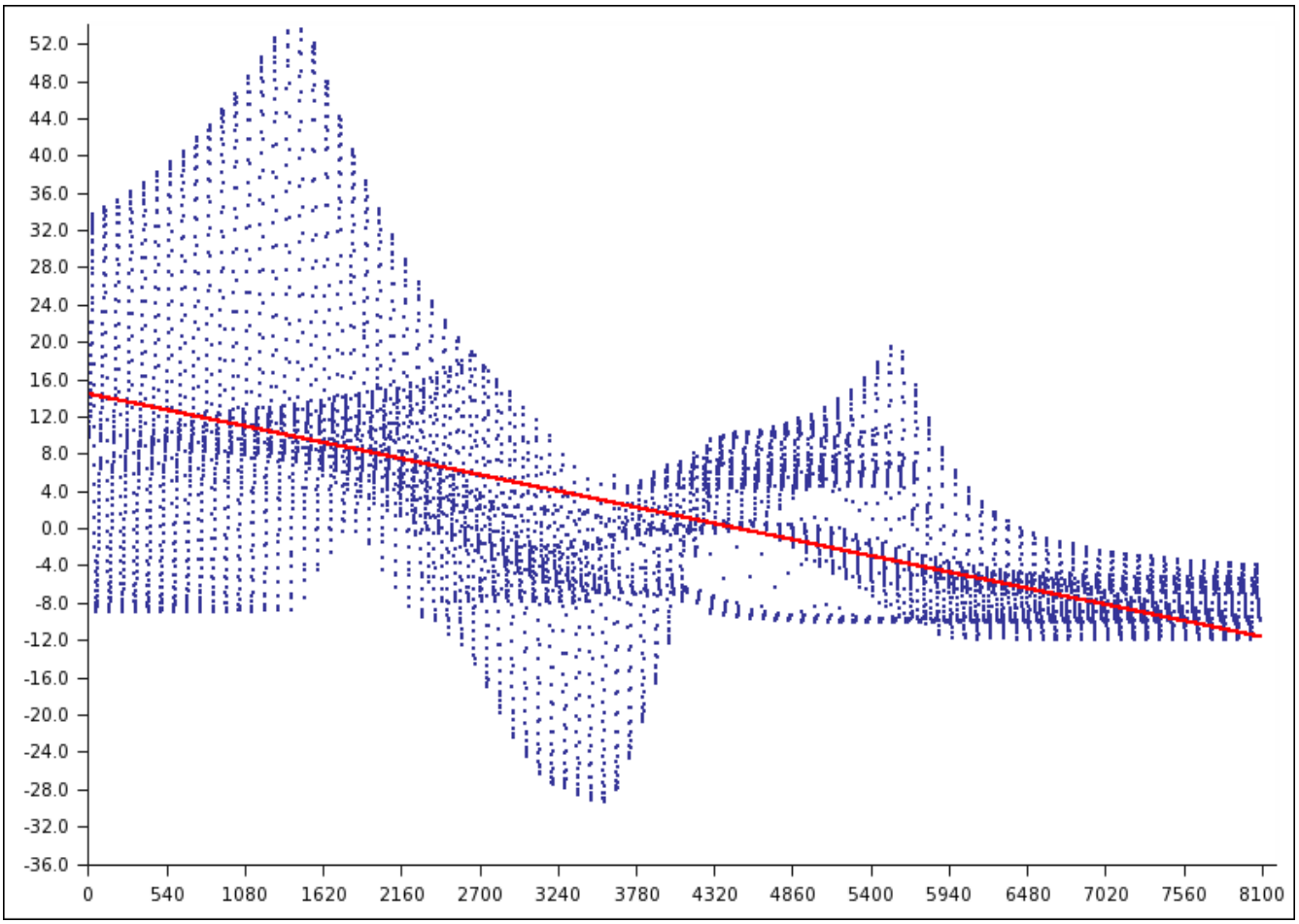


Figura 5.9. El modelo cabeza09.pi presentó la recta de regresión lineal con el mayor valor absoluto de dispersión de diferencias (su ordenada al origen), en su tercera iteración (en el plano XZ). El grado de variación (su pendiente) indica disminución de la magnitud de diferencias (como se aprecia en la gráfica de numeración de distancias).

De esta manera aprovechamos la noción unidimensional de los campos de distancia y sus diferencias, ya que nos brindó la posibilidad de controlar libremente a su representación: el eje X es simplemente la numeración ordenada de las diferencias en el plano correspondiente.

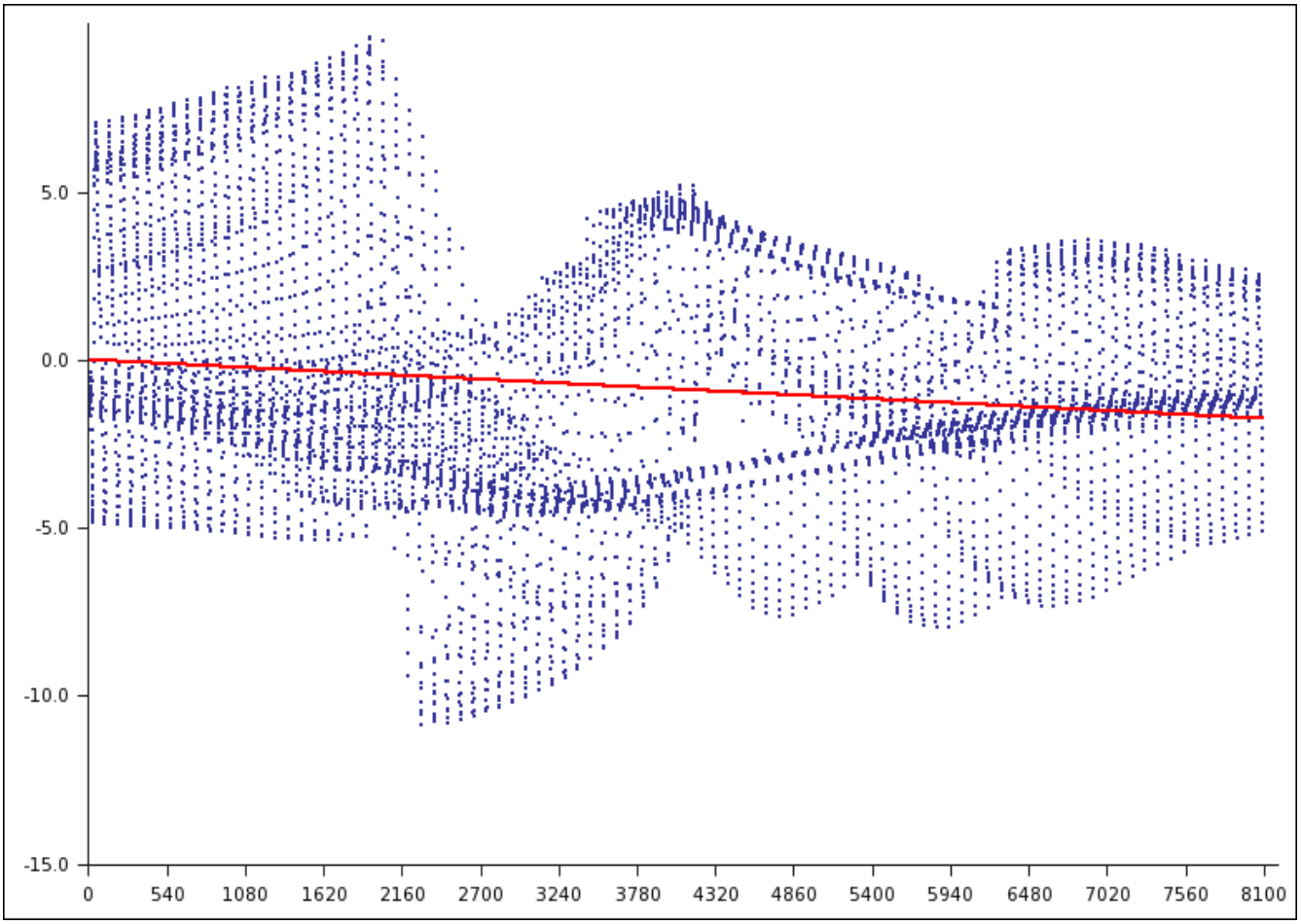
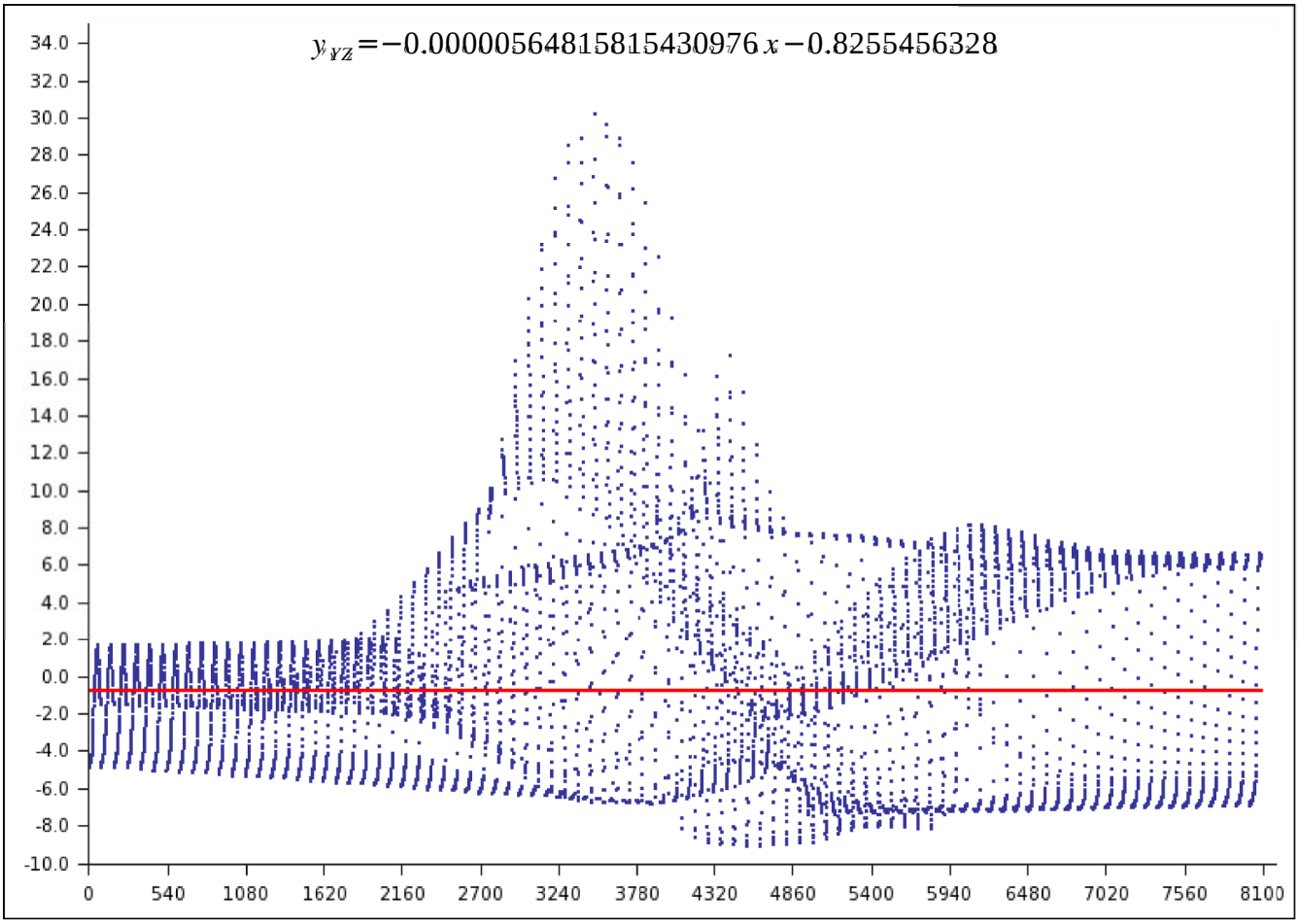


Figura 5.10. El modelo cabeza06.pi presentó la recta de regresión lineal con el menor valor absoluto de dispersión de diferencias (su ordenada al origen), en su tercera iteración (en el plano YZ). El grado de variación (su pendiente) indica disminución de la magnitud de diferencias (como se aprecia en la gráfica de numeración de distancias).

	|Mínimo|	|Máximo|	Desv. Estándar	Media
m (YZ)	0.0000056482	0.0017289251	*0.0005572144*	***-0.0000085735***
b (YZ)	0.0146391850	6.9212895283	*2.4901779776*	***-1.2461958200***
m (XY)	0.0000161143	0.0026934048	*0.0007431390*	***-0.0000733290***
b (XY)	0.1019520498	11.0349709779	*3.6337369882*	***-0.9498604655***
m (XZ)	0.0000372723	0.0032215592	*0.0009921505*	***-0.0013444929***
b (XZ)	0.2904490473	14.3571613496	*3.9752272019*	***5.8342763097***

Tabla 5.8.2. Resumen del análisis de resultados II.
Nota: Todo valor máximo o mínimo en esta tabla, hace referencia al valor absoluto correspondiente.

Cabe señalar que un límite de las rectas de regresión lineal es su naturaleza de primer grado, ya que no refleja nítidamente a comportamientos de orden superior; por ejemplo, como se aprecia en la fig.5.11, la recta de regresión lineal indica una dispersión y variación



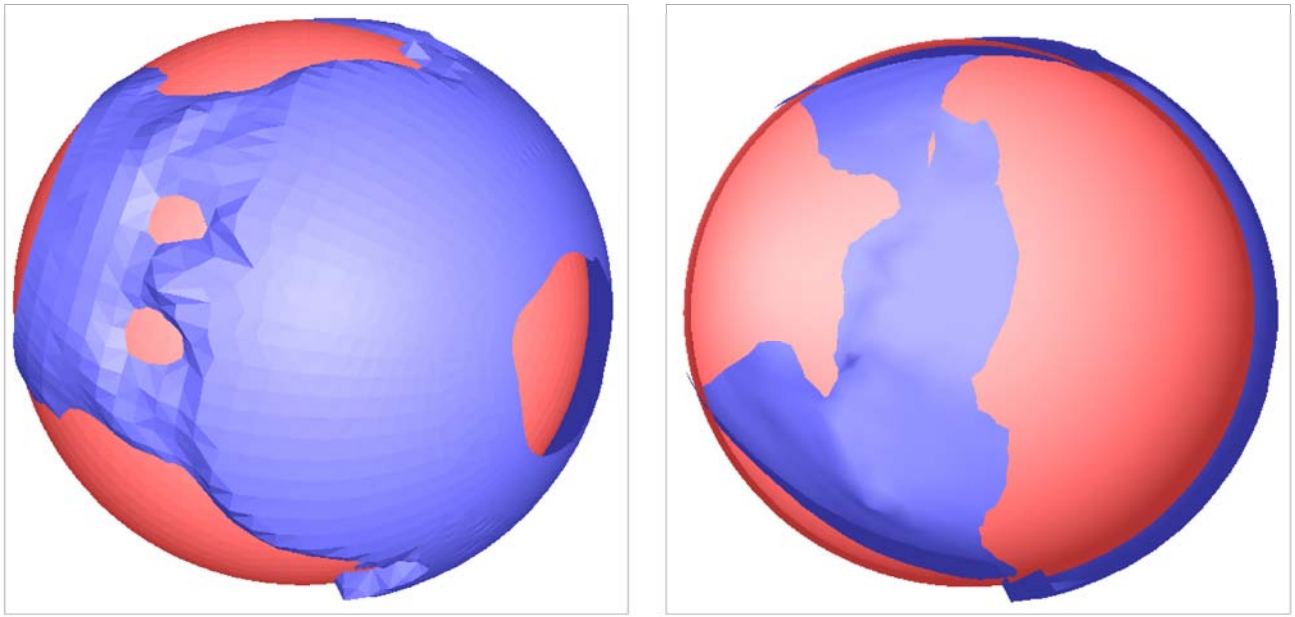
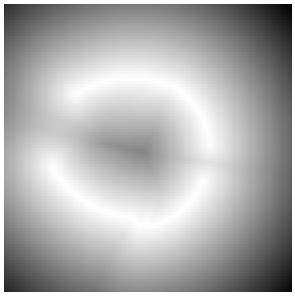


Figura 5.11. Modelo cabeza26.pi en su cuarta iteración, plano YZ.
Superior: Gráfica de numeración de distancias y su recta de regresión lineal.
Inferior izquierda: Despliegue de los modelos en el plano YZ.
Inferior derecha: Corte del plano YZ, con X=0.



tenue, aunque la presencia de los cortes en el casquete y cuello (para el plano YZ, X=0, como se aprecia en la muestra del campo de distancias), causan la dispersión que se aprecia en la gráfica de numeración (entre las distancias 2700 y 4860).

Para poder reflejar a dicho cambio se requeriría una regresión de una curva de mayor grado (por ejemplo, una parábola), lo cual no fue necesario para nuestro análisis.

Finalmente, a continuación se destacan a puntos de conclusiones y propuestas de mejora:

- El esquema del algoritmo es puramente lineal. Proporciona facilidad de implementación, y limites de primer grado.
- Una propuesta de mejora en el uso del tensor de inercia es considerar masa variable para cada punto de los modelos.
  - Todo punto del modelo de cabeza cuenta con masa estática (ya que su distribución es homogénea).
  - Ante el ajuste a los puntos alejados, analizar su cambio de masa: conforme se aproximen (e incrementen su densidad de distribución), asignarles una masa menor.
  - El algoritmo para implementar el cambio de masa, podría basarse en una analogía al decremento de área en los triángulos del modelo, considerados como alejados.
- En el estado actual de los programas del proyecto, la trasformación de puntos alejados consiste únicamente en su traslación hacia la superficie de la elipsoide correspondiente, a lo largo de su radio al origen. Una mejora (con el fin de eliminar al corte del cuello en el modelo de cabeza) puede ser el alterar el trayecto de la traslación del punto alejado; por ejemplo, emplear el radio del punto a un extremo de la elipsoide (a, b, ó c; dependería del ajuste), ó emplear un radio con curva de orden superior.

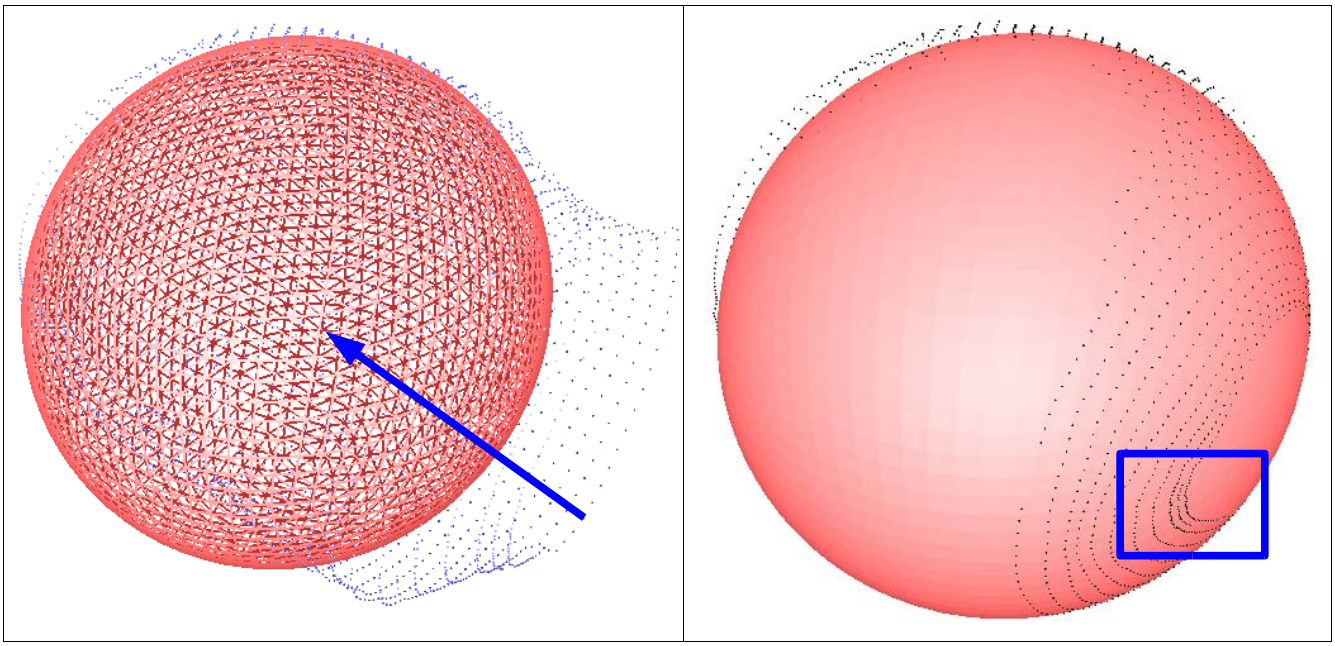


Figura 5.12.1. Ejemplo de la traslación lineal de los puntos alejados. Modelos cabeza09.pi y su elipsoide, primera iteración. Izquierda: Antes de la traslación. Derecha: Después de la traslación.

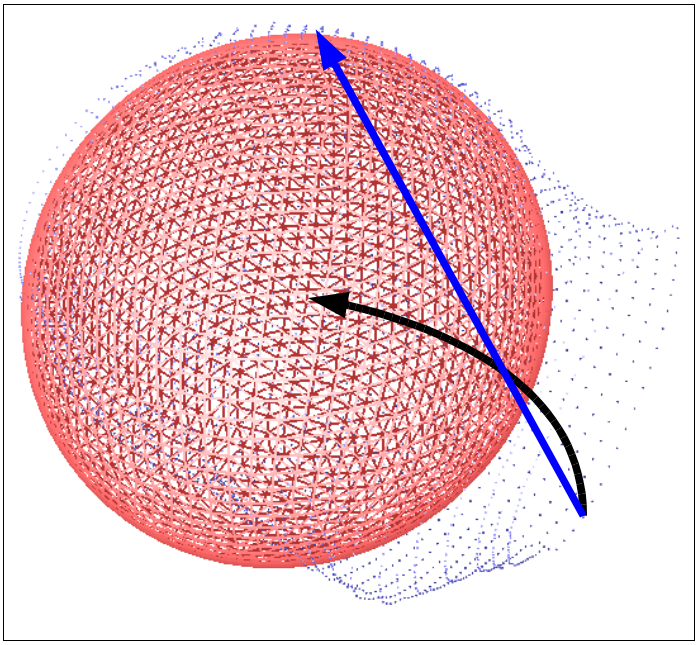


Figura 5.12.2. Ejemplo de propuesta en el ajuste de la traslación de los puntos alejados (outliers).

Recta Azul: Simboliza la traslación del punto alejado a un extremo de la elipsoide.

Curva negra: Simboliza la traslación del punto alejado al origen de la elipsoide.

La propuesta buscaría cerrar el corte en el cuello, para los modelos de cabezas.

- La recta de regresión lineal ideal, contiene a una pendiente y ordenada al origen iguales a cero; equivale a argumentar que todas las diferencias de distancias serían cero. Un ejemplo obvio, es el aplicar el criterio de similitud de un modelo con sigo mismo.
- La influencia de los cortes en los modelos de cabeza, provoca mayor reflejo de error (de acuerdo al criterio de similitud). Las pruebas son (1) mayor ordenada al origen y (2) mayor magnitud en la sumatoria del error. Para reducir a la influencia, convendría buscar el cierre de los cortes en el modelo de cabeza; la estrategia mostrada en la fig.5.12.2 es una propuesta para el corte del cuello. En el caso del corte del casquete, es necesario diseñar un ajuste distinto, ya que no se debe aplicar traslación a los puntos del casquete.
- Dentro del algoritmo de ajuste, se busca trasladar a las elipsoides para hacer coincidir a algunos de sus puntos a, b, ó c con el máximo ó mínimo valor en X, Y, ó Z del modelo de cabeza, con el fin de completar el casquete. Dado que el corte del casquete en los modelos de cabeza tienden a ser elipses, una propuesta de ajuste en el algoritmo es trasladar a la elipsoide buscando el hacer coincidir a un corte de elipse de la elipsoide, con la elipse del corte del casquete.
- Es posible extender la funcionalidad de los programas, ya que el diseño permite emplear a otros modelos tridimensionales que tengan representación analítica. El emplear modelos de otra naturaleza (como parches B-Spline, Bézier), requeriría una extensión en el diseño del trato a los puntos alejados.
- El programa de cálculo de campos de distancia, permite obtener el plano muestra solicitado de cualquier modelo PIF 3D (no presenta restricciones analíticas).
- Los campos de distancia al ser unidimensionales, permiten una amplia variedad de representaciones, lo cual facilita su uso e incorporación; la desventaja que presentan es que su obtención requiere un amplio periodo de tiempo.
- Como mejora para la asimilación del uso de los programas, es viable la implementación de la capa Controlador, como sistema.

APÉNDICES

A. Momentos, productos y tensor de inercia.
B. Ejemplo 2.1, PCA en Matlab.
C. OpenGL - Graphics Pipeline.
D. Diseños y algoritmos.
E. Programas y shell-scripts desarrollados y usados.

La verdad existe. Sólo se inventa la mentira
Georges Braque

Apéndice A. Momentos, productos y tensor de inercia.

Los momentos de segundo orden, tienen gran participación en el análisis matemático de la dinámica del sólido rígido (por ello se les conoce como *momentos de inercia*). Considerando a un sólido discreto con centro de masa en el origen $\boldsymbol{O}$, es posible definir a un vector unitario: el eje de rotación **n**, que pasa por el centro de masa, como se muestra en la siguiente *fig.A.1*,

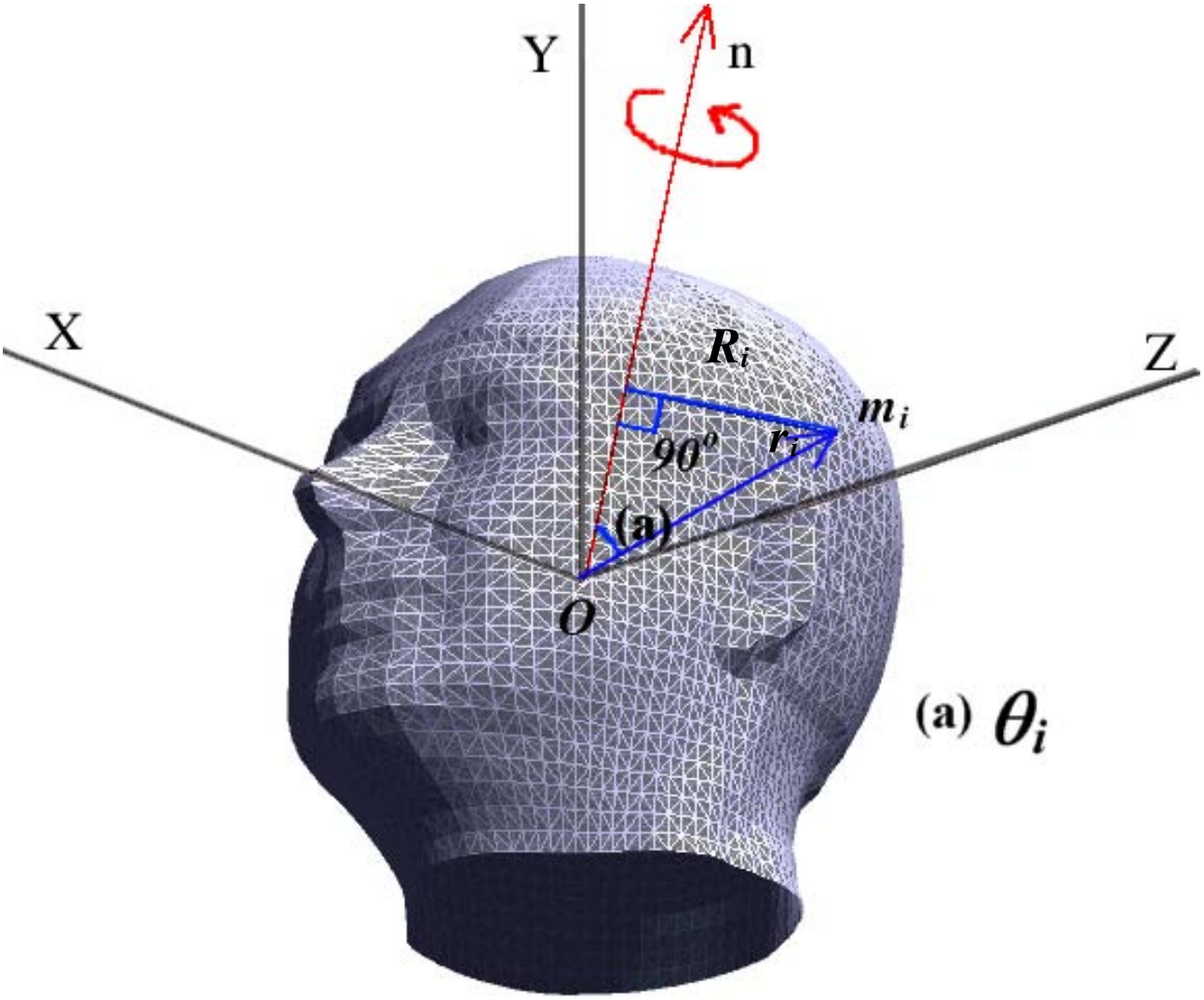


Figura A.1

La ecuación matemática del momento de inercia, es el producto de la distancia normal $\boldsymbol{R_i}$ de cada partícula $\boldsymbol{m_i}$ del sólido discreto al vector **n**. Por ende, para todo el sistema de partículas (todo el sólido discreto), la ecuación representativa del momento de inercia es:

$$I_O = \sum_{i=1}^{n} \boldsymbol{R}_i^2 \, m_i$$

Es posible expresar a $\boldsymbol{R_i}$ como el producto de la hipotenusa $\boldsymbol{r_i}$ con el seno del ángulo θ_i,

$$I_O=\sum_{i=1}^{n}\left(\left|\boldsymbol{r}_i\right|\sin\theta_i\right)^2 m_i \quad \text{, que se puede interpretar como:} \quad I_O=\sum_{i=1}^{n}\left|\boldsymbol{r}_i\times\mathbf{n}\right|^2 m_i$$

donde: $\boldsymbol{r}_i=x_i\mathbf{i}+y_i\mathbf{j}+z_i\mathbf{k}$, y el vector unitario, $\mathbf{n}=\cos\alpha\,\mathbf{i}+\cos\beta\,\mathbf{j}+\cos\gamma\,\mathbf{k}$

Expresando a $\boldsymbol{r_i}$, y a $\mathbf{n}$ como vectores columna, se puede desarrollar al tensor de inercia de la siguiente manera,

$$I_O=\sum_{i=1}^{n}\left|\boldsymbol{r}_i\times\mathbf{n}\right|^2 m_i=\sum_{i=1}^{n}\left(\boldsymbol{r}_i\times\mathbf{n}\right)\cdot\left(\boldsymbol{r}_i\times\mathbf{n}\right) m_i$$

$$I_O=\sum_{i=1}^{n}\left(\begin{bmatrix}0 & -z_i & y_i\\ z_i & 0 & -x_i\\ -y_i & x_i & 0\end{bmatrix}\begin{bmatrix}\cos\alpha\\ \cos\beta\\ \cos\gamma\end{bmatrix}\right)\cdot\left(\begin{bmatrix}0 & -z_i & y_i\\ z_i & 0 & -x_i\\ -y_i & x_i & 0\end{bmatrix}\begin{bmatrix}\cos\alpha\\ \cos\beta\\ \cos\gamma\end{bmatrix}\right) m_i$$

$$I_O=\sum_{i=1}^{n}\begin{bmatrix}y_i\cos\gamma-z_i\cos\beta\\ z_i\cos\alpha-x_i\cos\gamma\\ x_i\cos\beta-y_i\cos\alpha\end{bmatrix}\cdot\begin{bmatrix}y_i\cos\gamma-z_i\cos\beta\\ z_i\cos\alpha-x_i\cos\gamma\\ x_i\cos\beta-y_i\cos\alpha\end{bmatrix} m_i$$

$$I_O=\sum_{i=1}^{n}\left[\left(y_i\cos\gamma-z_i\cos\beta\right)^2+\left(z_i\cos\alpha-x_i\cos\gamma\right)^2+\left(x_i\cos\beta-y_i\cos\alpha\right)^2\right] m_i$$

al desarrollar cada sumando:

$$\left(y_i\cos\gamma-z_i\cos\beta\right)^2=y_i^2\cos^2\gamma-2\left(z_i\cos\beta\right)\left(y_i\cos\gamma\right)+z_i^2\cos^2\beta$$

$$\left(z_i\cos\alpha-x_i\cos\gamma\right)^2=z_i^2\cos^2\alpha-2\left(z_i\cos\alpha\right)\left(x_i\cos\gamma\right)+x_i^2\cos^2\gamma$$

$$\left(x_i\cos\beta-y_i\cos\alpha\right)^2=x_i^2\cos^2\beta-2\left(x_i\cos\beta\right)\left(y_i\cos\alpha\right)+y_i^2\cos^2\alpha$$

Factorizando los cosenos correspondientes, se obtienen los *momentos de inercia*,

$$I_{xx}=\sum_{i=1}^{n}\left(y_i^2+z_i^2\right) m_i \quad I_{yy}=\sum_{i=1}^{n}\left(x_i^2+z_i^2\right) m_i \quad I_{zz}=\sum_{i=1}^{n}\left(x_i^2+y_i^2\right) m_i$$

y los sumandos restantes, son los *productos de inercia*,

$$I_{xy}=I_{yx}=-\sum_{i=1}^{n}x_i y_i\, m_i \quad I_{yz}=I_{zy}=-\sum_{i=1}^{n}y_i z_i\, m_i \quad I_{zx}=I_{xz}=-\sum_{i=1}^{n}x_i z_i\, m_i$$

con lo cual, se obtiene la ecuación representativa del momento de inercia:

$$I_O=I_{xx}\cos^2\alpha+I_{yy}\cos^2\beta+I_{zz}\cos^2\gamma+2I_{xy}\cos\alpha\cos\beta+2I_{yz}\cos\beta\cos\gamma+2I_{zx}\cos\gamma\cos\alpha$$

Existe una forma matricial de expresar a los momentos y productos de inercia: el ***tensor de inercia***.

$$I\equiv\begin{bmatrix}I_{xx} & I_{xy} & I_{xz}\\ I_{yx} & I_{yy} & I_{yz}\\ I_{zx} & I_{zy} & I_{zz}\end{bmatrix}$$

Los elementos diagonales del tensor son los momentos de inercia y el resto de elementos son los productos de inercia. Se observa, que dicho tensor es simétrico, ya que es idéntico a su tensor transpuesto. Cuando el sólido discreto logra estar *alineado*, los productos de inercia son nulos,

$$I \equiv \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

Y los elementos diagonales se vuelven los *momentos principales de inercia*, fundamentales para el cálculo de los *ejes principales* del sólido discreto, empleados en el proyecto presente con el objetivo primordial de generar la *elipsoide equivalente*.

Apéndice B. Script MATLAB 6.5R13 del ejemplo 2.1, PCA.

```
% Limpieza del entorno Matlab
clear all;
close all;
% Genera la nube de puntos aleatoria y normalizada (centroide en el origen)
elip(1,:) = randn(1,30);
elip(2,:) = (4*randn(1,30));
% Rotacion para forzar la desalinecion de la nube de puntos
[r(1,:),r(2,:)] = cart2pol(elip(1,:),elip(2,:));
r(1,:)=r(1,:)-pi/6;
[elipNo(1,:),elipNo(2,:)]=pol2cart(r(1,:),r(2,:))
% Traslacion del centroide fuera del origen: (-2.5, 1.0)
elipNo(1,:) = elipNo(1,:)-2.5;
elipNo(2,:) = elipNo(2,:)+1.0;
% Graficacion de la nube de puntos
scatter(elipNo(1,:),elipNo(2,:));
axis image;
drawnow;
pause;
% Calcula los Componentes Principales (PC) de la nube de puntos
[pc,ev, pvar]=pcacov(cov(elipNo'))
% Grafica los Componentes Principales junto con la nube de puntos
hold on;
plot([-sqrt(ev(1)) sqrt(ev(1))]*pc(1,1),[-sqrt(ev(1)) sqrt(ev(1))]*pc(2,1),'r-');
plot([-sqrt(ev(2)) sqrt(ev(2))]*pc(1,2),[-sqrt(ev(2)) sqrt(ev(2))]*pc(2,2),'g-');
pause;
% Traslada la nube de puntos para obtener su centroide en el origen
figure;
elip(1,:)=elipNo(1,:)+2.5;
elip(2,:)=elipNo(2,:)-1.0;
% Graficacion de la nube de puntos con centroide en el origen
scatter(elip(1,:),elip(2,:));
axis image;
drawnow;
pause;
% Calcula los Componentes Principales (PC) para verificar que el centroide
% de la nube de puntos esta en el origen
[pc,ev, pvar]=pcacov(cov(elip'))
% Graficacion de los Componentes Principales junto con la nube de puntos
hold on;
plot([-sqrt(ev(1)) sqrt(ev(1))]*pc(1,1),[-sqrt(ev(1)) sqrt(ev(1))]*pc(2,1),'r-');
plot([-sqrt(ev(2)) sqrt(ev(2))]*pc(1,2),[-sqrt(ev(2)) sqrt(ev(2))]*pc(2,2),'g-');
pause;
% Rotacion de la nube de puntos de acuerdo a los Componentes Principales
elipr=(elip'*pc)'
% Graficacion de la nube de puntos alineados
figure;
scatter(elipr(1,:),elipr(2,:));
axis image;
drawnow;
pause;
% Recalcula los Componentes Prinpales para demostrar la alineacion de la
% nube de puntos
[pc2,ev, pvar]=pcacov(cov(elipr'))
```

```
% Graficacion de los Componentes Principales junto con la nube de puntos alineados
hold on;
plot([-sqrt(ev(1)) sqrt(ev(1))]*pc2(1,1),[-sqrt(ev(1)) sqrt(ev(1))]*pc2(2,1),'r-');
plot([-sqrt(ev(2)) sqrt(ev(2))]*pc2(1,2),[-sqrt(ev(2)) sqrt(ev(2))]*pc2(2,2),'g-');
pause;
% Reduccion de dimensionalidad (asignando y=0 a la nube de puntos)
elipr(2,:)=0;
% Graficacion de la nube de puntos con reduccion de dimension
figure;
scatter(elipr(1,:),elipr(2,:));
axis image;
drawnow;
pause;
% Graficacion del Componente Principal dominante, sobre la nube de puntos
% modificada
hold on;
plot([-sqrt(ev(1)) sqrt(ev(1))]*pc2(1,1),[-sqrt(ev(1)) sqrt(ev(1))]*pc2(2,1),'r-');
pause;
% Aplicando rotacion inversa de acuerdo a los Componentes Principales
elip=(elipr'*inv(pc))'
% Graficacion de la nube de puntos alineados
figure;
scatter(elip(1,:),elip(2,:));
axis image;
drawnow;
pause;
% Graficacion del Componente Principal (solo uno, por la reduccion dimensional)
% junto con la nube de puntos modificada
hold on;
plot([-sqrt(ev(1)) sqrt(ev(1))]*pc(1,1),[-sqrt(ev(1)) sqrt(ev(1))]*pc(2,1),'r-');
pause;
% Cierre de toda figura; las variables permanecen en memoria.
close all;
```

Apéndice C. OpenGL - Graphics Pipeline.

De modo general, el *Pipeline de Gráficos* simboliza a la conversión de la escena (con modelos 3D) en la imagen de despliegue 2D que recibe el usuario:

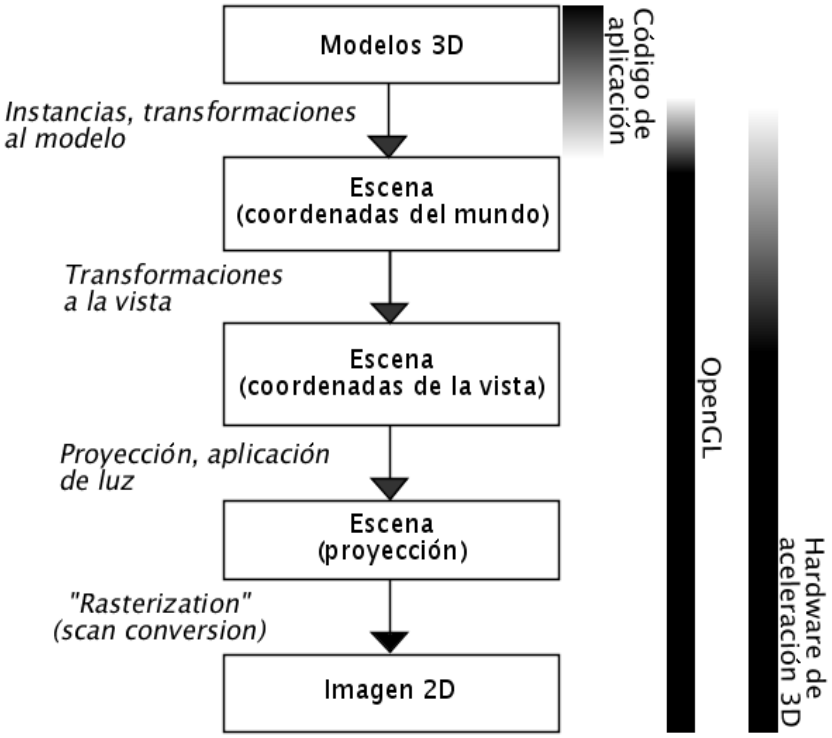
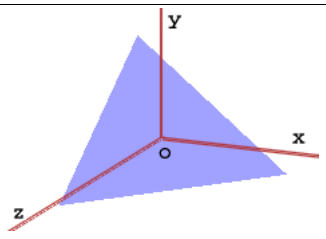


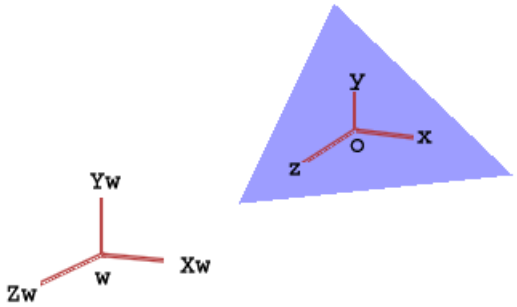
Figura C.1. ***Graphics Pipeline en OpenGL*.**
***Escena*: Modelos 3D y transformaciones que les son aplicadas.**
***Imagen*: Pixeles 2D.**
***Rendering*: Conversión de escena a imagen.**
***Mecanismo para aplicar rendering, en pasos: Graphics Pipeline*.**

Con ello, las coordenadas de objeto para cada polígono (triángulo), son transformadas en las coordenadas pixeles de imagen [Rad04],



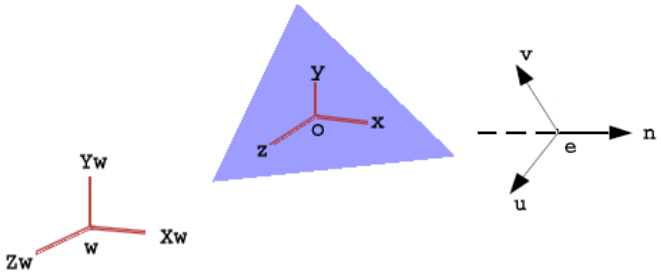
Se inicia con el *Sistema de Coordenadas de Objeto* (OCS),

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{OCS}$$



Con el *Sistema de Coordenadas del Mundo* (WCS), se aplican transformaciones necesarias a todo objeto, colocándolo en el *mundo*,

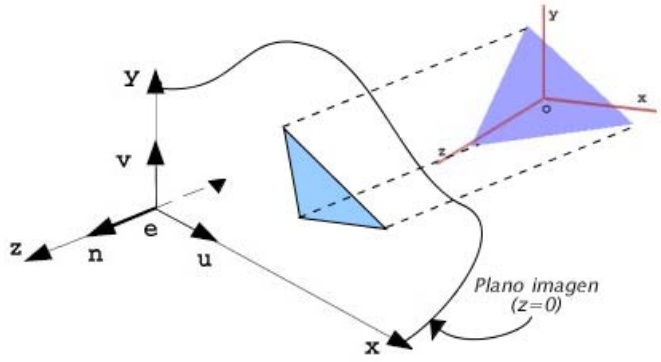
$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{WCS} = \begin{bmatrix} \\ M_w \\ \\ \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{OCS}$$



A continuación se introduce la posición del ojo (*e*) que visualiza a la escena, mediante el *Sistema de Coordenadas de Vista* (VCS).

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{VCS} = \begin{bmatrix} & M_v^{-1} & \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{WCS}$$

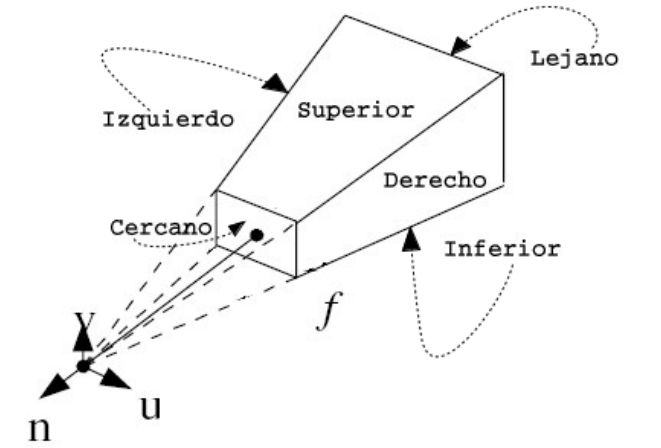
Se le pude interpretar como la traslación del ojo (e) al origen del objeto (o), y alineando (u, v, n) a (x, y, z).



Proyección Ortho

Posteriormente, se hace un corte a la escena, mediante una transformación que la proyecta al plano imagen, usando el *Sistema de Coordenadas de Corte* (CCS). Para este proyecto se implementaron dos posibilidades de proyección:

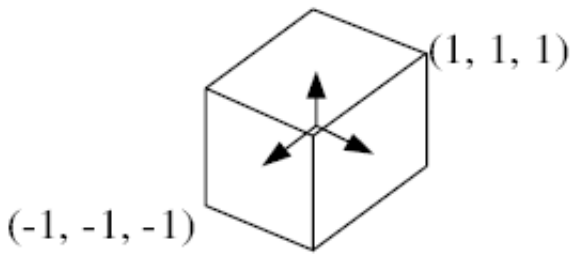
Ortho – Su transformación se encarga de crear el plano imagen 2D con z=0, al proyectar a los vértices de objetos en dirección paralela al eje Z.



Proyección Frustum

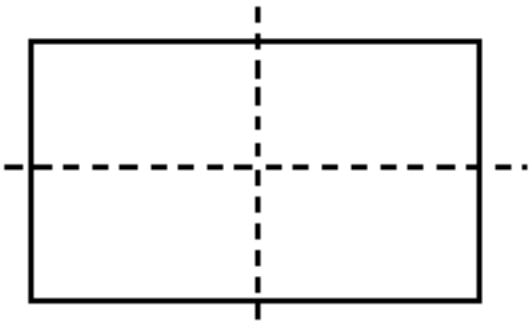
Frustum – Este caso presenta mayor complejidad, ya que la proyección no es paralela al eje Z: se busca proyectar la escena a la cúspide de la pirámide rectangular; la imagen plano que regresa es un corte previo a la cúspide (el plano cercano).

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}_{CCS} = \begin{bmatrix} & M_p & \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{VCS}$$



El siguiente paso se encarga de crear la vista canónica al hacer uso del *Sistema de Coordenadas de Dispositivo Normalizado* (NDCS).

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{NDCS} = \frac{1}{w} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}_{CCS}$$



Finalmente se transforman las coordenadas normalizadas al flujo de datos apto para su despliegue en el dispositivo correspondiente (usualmente el monitor), mediante el *Sistema de Coordenadas de Dispositivo* (DSC).

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{DCS} = \begin{bmatrix} \boldsymbol{M}_{viewport} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{NDCS}$$

Tabla C.1. Transformaciones en *Graphics Pipeline*.

Las transformaciones previas (*Graphics Pipeline*), se implementan en OpenGL mediante la multiplicación de las matrices correspondientes:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{DCS} = \underbrace{\begin{bmatrix} \boldsymbol{M}_{viewport} \end{bmatrix}}_{\substack{\textbf{\textit{Matriz OpenGL}} \\ \textbf{\textit{VIEWPORT}}}} \underbrace{\frac{1}{w} \begin{bmatrix} \boldsymbol{M}_{p} \end{bmatrix}}_{\substack{\textbf{\textit{Matriz OpenGL}} \\ \textbf{\textit{PROJECTION}}}} \underbrace{\begin{bmatrix} \boldsymbol{M}_{v}^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{M}_{w} \end{bmatrix}}_{\substack{\textbf{\textit{Matriz OpenGL}} \\ \textbf{\textit{MODELVIEW}}}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{OCS}$$

Tipo de Transformación	*Matriz OpenGL afectada*	*Funcion(es) OpenGL*
Modelado	MODELVIEW	*glTranslatef()* *glRotatef()* *glScalef()*
Vista	MODELVIEW	*gluLookAt()*
Proyección	PROJECTION	*glFrustum()* *gluPerspective()* *glOrtho()* *gluOrtho2D()*
Vista	VIEWPORT	*glViewPort()*

Tabla C.2. Algunas transformaciones en OpenGL para *Graphics Pipeline*.

Apéndice D.1. Diseños de plantilla XSL y esquema XML.

Como se especifica en el capítulo IV, el despliegue HTML de la información geométrica y mecánica en un modelo 3D PIF, se logra mediante la transformación XSL [Bon05] de un esquema XML, la cual se establece en la plantilla **information.xsl**, que se muestra a continuación,

XSL

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
 <head><title>PIF: 3D Model Information.</title></head>
 <body><center>
 <h2><i>3D Model Information.</i></h2>
 <xsl:apply-templates/>
 </center></body>
</html>
</xsl:template>
<!--
       XSL stylesheet. Information of PIF 3D Model
       Thesis 2005, Waldo Ram&iacute;rez Monta&ntilde;o.
-->
<xsl:template match="model"><center>
  File: <b>&quot;<xsl:value-of select="name"/>.pi&quot;</b><br/>
  Number of points: <b><xsl:value-of select="np"/></b>,
  number of triangles: <b><xsl:value-of select="nt"/></b>.<br/><br/>
  <i>Geometric centroid (x, y, z):</i><br/>
  (<xsl:value-of select="cg/cx"/>,
   <xsl:value-of select="cg/cy"/>,
   <xsl:value-of select="cg/cz"/>)
  <br/><br/>
 <table width="70%">
  <tr>
    <td align="center"><i>Eigenvalues.</i></td>
    <td align="center"><i>Eigenvectors.</i></td>
  </tr>
  <tr>
    <td align="center"><b>(1)</b> <xsl:value-of select="ev/val0"/>.</td>
    <td align="center"><b>(1)</b> (<xsl:value-of select="evec/vec0"/>).</td>
  </tr>
  <tr>
    <td align="center"><b>(2)</b> <xsl:value-of select="ev/val1"/>.</td>
    <td align="center"><b>(2)</b> (<xsl:value-of select="evec/vec1"/>).</td>
  </tr>
  <tr>
    <td align="center"><b>(3)</b> <xsl:value-of select="ev/val2"/>.</td>
    <td align="center"><b>(3)</b> (<xsl:value-of select="evec/vec2"/>).</td>
  </tr>
 </table><br/>
 <i>Inertia Tensor:</i><br/>
 <xsl:for-each select="ti/row">
  <xsl:value-of select="."/><br/>
 </xsl:for-each><br/>
 <i>PCA: "abc" values:</i>
 <table width="70%">
```

```
  <tr>
    <td align="center"><i>Inertia Ellipsoid.</i></td>
    <td align="center"><i>Homogeneous Ellipsoid.</i></td>
  </tr>
  <xsl:for-each select="ellipsoid">
  <tr>
    <td align="center"><b>(1)</b> <xsl:value-of select="inertia/abc0"/>.</td>
    <td align="center"><b>(1)</b> <xsl:value-of select="homogeneous/abc0"/>.</td>
  </tr>
  <tr>
    <td align="center"><b>(2)</b> <xsl:value-of select="inertia/abc1"/>.</td>
    <td align="center"><b>(2)</b> <xsl:value-of select="homogeneous/abc1"/>.</td>
  </tr>
  <tr>
    <td align="center"><b>(3)</b> <xsl:value-of select="inertia/abc2"/>.</td>
    <td align="center"><b>(3)</b> <xsl:value-of select="homogeneous/abc2"/>.</td>
  </tr>
  </xsl:for-each>
 </table>
</center></xsl:template>
</xsl:stylesheet>
```

El esquema XML de los modelos 3D PIF, proporciona la siguiente información:

Parámetro	Etiqueta (tag) XML
Nombre del modelo	name
Cantidad de puntos	np
Cantidad de triángulos	nt
Centroide geométrico	cg
valor x	cx
valor y	cy
valor z	cz
Eigenvectores	evec
vector 0	vec0
vector 1	vec1
vector 2	vec2
Eigenvalores	ev
valor 0	val0
valor 1	val1
valor 2	val2
Tensor de inercia	ti
fila 0	row 1
fila 1	row 2
fila 2	row 3
Valores a, b, c de elipsoide...	ellipsoid
de inercia	inertia (abc0, abc1, abc2)
homogénea	homogeneous (abc0, abc1, abc2)

Tabla D.1.1. Información del modelo PIF, proporcionada por su archivo XML.

Y lo presenta mediante el siguiente formato,

XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="./information.xsl"?>
<!-- File created with exportaInfoXML -->
<!-- Thesis 2005, Waldo Ram&iacute;rez Monta&ntilde;o. -->
<model>
 <name>Nombre del modelo</name>
 <np>Número entero</np>
 <nt>Número entero</nt>
 <cg>
  <cx>Número real (10 decimales)</cx>
  <cy>Número real (10 decimales)</cy>
  <cz>Número real (10 decimales)</cz>
 </cg>
 <evec>
  <vec0>Número real (10 decimales), Número real (10 decimales), Número real (10 decimales)</vec0>
  <vec1>Número real (10 decimales), Número real (10 decimales), Número real (10 decimales)</vec1>
  <vec2>Número real (10 decimales), Número real (10 decimales), Número real (10 decimales)</vec2>
 </evec>
 <ev>
  <val0>Número real (10 decimales)</val0>
  <val1>Número real (10 decimales)</val1>
  <val2>Número real (10 decimales)</val2>
 </ev>
 <ti>
  <row n="0">Tres números reales (10 decimales), separados por espacio</row>
  <row n="1">Tres números reales (10 decimales), separados por espacio</row>
  <row n="2">Tres números reales (10 decimales), separados por espacio</row>
 </ti>
 <ellipsoid>
  <inertia>
   <abc0>Número real (15 decimales)</abc0>
   <abc1>Número real (15 decimales)</abc1>
   <abc2>Número real (15 decimales)</abc2>
  </inertia>
  <homogeneous>
   <abc0>Número real (15 decimales)</abc0>
   <abc1>Número real (15 decimales)</abc1>
   <abc2>Número real (15 decimales)</abc2>
  </homogeneous>
 </ellipsoid>
</model>
```

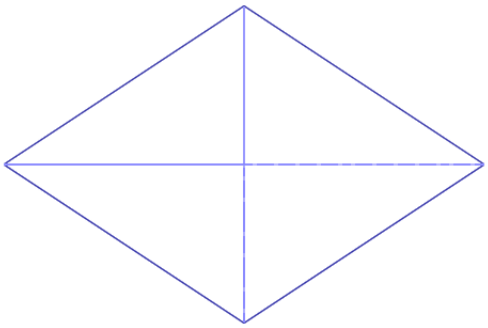
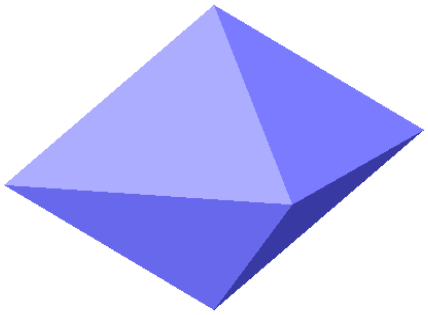
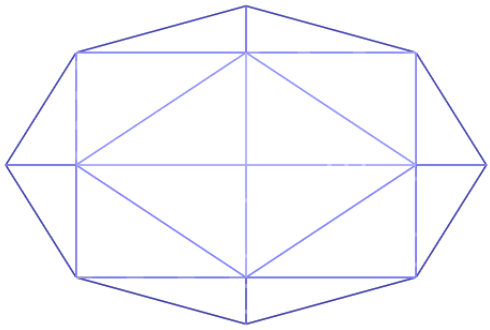
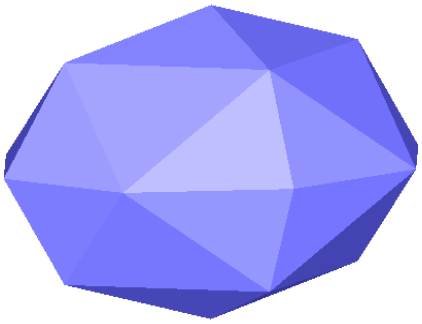
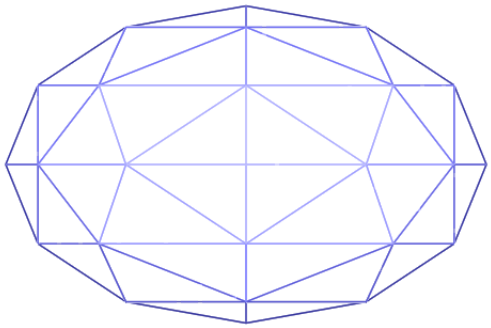
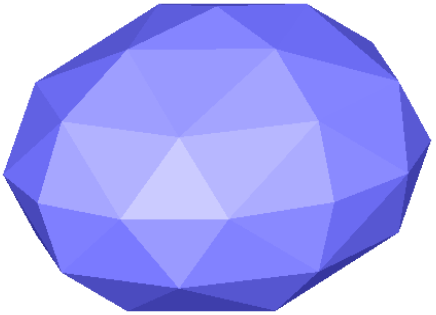
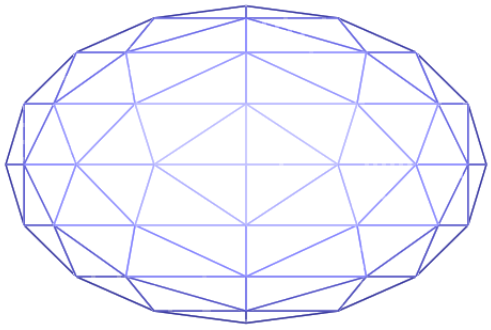
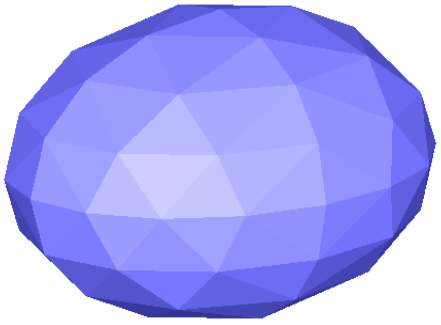
Cabe señalar que se verificó su reconocimiento en diversos navegadores de Internet:

- Mozilla 1.6.
- Mozilla Firefox 1.0.3.
- Epiphany 1.0.7.
- Netscape 7.

Apéndice D.2. Generador de elipsoides.

Mediante el algoritmo expuesto en el capítulo IV, el *método de subdivisión angular* permite crear a distintos modelos de superficie discreta que representan a una misma elipsoide 3D canónica, como se ejemplifica con los valores $a = 150.0,\ b = 99.0,\ c = 120.0,$ en la tabla D.2.1:

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}=1\ ,\ \text{por ejemplo:}\ \ \frac{x^2}{150.0^2}+\frac{y^2}{99.0^2}+\frac{z^2}{120.0^2}=1$$

Subdivisión angular			**Wireframe**	**Sólido**
n	***np***	***nt***		
1	6	8		
2	18	32		
3	38	72		
4	66	128		

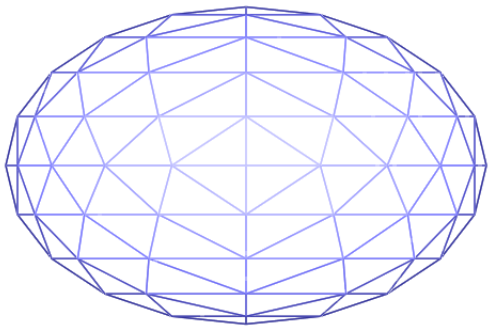
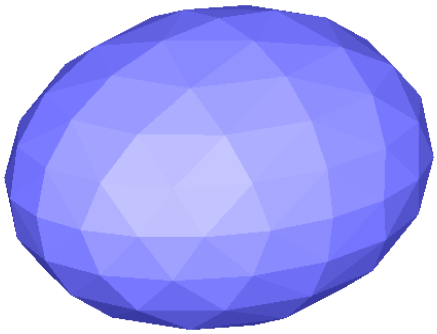
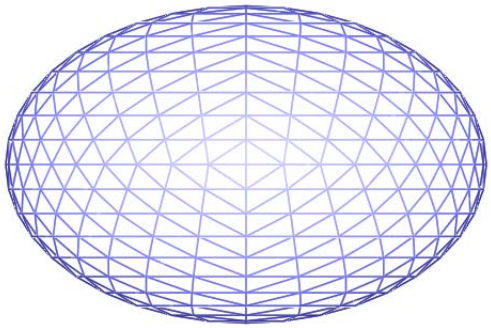
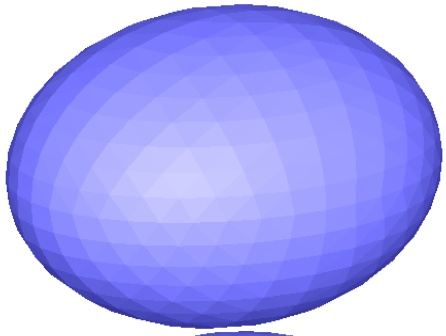
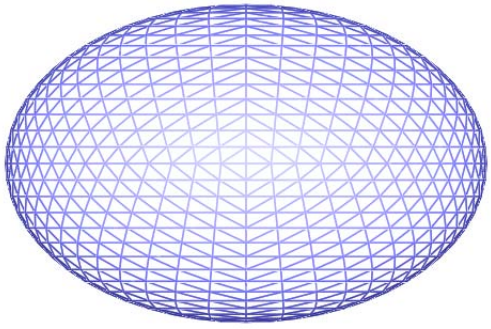
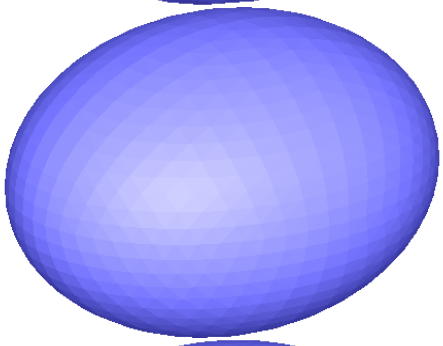
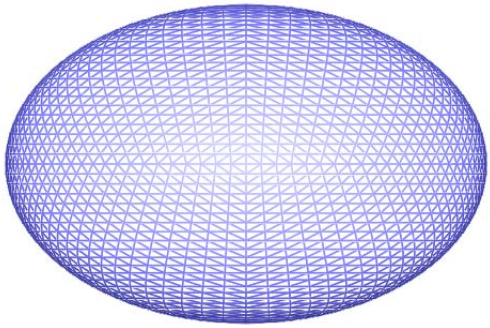
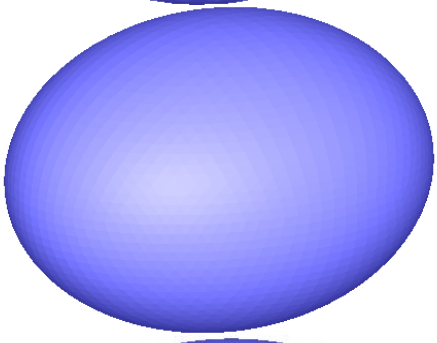
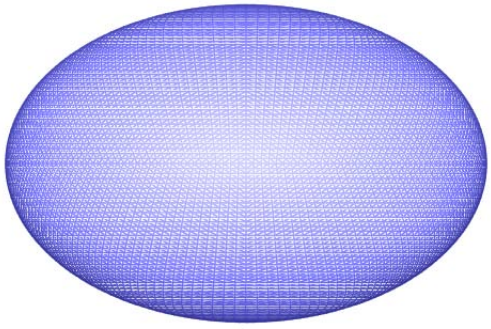
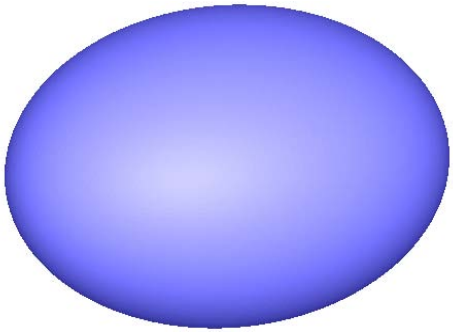
5	102	200		
10	402	800		
15	902	1800		
25	2502	5000		
50	10002	20000		

Tabla D.2.1. Nomenclatura: n-nivel, np-número de puntos, nt-número de triángulos.
Figuras obtenidas con el Visor PIF 3D, plano XY con proyección Ortho.

Apéndice D.3. Cálculo del Campo de Distancia Euclidiano.

0) *Triángulos 3D a 2D.* Este punto no se menciona explícitamente en el algoritmo, ya que establece el marco de trabajo para el cálculo de las muestras del campo de distancia, y por lo tanto, es la base lógica-matemática de procesos clave en el algoritmo. Después del punto 4, se le describe.

1) *Proceso de filtrado* (asignación de zona representativa).
Consiste en dividir al dominio $\boldsymbol{M} \subset \mathbb{R}^3$ en 27 zonas. Con el fin de evitar valores de distancia aleatorios e indeterminados se toma en cuenta a la zona "27+1", que es el complemento de $\boldsymbol{M}$: su unión forma al espacio $\mathbb{R}^3$.

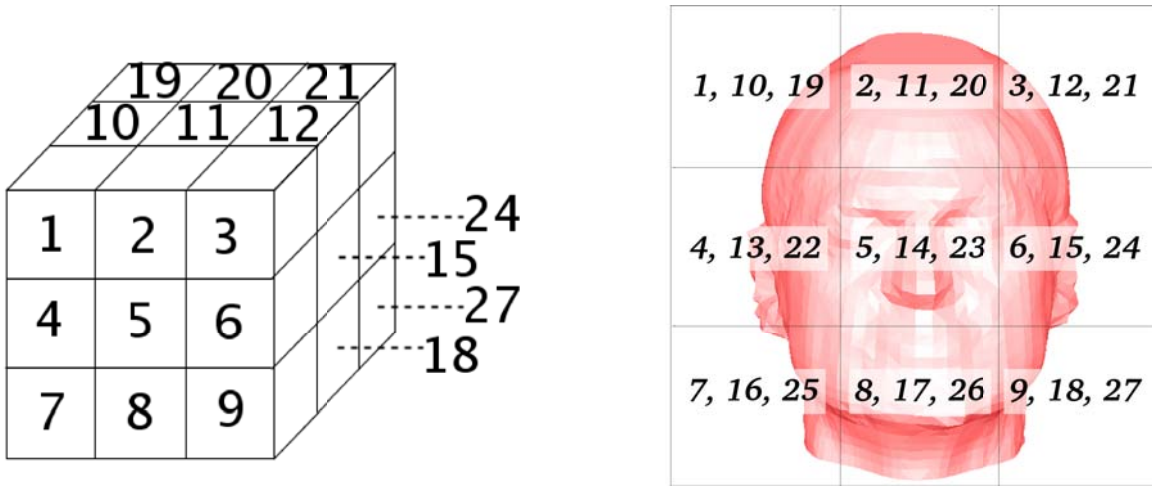


Figura D.3.1. Filtrado del dominio *M*.
Izquierda: 27 zonas del dominio (la zona 27+1 es el resto del espacio 3D).
Derecha: Ejemplo de un plano coronal con proyección ortogonal en cabeza07.pi.
El volumen de cada zona es de 100x100x100 milímetros cúbicos.

2) *Calculo de la distancia mínima del punto representativo de cada zona.* El volumen de todo el dominio $\boldsymbol{M}$ es un cubo de 300x300x300 [mm^3], en donde el punto representativo de casi toda zona, se obtiene mediante una esfera $\boldsymbol{E}$ con $r = \sqrt{150^2+150^2+150^2}$ [mm] de radio. La única zona excepción, es la 14, en donde su punto representativo es el origen. También se observa que los puntos representativos R1, R3, R7, R9, R19, R21, R25, R27 y R14 pertenecen al dominio $\boldsymbol{M}$, mientras que el resto *pertenece a la zona 27+1*. Lo anterior unifica al cálculo de la distancia mínima representativa de cada zona, pero en el cálculo del campo de distancia *sólo se analiza a puntos* $\vec{\mathbf{p}} \in \boldsymbol{M}$.

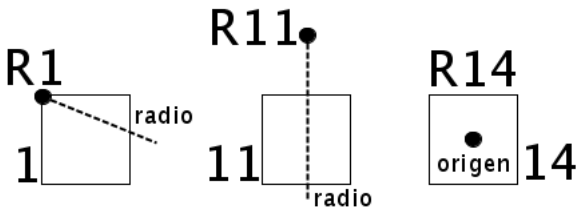


Figura D.3.2. Ejemplo de algunos puntos representativos (proyección ortogonal).
Mediante el radio de la esfera E se obtiene a R1 y R11; R14 es la única excepción posible (se encuentra en el origen).

3) *Obtención del triángulo significativo para cada zona.* A este paso también le llamamos *pasada cero*, ya que consiste en obtener a la distancia mínima de cada punto representativo, al modelo. Los triángulos resultantes se vuelven los triángulos representativos de la zona correspondiente, donde su zona de ubicación la determina su centroide geométrico.

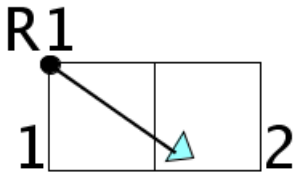


Figura D.3.3. Ejemplo de triángulo significativo (proyección ortogonal).
En este caso, la distancia Euclidiana mínima de R1 al modelo, finaliza en la zona 2, ya que el centroide del triángulo encontrado está en esa zona; se deduce que el triángulo significativo de la zona 1 se encuentra en la zona 2.

La distancia mínima encontrada en cada zona se usa para obtener a cada *patrón de distancia* β_Z que limita la cantidad de cálculos de distancias Euclidianas para todo punto que se encuentre en la zona correspondiente, y con ello: (1) reducir la cantidad de cálculos de distancias necesarias para obtener a la distancia mínima de cada punto $\vec{\mathbf{p}} \in \boldsymbol{M}$ y como consecuencia, (2) reducir el tiempo invertido en la creación de toda la muestra (el plano) del campo de distancia.

4) *Cálculo de las distancias de la muestra (plano) buscada.* En las 27 zonas, se analiza que cada uno de sus puntos:
(1) Cumpla con la tolerancia establecida por el patrón de distancia β_Z correspondiente: actúa como el radio de la esfera que limita al espacio de búsqueda en cada punto.

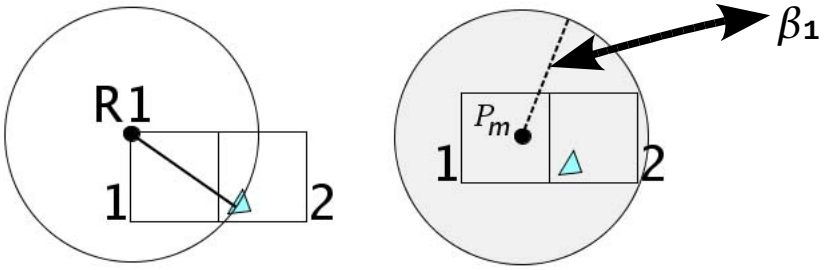


Figura D.3.4. El radio establecido por cada β_Z delimita la cantidad de búsquedas de distancias mínimas para cada punto del dominio *M*.
Izquierda, volumen limitado por β_1. Derecha, aplicación de β_1 en un punto Pm; se reduce la búsqueda de la distancia mínima: Pm consulta sólo con los triángulos que tienen centroide geométrico dentro del volumen de la esfera.

(2) Como lo ilustra la *fig.D.3.4*, para cada triángulo que cumple con la tolerancia β_Z del punto bajo análisis, ambos se convierten en los argumentos del criterio de *Ecuaciones de Frontera* para obtener a la mínima distancia Euclidiana de dicho punto.

Se recuerda que el diseño del algoritmo se basa en los requerimientos establecidos por los modelos analizados: tienden a ser (o son) elipsoides con centroide geométrico en el origen. El uso del algoritmo sobre modelos distintos, pueden generar resultados inesperados de distan-

cias, gracias a la tolerancia esférica de cada β_Z. La solución inicial para su uso en modelos de tendencia distinta, consiste en aplicar directamente al criterio de Ecuaciones de Frontera.

Criterio de *Ecuaciones de Frontera (Edge Equations)*.

El marco de trabajo para el proceso de obtención de la mínima distancia Euclidiana, es resultado de las transformaciones geométricas que convierten a los Triángulos 3D en 2D (sección 4.1). Un ejemplo estándar de triángulos 3D a 2D se ilustra en la *fig.D.3.5,*

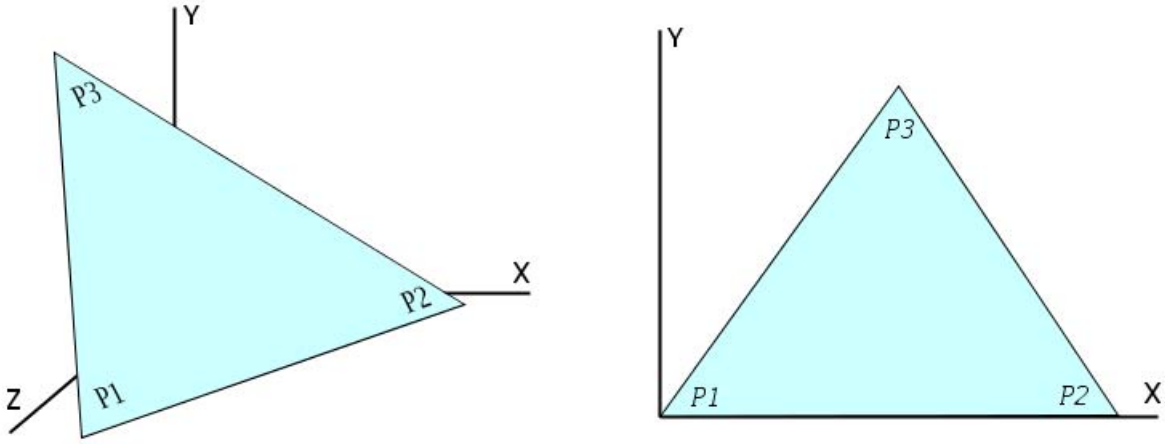


Figura D.3.5. Triángulo 3D a 2D. ***Izquierda: Triángulo 3D original. Derecha: Triángulo 2D resultante de las transformaciones geométricas.***

Debido a casos especiales (*fig.D.3.6*), para obtener a un triángulo 2D se debe cumplir:

- *Presentar las tres alternativas de un triángulo 3D a 2D.* Para ello, es imprescindible obtener los tres grupos de transformaciones geométricas que causan al triángulo 2D (respetando al sentido anti-horario de indexación en sus puntos). Las tres posibilidades permiten asimilar a los casos especiales: cuando las Ecuaciones de Frontera regresan dos resultados de signo negativo y uno de signo positivo (es decir, la interpretación de un punto que está *fuera* de dos lados del triángulo y *dentro* del tercer lado del triángulo).

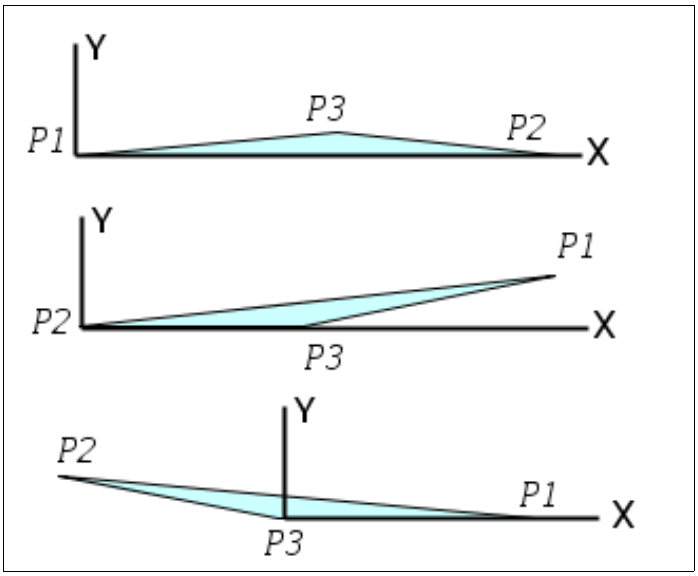


Figura D.3.6. Caso especial de triángulo 2D. ***Para las tres opciones, se cuenta con las matrices de transformaciones geométricas a aplicar al punto bajo análisis, y dependiendo del resultado de las Ecuaciones de Frontera, se elige la opción idónea para calcular la distancia mínima. En este ejemplo se muestra al caso especial que dio énfasis a esta estrategia: la presencia de triángulos que tienden a ser línea (en este ejemplo, P3 tiende a ser parte de P1P2).***

Una vez establecido el marco de trabajo con triángulos 2D, al punto $\vec{\mathbf{p}} \in \boldsymbol{M}$ se le aplican las transformaciones geométricas necesarias y lo nombramos P0, para así poder aplicarle el criterio de Ecuaciones de Frontera con cada lado del triángulo (P1P2, P2P3 y P3P1), y obtener los tres resultados esperados: números reales signados (considerando al cero como positivo) que señalan la ubicación de P0 en el plano XY y lo asocian con: el área del triángulo, ó una recta del triángulo, ó un punto del triángulo (*fig.D.3.7*).

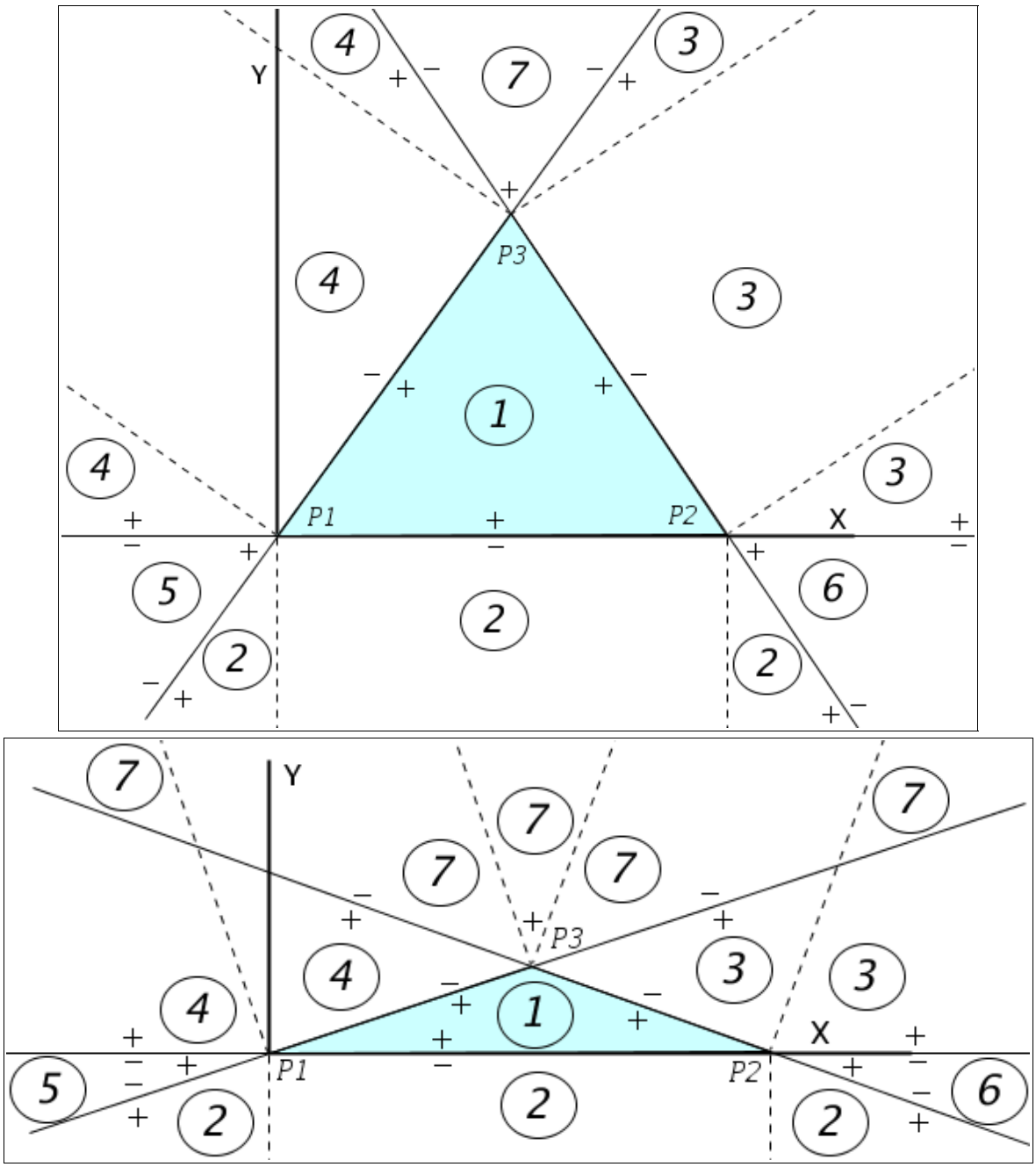


Figura D.3.7. Siete Secciones definidas por las Ecuaciones de Frontera.
Imagen superior: Secciones en un triángulo 2D estándar.
Imagen inferior: Secciones en un triángulo 2D tendiente a línea.
Para ambos triángulos 2D, en cada lado se emplean dos rectas auxiliares ortogonales (en esta figura, son las rectas segmentadas).

Se deduce que el caso ideal es la sección 1, ya que la obtención de la distancia mínima es inmediata (no requiere comparaciones). En el resto de secciones, el dúo de rectas auxiliares ortogonales de cada lado del triángulo 2D, complementan a las Ecuaciones de Frontera: permiten definir a las comparaciones internas de cada sección, eligiendo así a la mínima distancia, como se indica en la *tabla D.3.1* con la nomenclatura siguiente:

Sección: **Interpretación de las Ecuaciones de Frontera; una de siete posibles.**
Signos: **Patrón de tres símbolos (signos), que identifica a cada sección.**
Distancia: **Criterio(s) de elección para cada sección.**

Criterios de Elección de Distancias Mínimas	
Sección y Signos	*Distancia*
1 P1P2(+), P2P3(+), P3P1(+).	*La distancia es el valor de la dimensión Z del punto P0 (es decir, su proyección ortogonal al plano XY).*
2 P1P2(-) P2P3(+) P3P1(+)	Con P1 en el origen y P2 en el eje X positivo: • El valor X de P0 se encuentra en el intervalo de valores X de P1 y P2. *La distancia es del punto P0 a la línea P1P2.* • El valor X de P0 es menor que el valor X de P1. *La distancia es del punto P0 al punto P1.* • El valor X de P0 es mayor que el valor X de P2. *La distancia es del punto P0 al punto P2.*
3 P1P2(+) P2P3(-) P3P1(+)	Con P2 en el origen y P3 en el eje X positivo: • El valor X de P0 se encuentra en el intervalo de valores X de P2 y P3. *La distancia es del punto P0 a la línea P2P3.* • El valor X de P0 es menor que el valor X de P2. *La distancia es del punto P0 al punto P2.* • El valor X de P0 es mayor que el valor X de P3. *La distancia es del punto P0 al punto P3.*
4 P1P2(+) P2P3(+) P3P1(-)	Con P3 en el origen y P1 en el eje X positivo: • El valor X de P0 se encuentra en el intervalo de valores X de P3 y P1. *La distancia es del punto P0 a la línea P3P1.* • El valor X de P0 es menor que el valor X de P3. *La distancia es del punto P0 al punto P3.* • El valor X de P0 es mayor que el valor X de P1. *La distancia es del punto P0 al punto P1.*
5 P1P2(-) P2P3(+) P3P1(-)	Con P1 en el origen, P2 en el eje X positivo y P3 con valor Y positivo: • El valor X de P0 se encuentra en el intervalo de valores X de P1 y P2. *La distancia es del punto P0 a la línea P1P2. Es un caso especial.* • El valor X de P0 es mayor que el valor X de P2. *La distancia es del punto P0 al punto P2. Es un caso especial.* • El valor X de P0 se encuentra entre los valores X de P3 y P1. *La distancia es del punto P0 a la línea P3P1. Es un caso especial.* • La magnitud del valor X de P0 es mayor que la del valor X de P3 sobre el eje X. *La distancia es del punto P0 al punto P3. Es un caso especial.* • Como última alternativa, *la distancia es del punto P0 al punto P1.*

6 P1P2(-) P2P3(-) P3P1(+)	Con P2 en el origen, P3 en el eje X positivo y P1 con valor Y positivo: • El valor X de P0 se encuentra en el intervalo de valores X de P2 y P3. *La distancia es del punto P0 a la línea P2P3. Es un caso especial.* • El valor X de P0 es mayor que el valor X de P3. *La distancia es del punto P0 al punto P3. Es un caso especial.* • El valor X de P0 se encuentra entre los valores X de P1 y P2. *La distancia es del punto P0 a la línea P1P2. Es un caso especial.* • La magnitud del valor X de P0 es mayor que la del valor X de P1 sobre el eje X. *La distancia es del punto P0 al punto P1. Es un caso especial.* • Como última alternativa, *la distancia es del punto P0 al punto P2.*
7 P1P2(+) P2P3(-) P3P1(-)	Con P3 en el origen, P1 en el eje X positivo y P2 con valor Y positivo: • El valor X de P0 se encuentra en el intervalo de valores X de P3 y P1. *La distancia es del punto P0 a la línea P3P1. Es un caso especial.* • El valor X de P0 es mayor que el valor X de P1. *La distancia es del punto P0 al punto P1. Es un caso especial.* • El valor X de P0 se encuentra entre los valores X de P2 y P3. *La distancia es del punto P0 a la línea P2P3. Es un caso especial.* • La magnitud del valor X de P0 es mayor que la del valor X de P2 sobre el eje X. *La distancia es del punto P0 al punto P2. Es un caso especial.* • Como última alternativa, *la distancia es del punto P0 al punto P3.*

Tabla D.3.1. Criterios de Elección de Distancias mínimas.
La presencia de un caso especial (triángulo 2D tendiente a línea), se detecta en las secciones 5, 6 ó 7.
Para cualquier caso indeterminado, se reporta un mensaje de error, además de asignarle distancia cero para facilitar su detección.

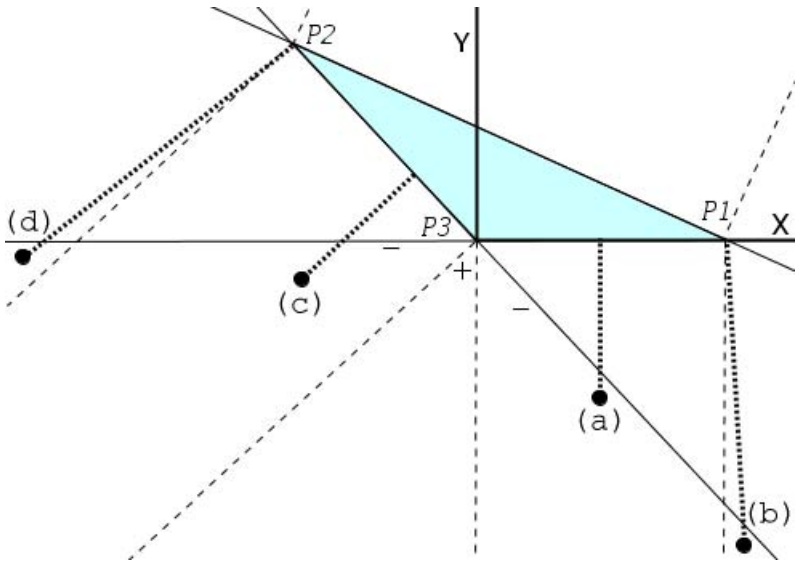


Figura D.3.8. Ejemplo de las cuatro posibilidades de mínima distancia Euclidiana ante casos especiales, en la sección 7.
(a) P0 a la recta P3P1.
(b) P0 al punto P1.
(c) P0 a la recta P2P3.
(d) P0 al punto P2.

5) Como se especifica en el apartado 4.7 (*Estimación de errores morfológicos*), cada muestra del campo de distancia consiste en 8100 valores de distancia en el plano elegido, y como punto final en este algoritmo, se almacena a los 8100 resultados en dos archivos de texto ASCII, así como el historial de transformaciones triángulos 3D a 2D del modelo.
   I. Archivo de 8100 distancias en punto flotante, con 10 dígitos decimales de precisión.
   II. Archivo de 8100 distancias redondeadas (con regresión lineal simple) a valores enteros dentro del rango [0...255], creando una imagen PGM de 90x90 pixels con colores en escala de grises.
      a) Consulta a los 8100 valores de distancia en punto flotante, obteniendo a los dos valores límite del intervalo: distancia máxima y distancia mínima.
      b) Calcula la pendiente m de la recta de regresión lineal simple, y genera su ecuación,

$$m = \frac{255.0}{d_{max} - d_{min}} \quad \text{... pendiente.}$$

$$g = (d - d_{min})\, m \quad \text{... ecuación de la recta.}$$

Donde: d_{max}, es la distancia máxima,
d_{min}, es la distancia mínima,
d, es la variable de 8100 distancias en punto flotante.
g, es el valor entero redondeado.

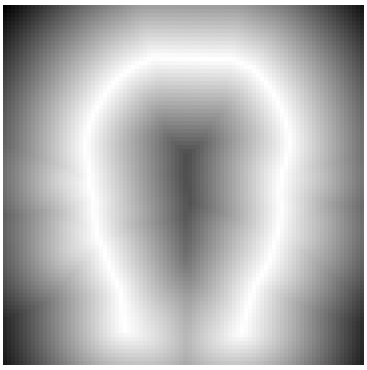


Figura D.3.9. Ejemplo de imagen PGM resultante.
Corresponde al plano coronal del modelo cabeza07.pi

   III. Archivo bitácora que contiene los tres conjuntos de transformaciones triángulo 3D a 2D, para cada triángulo en el modelo.

Notas: El cálculo de campos de distancia es no signado, ya que en la estimación de errores morfológicos sólo interesa la comparación de magnitudes de distancia, sin considerar que el (los) punto(s) esté(n) dentro o fuera del modelo.

Apéndice E. Programas y shell-scripts desarrollados y usados.

Programas

1) *cortaPlano.* Filtro de corte, aplicado a puntos (y triángulos) de un modelo PIF tridimensional. Su utilidad radica en el ajuste previo del casquete de los modelos de cabeza. Requiere de once parámetros, en donde nueve de ellos definen al plano que determina el corte: los puntos que se mantienen en el modelo se encuentran del lado del sentido anti-horario definido por los puntos P1↗P2↘P3, con la sintaxis siguiente:

```
cortaPlano File1 File2 x1 y1 z1 x2 y2 z2 x3 y3 z3

File1 – Archivo original del modelo PIF.
File2 – Archivo resultante del modelo PIF.
x1, y1, z1 – Punto 1 del plano de corte.
x2, y2, z2 – Punto 2 del plano de corte.
x3, y3, z3 – Punto 3 del plano de corte.
```

2) *aproximaElipsoideX0Y0Z0ConPlano.* Filtro de adaptación, aplicado a puntos (y triángulos) de un modelo PIF tridimensional. Su utilidad radica en la supresión de outliers en los modelos de cabeza. Requiere de diecisiete parámetros, los cuales definen al plano y elipsoide límites del ajuste: aplica traslación a todo punto que (1) exceda al espacio que almacena la elipsoide y (2) se encuentre del lado del sentido anti-horario definido por los puntos; la traslación depende de la definición analítica de la elipsoide, con lo cual aproxima a cada outlier al origen de la elipsoide (hasta que dicho outlier cumpla el ser parte del espacio determinado por la elipsoide). Emplea la sintaxis siguiente:

```
aproximaElipsoideX0Y0Z0ConPlano F1 F2 x1 y1 z1 x2 y2 z2 x3 y3 z3 a b c X0 Y0 Z0

F1 – Archivo original del modelo PIF.
F2 – Archivo resultante del modelo PIF.
x1, y1, z1 – Punto 1 del plano de adaptación.
x2, y2, z2 - Punto 2 del plano de adaptación.
x3, y3, z3 - Punto 3 del plano de adaptación .
a, b, c – Ejes analíticos (x, y, z) de la elipsoide.
X0, Y0, Z0 – Centroide (origen) de la elipsoide.
```

3) *dfields.* Obtiene la muestra del campo de distancia Euclidiano correspondiente a un modelo PIF tridimensional, generando (1) el archivo de distancias Euclidianas con 10 dígitos decimales (en milímetros) y (2) la imagen representativa (en escala de grises). El programa se limita a obtener muestras de planos XY, YZ, ó XZ ortogonales (con dimensión de 300X300 milímetros), para lo cual requiere cuatro parámetros con la sintaxis siguiente:

```
dfields model df_file plane_ID plane_int

model – Archivo del modelo PIF.
df_file – Archivo resultante con las distancias Euclidianas.
plane_ID – Indicador del plano objetivo: puede ser XY ó YZ ó XZ.
plane_int – Entero que establece la traslación del plano ortogonal solicitado.
```

4) *nuevoNivelElipsoide.* Generador de modelo de elipsoide nivel N, a partir de un modelo de elipsoide nivel 1. Se encarga de obtener a los ejes analíticos (a, b, c) de la elipsoide, y a partir de ellos crea la nueva elipsoide con un nivel mayor que uno, solicitando tres parámetros con la sintaxis siguiente:

```
nuevoNivelElipsoide F1 F2 n

F1 – Archivo del modelo elipsoide PIF (debe ser elipsoide nivel 1).
F2 – Archivo resultante que almacena al nuevo modelo elipsoide PIF.
n – Nivel del modelo elipsoide resultante (debe ser un entero mayor que 1).
```

5) *visor.* Visor3D de uno ó dos modelos PIF tridimensionales, en modo individual ó simultáneo. Emplea Mesa (basado en OpenGL) para desplegar la interpretación gráfica de los modelos PIF. De las diversas funciones que implementa, se destaca (1) la alineación geométrica de los modelos, (2) la generación de elipsoide homogénea, (3) las transformaciones geométricas a nivel modelo ó visual y (3) el soporte de despliegues sólido, wireframe y puntos. Para invocar al resto de funciones, presenta ayuda desde el perfil visual ó mecánico. Su ejecución requiere uno ó dos parámetros, mediante la sintaxis siguiente:

```
visor F1 [F2]

F1, F2 – Son archivos de modelos PIF.
         El archivo F2 es opcional y debe ser un archivo distinto a F1.
```

Shell-Scripts

1) *abc.sh.* Invoca al programa visor, simplificando la sintaxis al usuario, requiriendo únicamente al número entero que identifica al modelo de cabeza:

```
#!/bin/sh
if test $# -eq 0
  then
    echo "Missing head model's number!"
    echo "Syntaxis: ./abc.sh NN"
    echo "  Where: NN is model's number"
    echo "Waldo Ramirez Montano, Thesis 2005."
    exit 1
fi
echo "Visor3D for cabeza$1.pi..."
echo "./visor ./$1/cabeza$1.pi"
./visor ./$1/cabeza$1.pi
```

2) *e25.sh.* Invoca al programa nuevoNivelElipsoide, con el fin de generar a la elipsoide equivalente (nivel 25) del modelo de cabeza, simplificando la sintaxis al usuario, requiriendo dos parámetros: (1) el número entero que identifica al modelo de cabeza y (2) el número entero que representa a la iteración del proceso.

```
#!/bin/sh
if test $# -le 1
  then
    echo "Missing head model's number and/or iteration!"
    echo "Syntaxis: ./e25.sh NN MM"
    echo "  Where: NN is model's number"
    echo "  Where: MM is the iteration"
    echo "Waldo Ramirez Montano, Thesis 2005."
    exit 1
fi
echo "Generating level25 ellipsoid..."
echo "./nuevoNivelElipsoide ./$1/cabeza$1EHR_.pi ./$1/e$1_$2.pi 25"
./nuevoNivelElipsoide ./$1/cabeza$1EHR_.pi ./$1/e$1_$2.pi 25
```

3) *max_ev.sh.* Invoca al programa visor con el fin de que el usuario traslade al modelo elipsoide equivalente, a la sección del casquete del modelo de cabeza. Lo anterior agiliza la definición de outliers y busca evitar el ajuste de puntos del modelo de cabeza que se desea representar. La sintaxis requiere dos parámetros: (1) el número entero que identifica al modelo de cabeza y (2) el número entero que representa a la iteración del proceso.

```
#!/bin/sh
if test $# -le 1
  then
    echo "Missing head model's number and/or iteration!"
    echo "Syntaxis: ./max_ev.sh NN MM"
    echo "  Where: NN is model's number"
    echo "  Where: MM is the iteration"
    echo "Waldo Ramirez Montano, Thesis 2005."
    exit 1
fi
echo "Visor3D for cabeza$1_.pi and equivalent ellipsoid..."
echo "./visor ./$1/cabeza$1_.pi ./$1/e$1_$2R.pi"
./visor ./$1/cabeza$1_.pi ./$1/e$1_$2R.pi
```

4) *distanceField.sh.* Invoca al programa dfields con el fin de generar a las tres muestras del campo de distancia Euclidiano del modelo de cabeza y las tres muestras del campo de distancia Euclidiano del modelo elipsoide. La sintaxis requiere dos parámetros: (1) el número entero que identifica al modelo de cabeza y (2) el número entero que representa a la iteración del proceso. El diseño del script permite incrementar la cantidad de muestras resultantes, ya que abre la posibilidad de generar mayores planos ortogonales XY, YZ, XZ, mediante la adición de valores a la variable $var, también tomando en cuenta su adición en los nombres de los archivos resultantes.

```
#!/bin/sh
if test $# -le 1
  then
    echo "Missing head model's number and/or iteration!"
    echo "Syntaxis: ./distancefield NN MM"
    echo "  Where: NN is model's number"
    echo "  Where: MM is the iteration"
    echo "Waldo Ramirez Montano, Thesis 2005."
    exit 1
fi
echo "Getting the three planes of the distance flield in model cabeza$1.pi..."
for var in 00; do
  # echo "var is cabeza$var_.pi"
  ./dfields ./$1/cabeza$1_.pi ./$1/YZ/cabeza$1_YZ_$2_.pnm YZ $var
  mv ./triang_a_2D.txt ./$1/YZ/YZ_$2_.t2D
  ./dfields ./$1/cabeza$1_.pi ./$1/XY/cabeza$1_XY_$2_.pnm XY $var
  mv ./triang_a_2D.txt ./$1/XY/XY_$2_.t2D
  ./dfields ./$1/cabeza$1_.pi ./$1/XZ/cabeza$1_XZ_$2_.pnm XZ $var
  mv ./triang_a_2D.txt ./$1/XZ/XZ_$2_.t2D
done
echo "Done with model cabeza$1_.pi"
echo "Getting the three planes of the distance flield in model e$1_$2R_.pi..."
for var in 00; do
  ./dfields ./$1/e$1_$2R_.pi ./$1/YZ/e$1_YZ_$2_.pnm YZ $var
  mv ./triang_a_2D.txt ./$1/YZ/eYZ_$2_.t2D
  ./dfields ./$1/e$1_$2R_.pi ./$1/XY/e$1_XY_$2_.pnm XY $var
  mv ./triang_a_2D.txt ./$1/XY/eXY_$2_.t2D
  ./dfields ./$1/e$1_$2R_.pi ./$1/XZ/e$1_XZ_$2_.pnm XZ $var
  mv ./triang_a_2D.txt ./$1/XZ/eXZ_$2_.t2D
done
echo "Done. Waldo Ramirez Montano, Thesis 2005."
```

5) *sup_out.sh.* (1) Realiza la copia de seguridad del primer grupo de archivos resultantes de la iteración, moviéndolos a su carpeta correspondiente. (2) Invoca (sin parámetros) al programa aproximaElipsoideX0Y0Z0ConPlano, con el fin de recordar al usuario la sintaxis del programa y así poder realizar la supresión de outliers. La sintaxis requiere dos parámetros: (1) el número entero que identifica al modelo de cabeza y (2) el número entero que representa a la iteración del proceso.

```
#!/bin/sh
if test $# -le 1
  then
    echo "Missing head model's number and/or iteration!"
    echo "Syntaxis: ./sup_out.sh NN MM"
    echo "  Where: NN is model's number"
    echo "  Where: MM is the iteration"
    echo "Waldo Ramirez Montano, Thesis 2005."
    exit 1
fi
echo "Backup of cabeza$1.pi iteration's files..."
```

```
echo "mv ./$1/cabeza$1.pi ./$1/cNN/cabeza$1_$2.pi"
mv ./$1/cabeza$1.pi ./$1/cNN/cabeza$1_$2.pi
echo "mv ./$1/cabeza$1EHR_.pi ./$1/eNN/cabeza$1EHR_$2.pi"
mv ./$1/cabeza$1EHR_.pi ./$1/eNN/cabeza$1EHR_$2.pi
echo "mv ./$1/cabeza$1EHR_.xml ./$1/eNN/cabeza$1EHR_$2.xml"
mv ./$1/cabeza$1EHR_.xml ./$1/eNN/cabeza$1EHR_$2.xml
echo "mv ./$1/cabeza$1EHR.pi ./$1/eNN/cabeza$1EHR$2.pi"
mv ./$1/cabeza$1EHR.pi ./$1/eNN/cabeza$1EHR$2.pi
echo "mv ./$1/e$1_$2.pi ./$1/eNN/e$1_$2.pi"
mv ./$1/e$1_$2.pi ./$1/eNN/e$1_$2.pi
echo "mv ./$1/e$1_$2R.pi ./$1/eNN/e$1_$2R.pi"
mv ./$1/e$1_$2R.pi ./$1/eNN/e$1_$2R.pi
echo "mv ./$1/e$1_$2R_.pi ./$1/eNN/e$1_$2R_.pi"
mv ./$1/e$1_$2R_.pi ./$1/eNN/e$1_$2R_.pi
echo "mv ./$1/e$1_$2R_.xml ./$1/eNN/e$1_$2R_.xml"
mv ./$1/e$1_$2R_.xml ./$1/eNN/e$1_$2R_.xml
echo "Use the abc and traslation values to apply outliers criteria with the next
program:"
echo "./aproximaElipsoideX0Y0Z0ConPlano"
./aproximaElipsoideX0Y0Z0ConPlano
```

6) *end.sh.* Realiza la copia de seguridad del segundo grupo de archivos resultantes en la iteración, moviéndolos a sus carpetas correspondientes. La sintaxis requiere dos parámetros: (1) el número entero que identifica al modelo de cabeza y (2) el número entero que representa a la iteración del proceso.

```
#!/bin/sh
if test $# -le 1
  then
    echo "Missing head model's number and/or iteration!"
    echo "Syntaxis: ./end.sh NN MM"
    echo "  Where: NN is model's number"
    echo "  Where: MM is the iteration"
    echo "Waldo Ramirez Montano, Thesis 2005."
    exit 1
fi
echo "Backup of cabeza$1_.pi iteration's files..."
echo "mv ./$1/cabeza$1_.pi ./$1/cNN/cabeza$1_$2A.pi"
mv ./$1/cabeza$1_.pi ./$1/cNN/cabeza$1_$2A.pi
echo "mv ./$1/cabeza$1_.xml ./$1/cNN/cabeza$1_$2A.xml"
mv ./$1/cabeza$1_.xml ./$1/cNN/cabeza$1_$2A.xml
echo "End of iteration $2, for model cabeza$1.pi..."
```

Acrónimos

2D, 3D	*Bidimensional, tridimensional.*
API	*Application Programming Interface.*
ASCII	*American Standard Code for Information Interchange.*
CASE	*Computer Aided Software Engineering.*
CCS	*Clipping Coordinate System. (Véase apéndice C).*
CLI	*Command Line Interface.*
CVS	*Concurrent Versions System.*
DCS	*Device Coordinate System. (Véase apéndice C).*
EDT	*Euclidian Distance Transform (Véase sección 4.5).*
GLU	*The OpenGL Utility Library. (Véase en el glosario).*
GLUT	*The OpenGL Graphics Library Utility Toolkit. (Véase en el glosario).*
GPU	*Graphics Processing Unit. (Véase en el glosario).*
GUI	*Graphical User Interface.*
MVC	*Model View Controller. (Véase sección 3.3).*
NDCS	*Normalized Device Coordinate System. (Véase apéndice C).*
OCS	*Object Coordinate System. (Véase apéndice C).*
OpenGL	*Open Graphics Library.*
PCA	*Principal Components Analysis.*
PGM	*Portable Graymap Format (Véase "Portable Anymap" en el glosario).*
PIF	*Point Index Format (Véase sección 3.1, 4.1).*
PNM	*Portable Anymap (Véase en el glosario ó sección 4.1).*
SAR	*Specific Absorption Rate (Véase en el glosario).*
SCM	*Software Configuration Management. (Véase sección 3.2).*
SOM	*Self-Organizing Map* (*Véase Clustering en el glosario*).
SVD	*Singular Value Decomposition.*
SVN	*Subversion (Véase en el glosario).*
VCS	*Viewing Coordinate System. (Véase apéndice C).*
VRML	*Virtual Reality Modeling Language.*
WCS	*World Coordinate System. (Véase apéndice C).*
X11	*X Windows System (Vease X11 en el glosario).*
XML	*Extensible Markup Language (Véase XML en el glosario).*
XSL	*Extensible Stylesheet Language (Véase XSL en el glosario).*

Glosario

Antropometría	*Disciplina que estudia las medidas y proporciones del cuerpo humano.*
Biometría	*Estudio estadístico de observaciones y fenómenos biológicos.*
Callback	*Código ejecutable que se pasa como parámetro a otro código; lo anterior permite a una capa de software de bajo nivel el invocar a una función para su ejecución en una capa de alto nivel, para así responder al evento.*
Código abierto (Open-source)	*"Open source" denota que los orígenes de un producto de software son públicamente accesibles, en parte o por completo. Lo anterior brinda la disponibilidad gratuita del código fuente, lo cual permite a cualquiera el crear una nueva versión del software y distribuirla.*
Clustering	*Métodos no supervisados de análisis de datos diseñados para descubrir patrones y estructuras en cúmulos o grupos (cluster). Algunos métodos son: jerárquico (construye un árbol jerárquico con ramas de grupos de datos), k-medias (dividen a los puntos de datos en k grupos), SOM (algoritmo que a partir de un proceso de entrenamiento agrupa a los datos de acuerdo al criterio definido). Los grupos resultantes son solo estimaciones de la realidad.*
Dosimetría	*Procedimiento para la cuantificación de la dosis de radiación de cualquier tipo.*
Elipsoide equivalente	*La elipsoide equivalente de un cuerpo es la elipsoide sólida homogénea, centrada en el centro de gravedad del cuerpo, con los mismos momentos y ejes principales de inercia. Una elipsoide sólida homogénea es el cuerpo más simple con tres momentos principales de inercia distintos. Cada cuerpo tiene una elipsoide equivalente única, pero una elipsoide homogénea dada, corresponde a un número infinito de otros cuerpos, más complejos. Las dinámicas rotacionales de un cuerpo dependen sólo de su elipsoide equivalente, no de su forma detallada.*
GLU	*Conjunto de rutinas auxiliares que al hacer uso de comandos del núcleo de la interfaz OpenGL, le complementan, garantizando que cualquier implementación OpenGL le soporta. Nótese que el prefijo para las rutinas de Utility Library es glu, en lugar de gl.*
GLUT	*Conjunto de herramientas multiplataforma que implementan a un API simple: permite construir con facilidad la interfaz de usuario para la aplicación en desarrollo. Nótese que el prefijo para las rutinas de Utility Toolkit es glut, en lugar de gl.*
GPU	*Hardware acelerador 3D presente en tarjetas gráficas avanzadas.*
Graphics Pipeline	*Máquina de estados que establece al proceso encargado en determinar la conversión de las coordenadas de objetos 3D a los pixeles 2D desplegados en el dispositivo de salida. Para mayor detalle consulte al apéndice C.*
Maniquí (Phantom)	*Modelo del objeto o parte anatómica construido(a) físicamente o por computadora para simular efectos físicos en dicho objeto o parte anatómica.*
Morfología	*Rama de la biología que estudia la forma y la estructura de los seres vivos, así como las modificaciones y transformaciones que experimentan.*

Morfometría	*Disciplina que agrupa un conjunto de métodos para la descripción cuantitativa, análisis e interpretación de la forma y su variación. Es rama de diversas ciencias, por ejemplo: biometría, antropometría y geometría.*
Portable Anymap	*PNM, familia de formatos de imágenes: portable bitmaps (PBM, dos colores), graymaps (PGM, escala de grises), y pixmaps (PPM, truecolor) con posibilidad de expresarlos en formato binario ó ASCII. No existe un formato exclusivo de archivo para PNM.*
Punto externo (Outlier)	*Se le llama punto externo a cualquier punto del modelo de cabeza humana bajo análisis, que está "alejado" (al exterior) del modelo preliminar que la representa (en este caso, de su elipsoide homogénea).*
SAR	*Tasa de Absorción Específica. Métrica establecida para realizar la medición de la cantidad de energía electromagnética en radiofrecuencia que es absorbida por el cuerpo humano.*
Subversion	*Sistema de control de versiones bajo licencia de código abierto que como funciones principales, administra el acceso y cambios al repositorio de código del proyecto, y mantiene su historial de cambios realizados.*
X11	*Sistema de ventanas gráficas, para el despliegue de mapas de bits. Es el conjunto de herramientas y protocolo estándares para la construcción de GUI en sistemas operativos UNIX, o similares a UNIX (por ejemplo: Linux).*
XML	*Lenguaje esquemático de propósito general diseñado para poder crear lenguajes esquemáticos de propósitos especiales. Permite describir a distintos tipos de datos, con la meta principal de facilitar su compartición a lo largo de sistemas distintos.*
XSL	*Familia de lenguajes que permite describir el cómo deben ser formateados o transformados los archivos codificados en el estándar XML.*

Bibliografía

Hay un libro abierto siempre para todos los ojos: la naturaleza
Jean Jacques Rousseau

[Ahn95] J.Ahn. *Fast Generation of Ellipsoids.* Graphics Gems V (editado por Alan W. Paeth), pp.179-190.
Systems Egineering Research Insitute, KIST, Corea del Sur.
Academic Press, Inc. 1995.

[AK05] K. Astikainen, R. Kaven. *Statistical analysis of array data: Dimensionality reduction, Clustering.* Febrero, 2005, pp. 4, 8-9.
http://www.cs.helsinki.fi/u/skaski/bioinf_semin05/slides_lect05.ppt

[Bas04] M.C. Bastarrica. *Atributos de Calidad y Arquitectura del Software*. Departamento de Ciencias de la Computación, Universidad de Chile.
JIISIC, 2004.

[Bie03] G. van der Bie. *Anatomy - Human Morphology from a phenomenological point of view*. Sample of the content: *Morphological characteristics based on the skeleton*. Driebergen, Netherlands, Louis Bolk Instituut, Publicación GVO 03 77 pp., 2003.

[Bon05] P.Bondre. *XSLT - Efficient Programming Techniques*. 2005.
http://www.xml.org/xml/resources_featured_articles.shtml

[BS03] H. Sánchez, E. Bribiesca. *Medida de similitud para objetos 2D y 3D a través de una energía de transformación óptima*. Computación y Sistemas. Vol. 7 No. 1 pp. 66-75. © 2003, CIC-IPN, ISSN 1405-5546.

[Cau04] G. Caudillo. *La Antropometría como herramienta de diseño de ortesis y prótesis*. Correos SMOPAC, Vol. 4. Foro de Intercambio Técnico Científico Multidisciplinario Ortoprotésico y Rehabilitación, 2004.

[CFP05] B. Collins-Sussman, B.W. Fitzpatrick, C.M. Pilato. *Version Control with Subversion: For Subversion 1.1: (book compiled from Revision 1337)*. Sitio oficial del documento: http://svnbook.red-bean.com/
O'Reilly Media, 2005.

[Dea05] J. Dean. *Model-View-Controller (MVC) Architecture*. Sitio oficial del documento: http://www.jdl.co.uk/briefings/index.html#mvc
JDL, 2005.

[DH03] C. Ding, X. He. *Principal Component Analysis and Effective K-means Clustering*. University of California, Berkeley. LBNL Tech Report 52983, 2003.

[Dia01] Diagnostic Strategies. *Calculating Similarity Between Cases.*
http://www.DiagnosticStrategies.com

[DRI99] Direct Rendering Infrastructure, Agosto 1999.
http://dri.freedesktop.org/wiki/FrontPage

[DS01] JM. Díaz de la Cruz, J.J. Scala. *Geometría de Masas*. Publicaciones de la ETSIIM. Madrid. MecFunNet, http://mecfunnet.faii.etsii.upm.es/

[DSL02] B. Dogdas, D. Shattuck, R. M. Leahyc. *Segmentation of Skull in 3D Human MR Images Using Mathematical Morphology*. Actos de SPIE Medical Imaging Conference, Vol. 4684, documento 180, pp. 1553-1562, 2002.

[Ell02] J. Ellis. *Part1B Advanced Physics. Dynamics. Lecture Handout: 1.* University of Cambridge, Department of Physics. 2002.

[FD90] P. Feilder, S. Dart. *Software Configuration Management: Advances in Software Development Environments*. Software Engineering Institute, Carnegie-Mellon University, 1990.

[FG95] D. Frerichs, C. Graham. *VRML,* Septiembre 1995. http://silicon-valley.siggraph.org/MeetingNotes/VRML.html

[Hal00] C. Halsall. *Creating Real Time 3D Graphics With OpenGL*. O'Reilly Network, Junio 2000. http://www.oreillynet.com/pub/a/network/2000/06/23/magazine/opengl_intro.html.

[Han01] H. Hantke. *3D-Visualization (VRML) of Morphology and Flow in groyne fields*. EGS XXVI General Assembly, Nice, Francia, Marzo 2001.

[HSEBCKB05] P. Hannah, A. Starr, J. Esteban, C. Bell, P. Civico, G. Kelly, R. Barker. *PCA - principal Component Analysis.* datafusion.org.uk, http://www.eng.man.ac.uk/mech/merg/Research/datafusion.org.uk/techniques/pca.html , 2005.

[Ier04] L. Ieronutti. *Employing Virtual Humans for Education and Training in X3D/VRML Worlds*. Proceedings of LET-Web3D 2004: 1st International Workshop of Web3D Technologies in Learning, Education and Training, pp. 23-27, Octubre 2004.

[Jon95] M.W. Jones. *3D Distance from a Point to a Triangle*. Technical Report CSR-5-95. Department of Computer Science, University of Wales Swansea, Febrero 1995.

[Kil96] M.J. Kilgard. *The OpenGL Utility Toolkit (GLUT) Programming Interface API Version 3*. 1997. http://www.opengl.org/resources/libraries/glut/spec3/spec3.html

[Kle04] L.A. Klein. *Sensor and Data Fusion. A Tool for Information and Decision Making*. SPIE Vol. PM138, pp. 72-73. Julio, 2004.

[Kre99] E. Kreyszig. *Advanced Engineering Mathematics*. John Wiley & Sons, octava edición, pp.1145-1150, 1999.

[Kwo98] Y.H. Kwon. *Mechanical Basis of Motion Analysis*. Visol Coorporation, Corea, 1998.

[Lan05] F.C. Langbein. *Graphics*. Enero, 2005.
http://www.langbein.org/teaching/graphics/

[LCB01] S. Lanzavecchia, F. Cantele, P.L. Bellon. *Alignment of 3D structures of macromolecular assemblies*. Bioinformatics, Vol. 17, N.1, pp. 58-62. Enero 2001.

[LVK01] L.W. Leung, B. King, V. Vohora. *Comparison of Image data Fusion Techniques Using entropy And INI.* Secciones 2.1, 2.2.
22nd Asian Conference on Remote Sensing, Singapure 2001.

[Már04] J. Márquez. *Correction of Non-Linear Geometric Distortions in 2D Images. A tutorial*. Reporte técnico. Laboratorio de Análisis de Imágenes y Visualización, CCADET-UNAM, 2004.

[MBBSG00] J. Márquez, T. Bousquet, I. Bloch, F. Schmitt, C. Grangeat. *Construction of Human Head Models for Anthropometry and Dosimetry Studies of Hand-Held Phones*. Revista Mexicana de Ingeniería Biomédica, Vol. XXI, Núm. 4, pp. 120-128. Octubre-Diciembre 2000.

[MBBSG03] J. Márquez, I. Bloch, T. Bousquet, F. Schmitt, C. Grangeat. *Anatomical Shape-Based Averaging for Computer Atlas Construction and Craniofacial Anthropometry*. EU-LAT, e-Health Cuernavaca, México, 1 - 4 Diciembre 2003.

[MM02] G. Martínez-Santiago, E. Millán-Hernández. *Determinación del perfil antropométrico en una empresa metalmecánica*. Revista LatinoAmericana de la Salud en el Trabajo, Vol.2, Num.1, Enero-Abril 2002.

[MMCL04] J. Meléndez, D.A. Macaya, J. Colomer, D.A. Llanos. *Symptom based representation for dynamic systems diagnosis. Application to Electrical Power Distribution.* Control Engineering and Intelligents Systems – eXiT, IIIA, Universidad de Girona, España. Noviembre 2004.
http://exit.udg.es

[Pau93] B. Paul. *The Mesa 3D Graphics Library*. Agosto 1993.
http://www.mesa3d.org/

[Pos91] J.Poskanzer. *PNM - portable anymap file format*.
Manual de Unix/Linux (comando: man pnm), Septiembre 1991.

[Pre98] R. Pressman. *Ingeniería del Software. Un Enfoque Práctico*. 4ta edición, Editorial McGraw-Hill, 1998.

[PT92] B.A. Payne, A.W. Toga. *Distance field manipulation of surface models*. IEEE Computer Graphics and Applications. Vol. 12, Num.1, pp. 65-71. Enero 1992.

[Rad04] J.Radulovich. *CSCD18 Tutorial Notes.* CSCD18 course WebPage Fall 2004. http://www.utsc.utoronto.ca/~radulov/cscd18/Fall2004/tutorials

[Rit99] G.X. Ritter. *Image Algebra*. CRC press. pp.143. 1999.

[SA03] M. Segal, K. Akeley. *The OpenGL® Graphics System: A Specification (Version 1.5)*. 1992-2003 Silicon Graphics, Inc.

[SDBGSHF04] F. Ségonne, A.M. Dale, E. Busa, M. Glessner, D. Salat, H.K. Hahn, B. Fischl. *A Hybrid Approach to the Skull Stripping Problem in MRI*. Neuroimage, Vol 22 pp. 1060-1075, 2004.

[SF02] Á. de Salles, C. E. Fernández. *Riesgos a la Salud Provocados por los Teléfonos Celulares Móviles - Una Discusión Reabierta: Los Efectos a Largo Plazo*. InfoSUIS, Versión web ISSN 1688-0994, Octubre 2002.

[SGI92] SGI. *OpenGL The Industry's Foundation for High Performance Graphics*. 1992.
http://www.opengl.org/

[SJ01] R. Satherley, M.W. Jones. *Hybrid Distance Field Computation*. Volume Graphics 2001, Springer Verlag, pp. 195-209, 2001.

[Som92] I. Sommerville. *Software Engineering*. Lancaster University. 4ta edición, Addison-Wesley Publishing Company, 1992.

[SOM04] A. Sud, M.A. Otaduy, D. Manocha. *DiFi: Fast 3D Distance Field Computation Using Graphics Hardware*. EUROGRAPHICS, Vol. 23, Num.4, 2004.

[TMR04] A. Turolla, L. Marchesotti, C.S. Regazzoni. *Multicamera Object Tracking in Video Applications*. DIBE, Universidad de Genoa, Italia. Marzo 2004.

[Ver01] M. Verleysen. *Principal Component Analysis (PCA)*. Université catholique de Louvain, Microelectronics laboratory, Bélgica. Septiembre, 2001.

[Wal94] D. Walker. *Silicon Valley SIGGRAPH meeting Panel: Computer Graphics in Tomorrow's Games*. Octubre 1994.
http://silicon-valley.siggraph.org/MeetingNotes/Games.html

[WMRCS01] A. Wahle, S. Mitchell, S. Ramaswamy, K. Chandran, M. Sonkaam. *Four-dimensional coronary morphology and computational hemodynamics*. Society of Photo-Optical Instrumentation Engineers, 2001.

[WW89] B. Wyvill, G. Wyvill. *Field-Functions for Implicit Surfaces*. University of California, Visual Computer, Num.5, 1989.

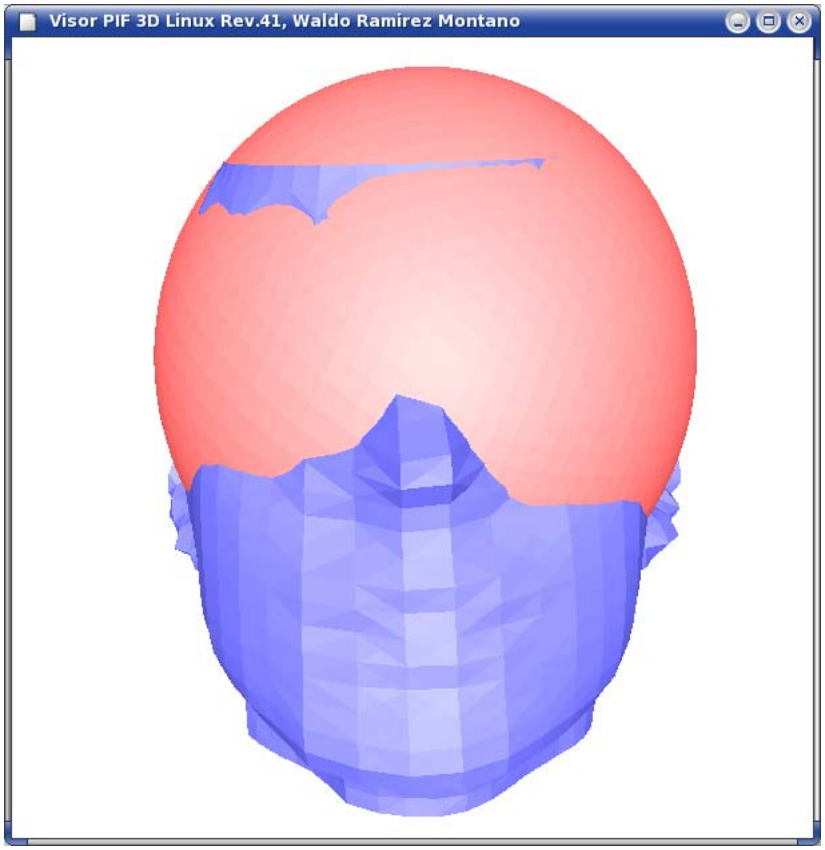
[Yu02] J. Yu. *Retrieval and Clustering of Texts. Application of Singular Value Decomposition Method*. Institute of Computational Linguistics, Peking University, Beijing. 2002.

Agradecimiento a los sitios públicos de frases célebres:
http://www.proverbia.net/
http://www.durango.net.mx/frases/

El decir que no hay nada significa decir que hay algo, tal vez todo.
Waldo Ramírez Montaño

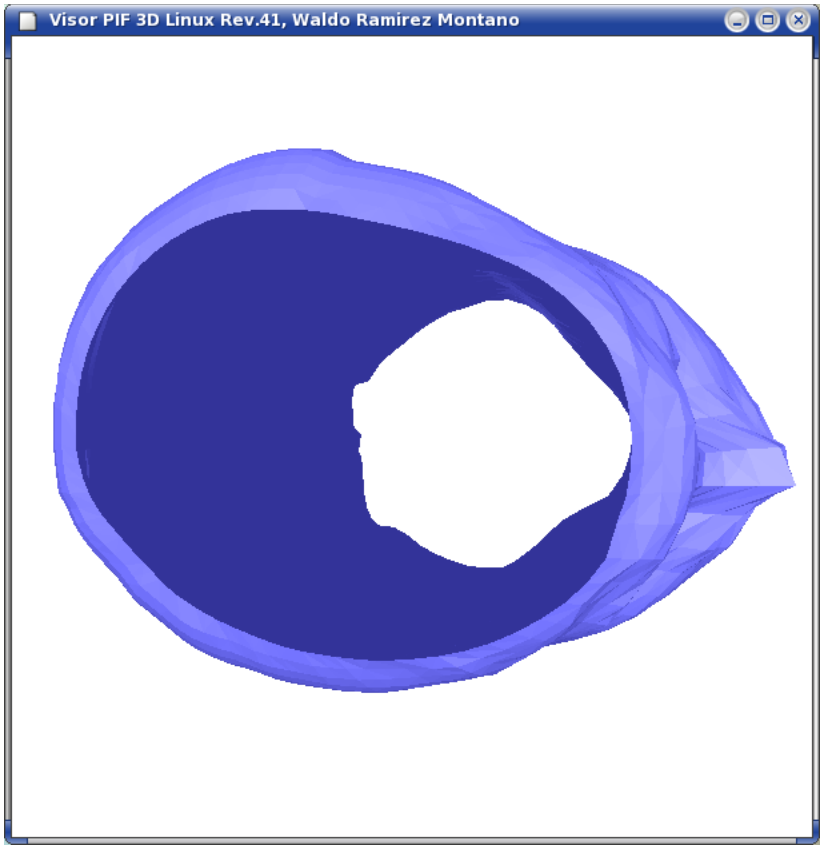
EJEMPLOS ADICIONALES

Las palabras crean imágenes, y las imágenes crean palabras



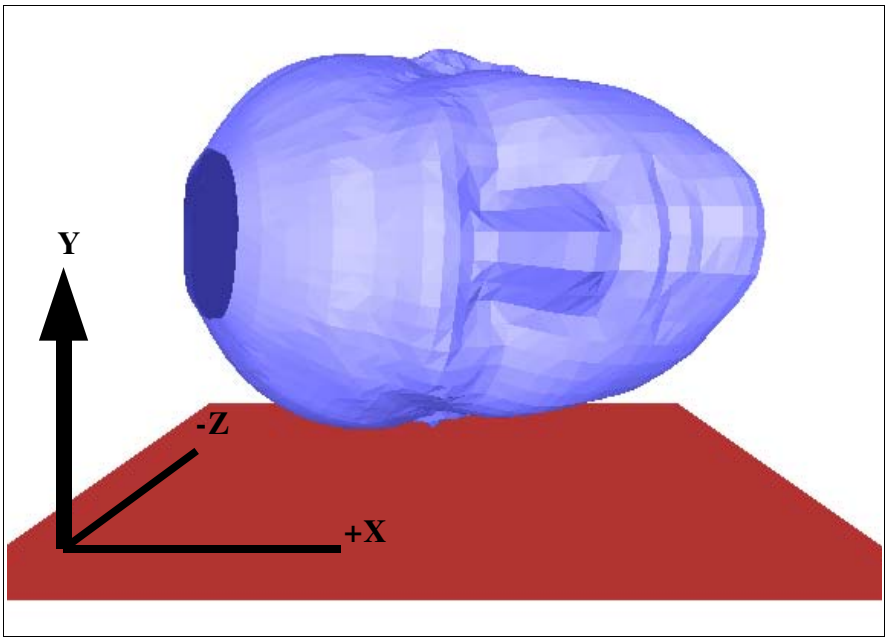
Ejemplo de un modelo cuyo proceso finalizó en la segunda iteración: cabeza38.pi

Se observa que su elipsoide excede al ajuste esperado.

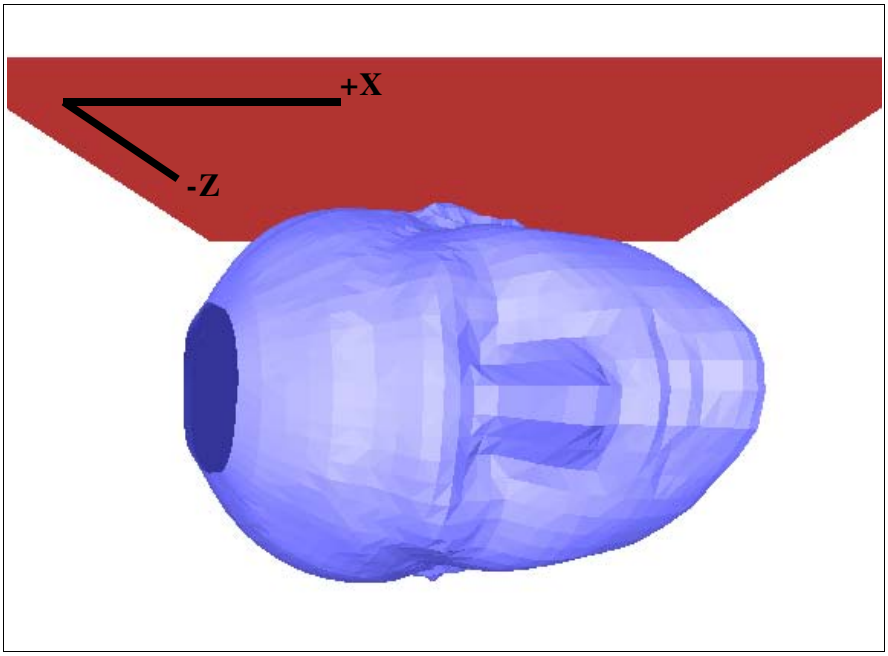


Lo anterior es consecuencia de un corte abrupto en la sección del casquete del modelo cabeza38.pi.

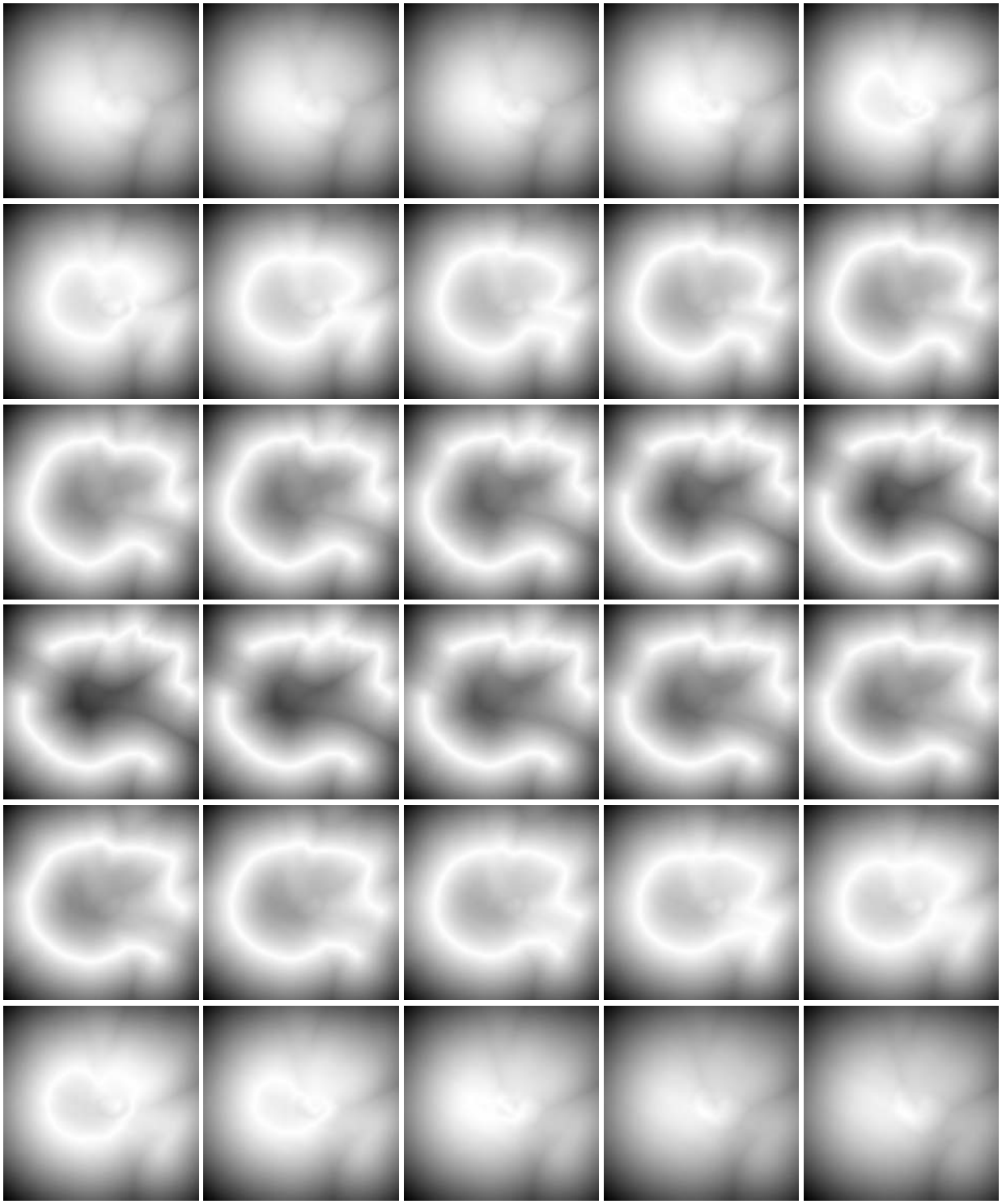
Una extensión del algoritmo podría emplear parches spline, B-spline para completar iterativamente a la sección del corte, obteniendo así una elipsoide de mejor ajuste.



Extensión de la muestra del campo de distancia.
En el diseño del algoritmo que obtiene la muestra del campo de distancia correspondiente al modelo bajo análisis, se consideró la posibilidad de extender a la muestra, mediante la obtención de planos paralelos XY, YZ ó XZ.

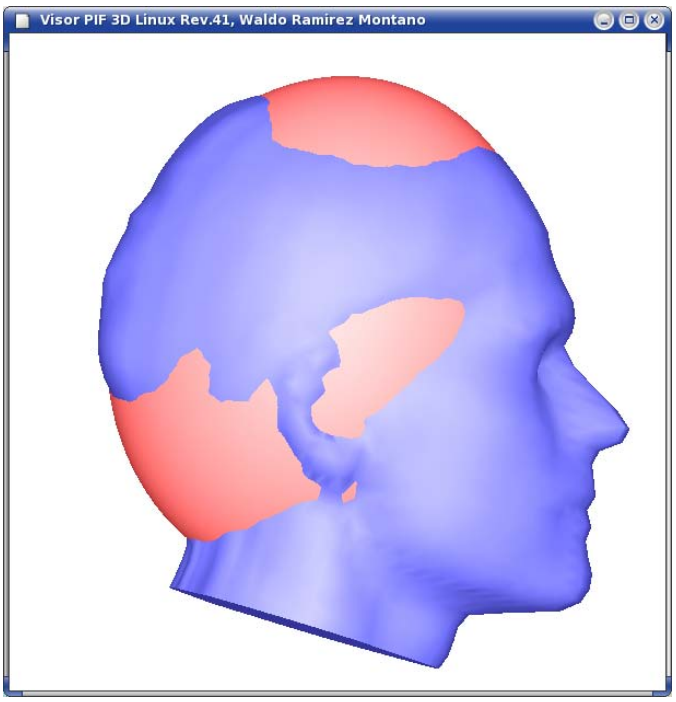


En el ejemplo, el muestreo del campo de distancia del modelo cabeza04.pi inicia con un plano XZ y su valor –Y correspondiente (figura superior), y continua hasta finalizar en un plano XZ con determinado valor +Y (figura inferior). Se tiene la finalidad de generar 30 planos muestra (resultados en la siguiente página).



Extensión de la muestra del campo de distancia, a lo largo de los planos XZ. Modelo: cabeza04.pi (alineado en su primera iteración).

El programa *dfields* permite obtener más de un plano ortogonal XZ (lo cual también aplica para XY, YZ). En este ejemplo, el muestreo XZ inicia en Y=-30 * 3.33333[mm] e incrementa a Y con 6.66666[mm], para cada muestra subsecuente.



En el modelo cabeza04.pi se efectuaron 10 iteraciones, con lo cual el ajuste de su elipsoide mejoró sustancialmente.

Se aprecia que un corte de menor dimensión en el casquete del modelo de cabeza es una mejora para la elipsoide representativa (figura inferior).