Capítulo 2.- Métodos para comparación y visualización

En la comparación de formas de objetos hay distintos métodos recientes [MARQUEZ2006], ya sea por diferencias simétricas o bien por mapeo de un objeto en un campo de distancia del objeto de referencia y asignando el valor leído a la superficie del primero (objeto a comparar), para poder efectuar una comparación exitosa se debe tener en cuenta los parámetros y los atributos extrínsecos e intrínsecos de ambos objetos en cuestión porque de ello dependerá error o diferencia que entre ellos. Dentro de los parámetros y atributos nos hallamos con la alineación de dos objetos en la cual si dos objetos en el espació no están alineados las diferencias que se miden aumentan. En esta tesis la alineación se realiza haciendo transformaciones geométricas [ver anexo A] por parte del usuario. En el laboratorio de LAIV de CCADET ya se desarrolló software que efectúa este trabajo automáticamente utilizando varios métodos como por ejemplo: Por componentes principales o penalización de rasgos. Los rasgos de un objetos que en nuestro caso son cabezas humanas, las diferencias intrínsecas son los rasgos diferentes que tienen dos cabezas como son la posición y/o tamaño de la oreja, volumen de la cabeza, mentón etc.

2.1.- Base de datos: Construcción de modelos de Cabezas con VRML

VRML (Virtual Reality Modeling Languaje) es un formato para aplicaciones de Realidad Virtual (VR). Con este formato se pueden contruir mundos virtuales tales como casas, vehículos, personas, escenas, robots, etc. La base de datos de las cabezas humanas están hechas en formato VRML.

La adquisición numérica de cabezas humanas se realizó con un escáner láser 3D de *Cyberware*. Detalles sobre antecedentes del proyecto y construcción de la base de datos se dan en [MARQUEZ2000] . El escáner rota alrededor de la cabeza produciendo información 3D de la distancia, como profundidad (en inglés “range image”). Esta adquisición de datos consiste en tomar los perfiles meridianos y variando el número de puntos dependiendo de los relieves.

También hay paso de incremento Δy entre los puntos 1mm en la altura 'y' de proyección y un paso .5mm en la posición angular θ. Las diferencias de profundidad $\Delta\rho$ pueden variar mucho especialmente detrás de la oreja. Después para la construcción del modelo se utilizan imágenes resultado de la acción del escaner en una resolución de 480×580×5 bytes en punto flotante; las imágenes se filtran para quitar el ruido y los artefactos (zonas de información faltante o espúrea), preservando la nitidez para después localizar con procesamiento de imágenes en coordenadas cilíndricas el objeto escaneado en el espacio[MARQUEZ2000]. La figura 10 y 11 muestra el concepto de escaneo y de interpretación con un cilindro e imágenes en escala de grises

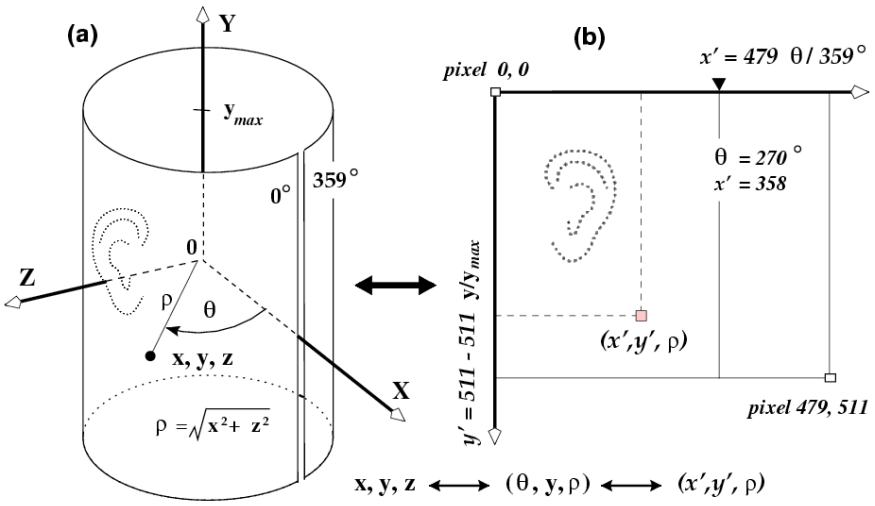

Figura 1: Representación e interpretación del escáner láser

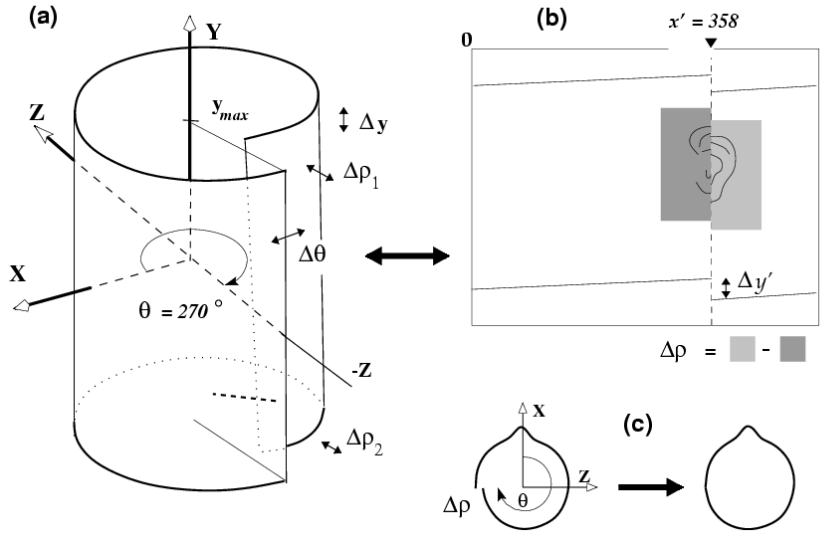

Figura 2

La construcción del *phantom* (mallado triangular) incluye formatos de conversión y técnicas de procesamiento de imágenes, obteniendo así un código de niveles de grises que va cambiando conforme va rotando el escaner. [MARQUEZ2000]

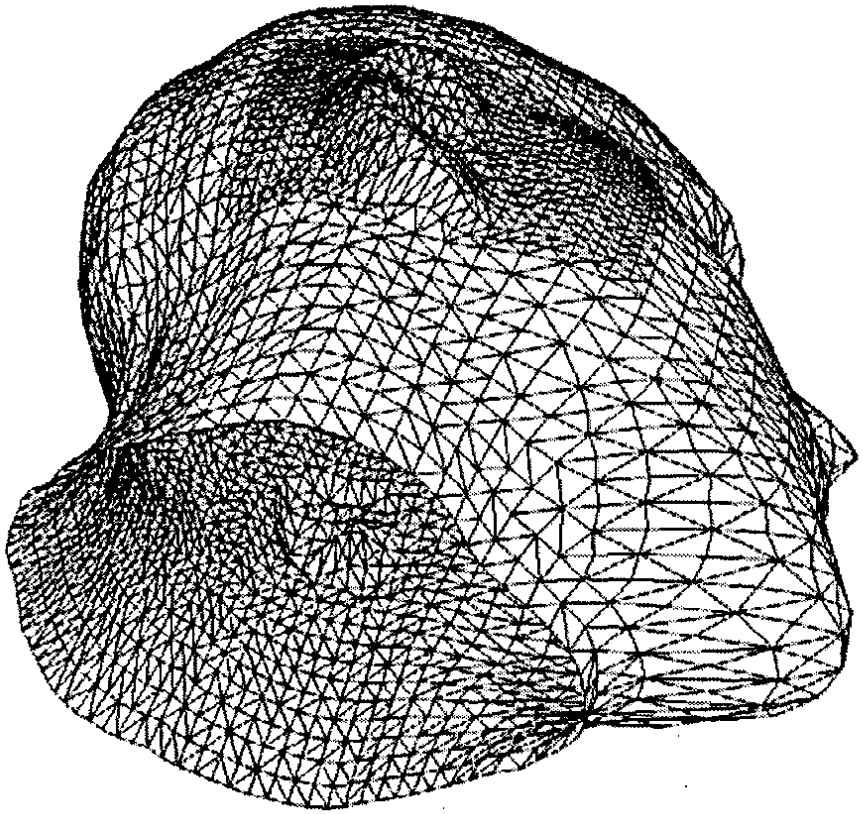

Figura 3: modelo obtenido del escáner láser. La zona alrededor de la oreja se malló a mayor resolución por ser de interés particular en el proyecto original.

El mallado de las cabezas es llevado a un archivo de formato VRML 1.0 el cual lleva una parte que son las coordenadas de los vértices y otra que lleva los índices para mandarlo a dibujar.

En formato VRML 1.0 para describir los vértices se utiliza la palabra “Coordinate3” el cual significa que se pondrán en una matriz de $n \times 3$ todos los vértices de la superficie de interés. Los vértices están relacionados entre sí para hacer una triangulación esta relación se crea con la instrucción “IndexedFaceSet” que al igual a la anterior es una matriz en la cual relaciona los vértices mientras estén antes un valor de “-1”. La matriz de vértices también se interpretan como una lista de ternas en un sistema coordenado de

tres dimensiones y se relacionan con una lista de índices necesarios para triangularizar los vértices y así construir la superficie del mallado.

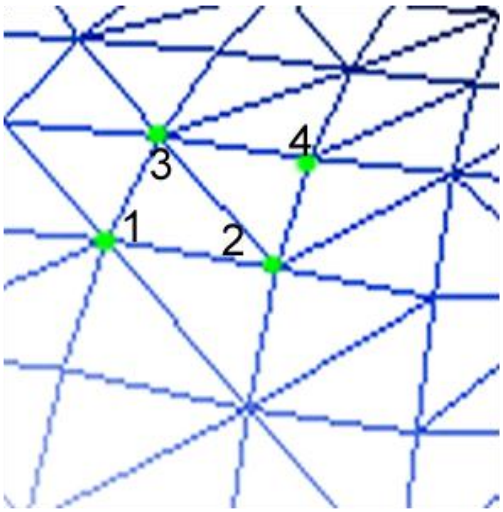

Figura 4: Agrupación de índices y vértices en VRML

Vertices:

```
Coordinate3 {
        point [
                73.381905 -91.289062 -70.172577,
                74.798843 -97.695312 -73.695381,
                        .
        ]

        }
```

Indices:

```
IndexedFaceSet {
        coordIndex [
                1,2,3,-1,
                2,3,4,-1,
                .
                .
                ]
        }
```

2.2.- Campo de distancia con la transformada Euclidiana de distancia (EDT)

Consideremos un objeto binario A, en 3D, el cuál puede tener varias componentes conexas. El *campo de distancia Euclideana* (EDT) en 3D (y mapa, en 2D), o también *transformada de distancia Euclideana* [BORGEFORS86] obtenido a partir de A, es una imagen 3D, o volumen donde el atributo del voxel (o pixel, en 2D) es la distancia Euclideana de ese punto al punto más cercano del objeto A, y de hecho es la distancia al punto más cercano del borde o frontera de A. Matemáticamente, definimos más en general el *EDT con signo del objeto A* como:

$$\mathbf{D}_{\pm}(\partial \mathrm{A}) = \{(\mathbf{p}, d(\mathbf{p})) \mid d = \mathrm{sgn} \cdot \min_{\mathbf{q} \in \partial \mathrm{A}} \|\mathbf{p} - \mathbf{q}\|\} \qquad (1)$$

$$\mathrm{sgn} = \begin{matrix} +1 & if \ \mathbf{p} \in \mathrm{A}^{\mathrm{c}} \\ -1 & if \ \mathbf{p} \in \mathrm{A} \end{matrix} \qquad (2)$$

donde **p**, **q** tienen coordenadas (*x*,*y*,*z*), en el caso tridimensional y "∂A" es la frontera (superficie externa) o "borde tridimensional" del objeto A, y puede tener varias componentes conexas. La superficie externa ∂A de un objeto discreto y binario, en 3D, se encuentra fácilmente directamente de la definición de "frontera" *6-conexa* [HERMAN1998]: es la lista de voxeles de A que tienen al menos un vecino 18-conexo (conectividad por borde) con el fondo (A^c). Si se busca una frontera *18-conexa*, entonces es la lista de voxeles de A que tienen al menos un vecino 6-conexo (por cara) con el fondo (A^c).

Existen varios algoritmos para obtener el campo de distancia en una imagen discreta los cuales se pueden adaptar para obtener el campo de distancia de un objeto tridimensional, entre esos algoritmos hallamos los siguientes:

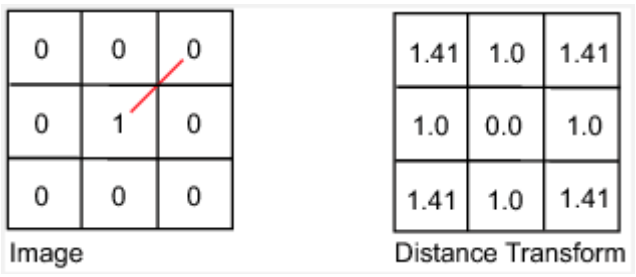

Figura 5

City block

$$V = (x, y) \quad (3)$$

$$D(V_0, V) = (x - x_0) + (y - y_0) \quad (4)$$

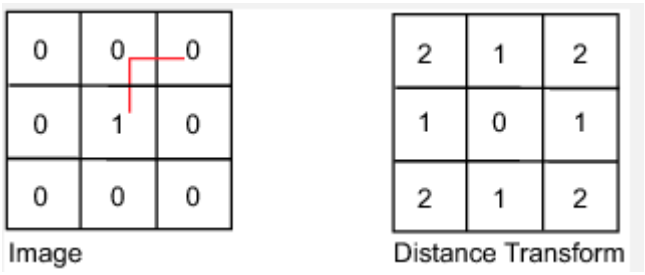

Figura 6

Chessboard

$$V = (x, y) \quad (5)$$

$$D(V_0, V) = \max((x - x_0), (y - y_0)) \quad (6)$$

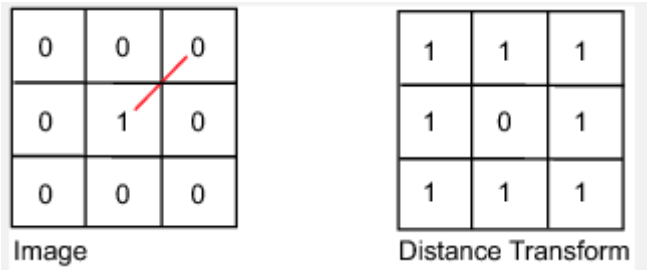

Figura 7

Quasi-Euclidean

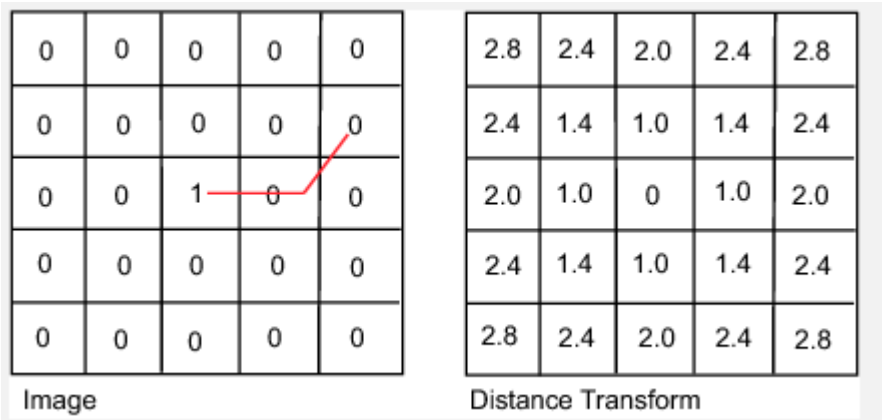

Figura 8

Se debe adaptar a 3 dimensiones algún algoritmo para obtener el campo de distancia, si ahora en vez de ver los pixeles en el plano utilizamos voxeles, entonces el algoritmo se repetiría *n* veces más en un eje de profundidad, como se muestra en la figura 43

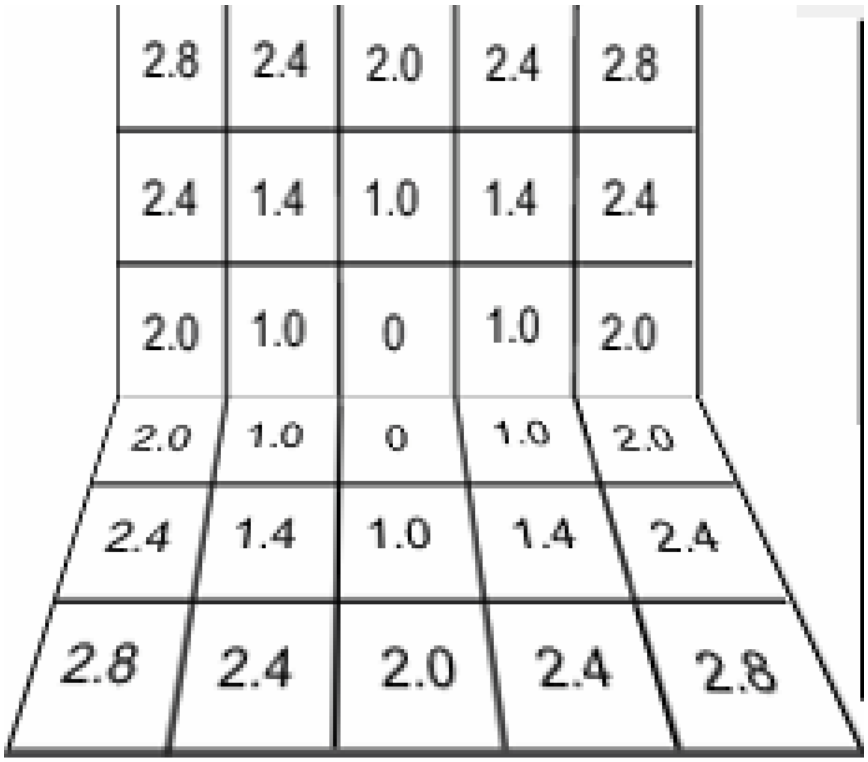

Figura 9: Campo de distancia tridimensional

Usamos en nuestro caso una implementación rápida realizada por el Dr. Márquez, en la cuál se utiliza la versión Euclideana, más exacta y aprovechando tablas de búsqueda que pre-calculan la distancia de una diferencia entre puntos. Dicha implementación, en el caso 2D, se describe en detalle en la tesis [SIMONandDAVIGNON2009].

Lo que se visualiza en la ecuación 1 es la *restricción del campo de distancia (con signo) de un objeto A a un objeto B* (o más bien, a su superficie o frontera *∂B*, representada por un mallado triangular):

$$\mathbf{D}\ \partial A\ \big|_{\partial B} \tag{7}$$

donde *la restricción de f a B, denotada como* $f|_B$ indica considerar sólo aquellos valores de *f* restringidos al conjunto *B* (o sea, *f* queda definida solamennte en (*dom f*) ∩ *B*). Como (1) asigna un valor escalar a cada punto de *∂B*, obtenemos una *frontera atribuida*.

Como los objetos en imágenes binarias, sus fronteras y el campo escalar **D** son también funciones (con atributo “1” donde hay objeto y “0” donde no), podemos escribir:

$$\mathbf{D}\ \partial A \quad x, y, z \ \Big|_{\partial B\ x,y,z} \tag{8}$$

y referirnos a puntos $(x,y,z)\in\partial A$ ó $(x,y,z)\in\partial B$. Equivalentemente, podemos considerar:

$$\mathbf{D}\ \partial B \quad x, y, z \ \Big|_{\partial A\ x,y,z} \tag{9}$$

A y *B* pueden ser cabezas distintas o puede ocurrir que simplemente $A=\mathbf{T}(B)$, o más explícitamente: $A(x,y,z)=B(\mathbf{T}(x,y,z))$, donde **T** es una transformación geométrica, por ejemplo una rotación con traslación, reducción, o en general cualquier deformación no-lineal. Si **T** es la identidad, en el caso anterior tenemos $B=A$ y por tanto:

$$\mathbf{D}\ \partial A \ \Big|_{\partial B} = \mathbf{D}\ \partial A \ \Big|_{\partial A} = 0, \quad \text{para todo}\,(x, y, z)\in\partial A\,, \tag{10}$$

pues por definición, los valores del campo de distancia son justamente cero a lo largo de ∂A. Si **T** es una rotación muy pequeña o cualquier trasformación muy ligera, los valores de $\mathbf{D}\ \partial A \ \Big|_{\partial B}$ serán también de magnitud reducida. La escala de color que se diseñó, permite distinguir valores positivos y negativos de **D,** fuera y dentro de ∂A.

En el caso más general de dos objetos distintos *A*, *B,* lo anterior permite definir un criterio del *grado de alineación* (correspondencia, registro o "matching", en inglés) entre *A* y *B,* permitiendo hallar aquella transformación $\mathbf{T}_{\min}$, aplicada a *B* que minimiza la suma de los valores absolutos del campo de ∂A restringido a ∂B:

$$\mathbf{T}_{\min} = \arg\min_{\mathbf{T}} \sum_{(x,y,z)\in\partial(\mathbf{T}(B))} \text{abs}\ \mathbf{D}\ \partial A \quad x, y, z \ \Big|_{\partial(\mathbf{T}(B))} \tag{11}$$

donde $(\mathbf{T}(B(x, y, z))) = B\ \ \mathbf{T}(x, y, z)\ \ = B(x', y', z')$, de modo que si cambia **T**, se visitan y suman puntos distintos de **D**. Nótese que en el caso de objetos distintos, $A \neq \mathbf{T}_{\min}(B)$. Normalmente la minimización se refiere al conjunto de parámetros de **T** (ángulos, traslación, escalas, etc.) que permiten minimizar la suma anterior, optimizando la alineación entre *A* y *B*.

Una vez alineados lo más posible, el valor mínimo puede interpretarse como una medida de disimilitud entre *A* y *B*, siendo cero cuando son iguales:

$$disim(A,B)=\min_{\mathbf{T}}\left\{\frac{1}{card(\partial(\mathbf{T}(B)))}\sum_{(x,y,z)\in\partial(\mathbf{T}(B))}\text{abs}\ \ \mathbf{D}\ \ \partial A\quad x,y,z\ \Big|_{\partial(\mathbf{T}(B))}\right\}\qquad(12)$$

donde se introdujo una normalización respecto al objeto a comparar, pues la suma es sobre su frontera, tras una transformación. En otras métricas, la normalización se hace respecto al objeto de referencia *A*, el cuál se mantiene fijo, pero, usando la expresión alternativa (3), se calcularía **D**(∂(**T***B*)) con cada cambio en **T,** lo cuál resultaría impráctico. Finalmente, el "grado de alineación" del que se habló, dada cualquier **T**, no necesariamente óptima, es la expresión entre corchetes, en (6), es decir:

$$match(A,B,\mathbf{T})=\frac{1}{card(\partial(\mathbf{T}(B)))}\sum_{(x,y,z)\in\partial(\mathbf{T}(B))}\text{abs}\ \ \mathbf{D}\ \ \partial A\quad x,y,z\ \Big|_{\partial(\mathbf{T}(B))}\qquad(13)$$

2.3.- Medidas de comparación morfológicas

Para medir las diferencias morfológicas un objeto se necesita un objeto de referencia, en este caso se utiliza una cabeza del cual se obtiene su campo de distancia para depués superponer otra cabeza y la superficie de la misma sea mapeado con una paleta de color. Los colores nos muestra que tan diferente es un objeto con respecto a otro y la gama de colores representa una escala de medición de que tan alejado está un puto de la superficie con el puto más cercano a la superficie de referencia.

2.3.1.- Diferencias simétricas de dos cabezas:

La diferencia simétrica de dos conjuntos se expresa como:

Sea el conjunto A y B

$$A \subseteq \quad v_1..,v_k,...,v_n\} \qquad (14)$$

$$B \subseteq \quad v_1..,v_k,...,v_n\} \qquad (15)$$

La diferencia simétrica es:

$$D = A \cup B - A \cap B \qquad (16)$$

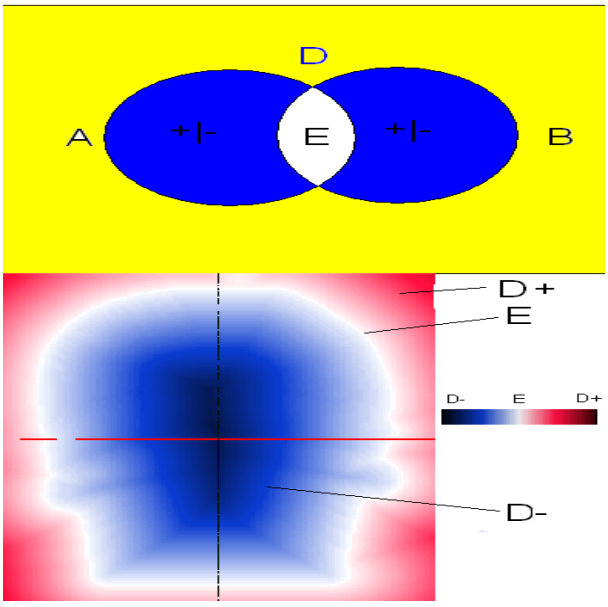

Figura 10: diagrama de Venn para diferencias simétricas de dos conjuntos en azul y la diferencia simétrica de un campo de distancia con un campo de distancia

La diferencia simétrica se usa entre dos objetos binarios, y al resultado se le obtiene su campo de distancia; este campo se puede visualizar de dos formas: como una proyección del campo en un corte transversal y la otra es visualizando la cabeza dentro de un campo de distancia y dependiendo del color representa la distancia que hay entre una superficie del modelo con la superficie de otro modelo que genera el campo de distancia en cuestión; como lo muestra la figura 20.

Si se realiza la suma cuadrática del campo de distancia diferencial en cada uno de sus componentes acotados se obtiene el error cuadratico de diferencias entre dos modelos y de la misma forma el error muestral cuadratico.

$$error^2 = \sum \frac{(\mathbf{D}_{\pm}(\partial \mathrm{A}) - \mathbf{D}_{\pm}(\partial B))^2}{N} \qquad (17)$$

$$error^2{}_{muestral}{}' = \sum \frac{(\mathbf{D}_{\pm}(\partial \mathrm{A}) - \mathbf{D}_{\pm}(\partial B))^2}{n-1} \qquad (18)$$

Si nuestra muestra del campo de distancia en cuestión, va a ser la intersección de la superficie a comparar con el campo de distancia del modelo de referencia obtenemos que el error cuadratico es:

$$error^2{}_{muestral}{}' = \sum \frac{(\mathbf{D}_{\pm}(\partial \mathrm{A}) \cap \partial B)^2}{n-1} \qquad (19)$$

Donde se puede observar esta intersección en la figura 20 con ayuda de una paleta de psudo-color mapeando el modelo dentro del campo de distancia $\mathbf{D}_{\pm}(\partial \mathrm{A})$.

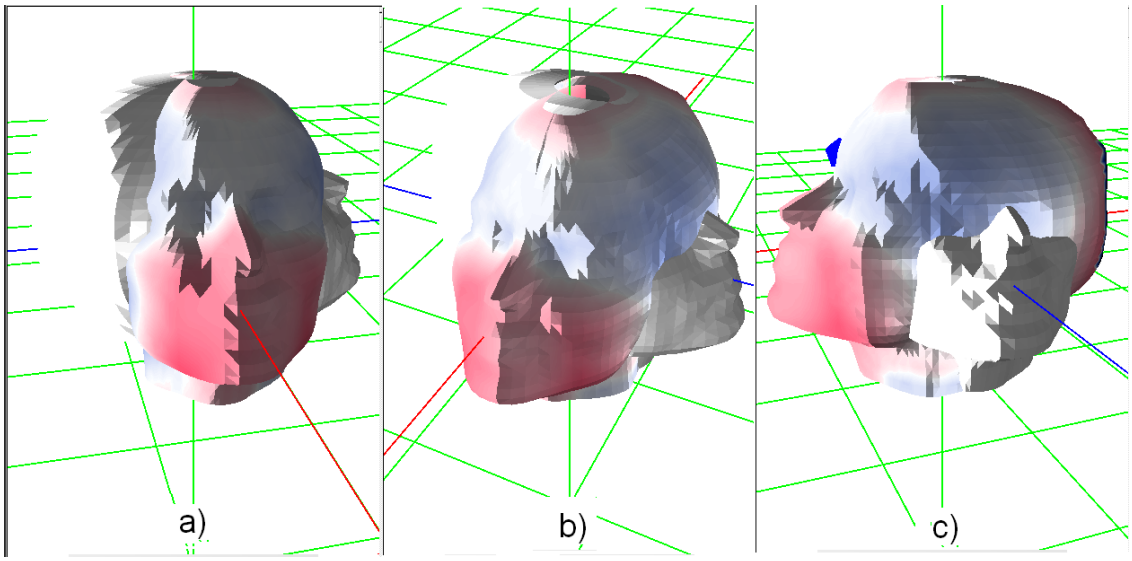

Figura 11: Modelo en diferentes alineaciones y superpuesto dentro de un campo de distancia

La diferencia simétrica entre dos objetos tri-dimensionales depende de su posición, porque puede que sean dos mismos modelos pero sin estar alineados y así aumentar la diferencia simétrica entre ambos. Para alinear los modelos en el laboratorio utilizamos el método de componentes principales para obtener una aproximación de una elipsoide y poder alinearlas con sus respectivos semi-ejes. Nuestro trabajo actual es la de visualizar dichas diferencias simétricas y el cual nos dará un error estadístico para mediciones.

El volumen de la diferencia simetrica de los dos objetos A, B, si están perfectamente alineados, sirve como una medida de la diferencia (o de similitud, si es pequeña) entre ambos, y sirve también como medida de error de alineación, al poder minimizarse durante un proceso de registro geométrico. Tenemos así dos aplicaciones de la visualización de la diferencia entre campos de distancia: (a) evaluar métodos de alineación (y de hecho como parte de un proceso iterativo de alineación) y (b) comparar dos cabezas de individuos diferentes, una vez que están alineadas.

2.4.- Métodos de visualización científica de morfometría

2.4.1.- Visualización de cabezas en un campo de distancia con una paleta de colores

La base de datos de las cabezas que se tienen están en formato VRML así que sólo se utiliza el arreglo de vértices y el de índices que se cargan en memoria para después mandar a dibujar la cabeza.

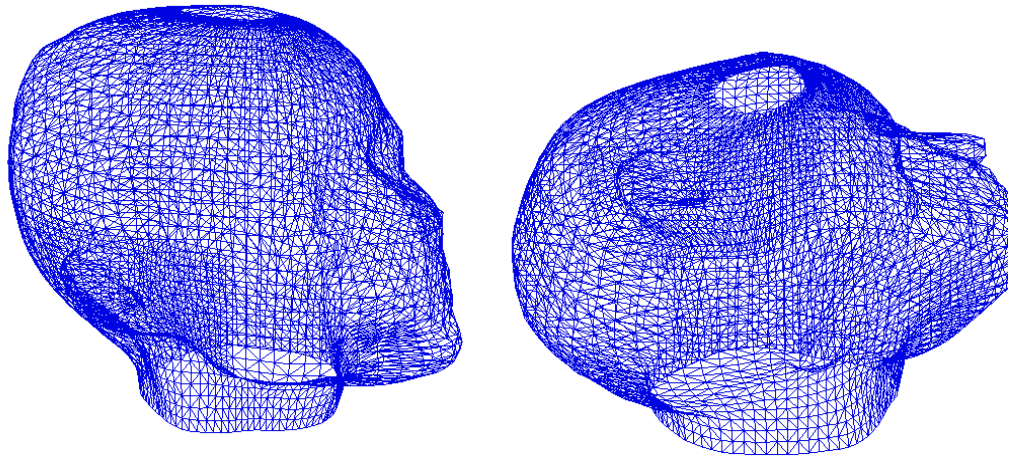

Figura 12: Modelo construido a base del archivo VRML

Ya después de visualizar las cabezas se texturizaron conforme a una paleta de colores, LUT (de *Look-Up Table*) o función de transferencia para *pseudo-color*, a partir del valor de distancia leido del campo EDT calculado para el objeto de referencia (con el que se compara). Para validar el método, se usa una esfera de referencia como objeto de control: el primer algoritmo de prueba es texturizar la cabeza dependiendo de la distancia de sus vértices con respecto a una esfera de radio *r* con centro en el origen. La esfera de referencia tendrá un radio aproximado a que cubra gran parte del phantom. [SIMONandDAVIGNON2009]

$x^2 + y^2 + z^2 = r^2$ *Ecuación de la esfera*

h_{error} : *Valor para el rango de error*

$D_{error}(x, y, z)$: *Campo a mapear*

$$D_{error}(x, y, z) = \begin{cases} x^2 + y^2 + z^2 \le (r + h_{error})^2 \\ x^2 + y^2 + z^2 \ge (r - h_{error})^2 \end{cases} \quad \forall\, x, y, z, r, h_{error} \in \Re \qquad (20)$$

Entonces, de la función anterior se necesita obtener otra función para mapear con una paleta de colores (LUT).

$$\bar{v} = x_v \hat{i} + y_v\, j + z_v k \qquad \textit{vector de posición de vértices}$$

$$f_{mapeo}(x_v, y_v, z_v) = \begin{cases} 0 & si \quad \dfrac{\sqrt{x_v^2 + y_v^2 + z_v^2} - r}{(r + h_{error})} + \dfrac{1}{2} < 0 \\ \dfrac{\sqrt{x_v^2 + y_v^2 + z_v^2} - r}{(r + h_{error})} + \dfrac{1}{2} & \\ 1 & si \quad \dfrac{\sqrt{x_v^2 + y_v^2 + z_v^2} - r}{(r + h_{error})} + \dfrac{1}{2} > 1 \end{cases} \qquad x_v, y_v, z_v \in \Re \qquad (21)$$

Del módulo del vector de posición de los vértices del phantom se resta el radio de la esfera que se está usando como referencia y es dividida entre la suma del radio de la esfera (se toma el radio de manera arbitraria para usarla como valor máximo) con la *h* de error y se hace esto para que los valores queden dentro del intervalo de la paleta de colores o sea entre 0 y 1. Se suma 0.5 al final por 2 razones

1.- Para evitar errores negativos que pueden ser mapeados
2.- Para desfasar el error 0 a 0.5 y quede a la mitad de la paleta de colores

En la figura se muestra el intervalo o gama de la textura de la paleta de colores y en la siguiente se muestra el mapeo para el error. Se muestra también, arriba de la primera una escala de grises de blanco a negro; notar que el valor de 0.5 de la escala en color corresponde a un gris medio (valor 128, en una escala de 0 a 255), en la paleta de grises:

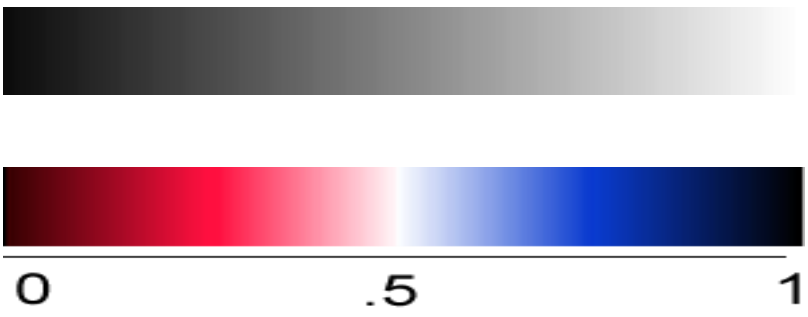

Figura 13: (Arriba) Escala de grises. (Abajo) escala en color como textura normalizada por OpenGL al intervalo [0,1].

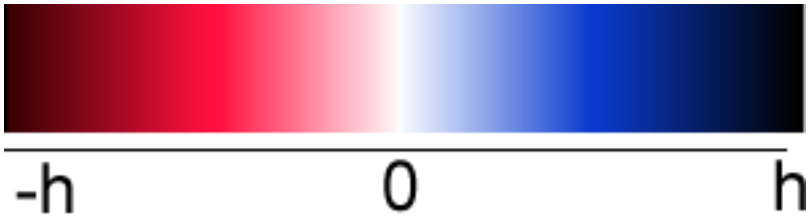

Figura 14: textura mapeada en "h" unidades de distancia con signo

Entonces al mandar a dibujar la cabeza y la esfera se puede ver que en blanco se visualiza la intersección de ambas (distancia Euclidiana cercana al valor 0.0) y los demás valores dependen de la distancia que hay entre la esfera de referencia con los vertices es el color tanto externos (distancia "positiva", en azules) como internos (distancia "negativa", en rojos), de acuerdo a la definición de la ecuación (2.1).

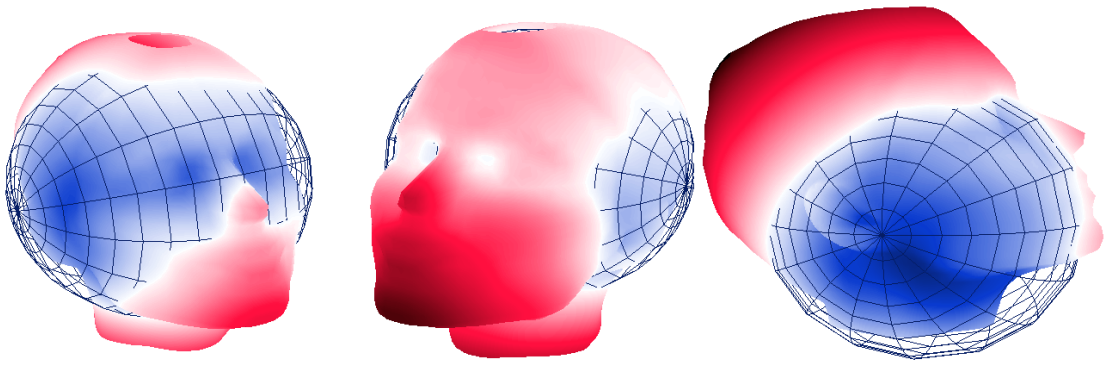

Figura 15: Representación gráfica de la diferencia simétrica utilizando una paleta de colores; de un modelo de cabeza con una esfera, renderizada en mallado transparente.

Usando MFC se colocó un "Slider Control" para aumentar o reducir el intervalo dinámico de error y con ello se obtiene un mayor contraste para el mapeo de color sobre la cabeza (Figura 20), permitiendo apreciar la distribución de distancias muy pequeñas. Notar que el negro, como valor de saturación de distancias mayores a la escala, tanto positivas como negativas, ayuda a visualizar mejor la zona cercana a la intersección.

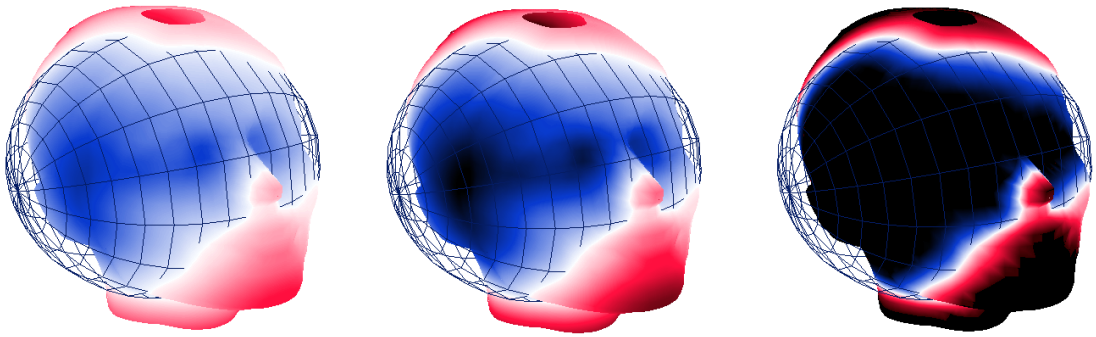

Figura 16: Resolución en las medidas de distancia utilizando diferentes contrastes

Con las ecuaciones anteriores se puede hacer una nueva ecuación más general para mapear otro tipo de mediciones y parámetros, no únicamente la distancia que hay entre los vértices de una figura con una esfera, Tal formulación quedaría como sigue:

$$f_{mapeo}\ \ g(x,y,z) = \begin{cases} 0 & si \quad \dfrac{g(x,y,z)}{\left|\max\ \ g(x,y,z)\right| + h_{rango}} + \dfrac{1}{2} < 0 \\ \dfrac{g(x,y,z)}{\left|\max\ \ g(x,y,z)\right| + h_{rango}} + \dfrac{1}{2} & \\ 1 & si \quad \dfrac{g(x,y,z)}{\left|\max\ \ g(x,y,z)\right| + h_{rango}} + \dfrac{1}{2} > 1 \end{cases} \tag{22}$$

$h_{rango} \in \Re$

$\mathrm{Im}(g(x,y,z)) \subset \Re$

Se utiliza el máximo valor que tiene la función *g* para no umbralizar vértices menores al máximo de *g* y con h se puede aumentar o reducir el contraste al mapear la figura. Con esta última ecuación solo queda definir a $g(x,y,z)$ que más adelante será el valor de la transformada de distancia Euclidiana, o una función de la misma (raíz, cuadrado o logaritmo, por ejemplo, para realzar ya sea valores pequeños o grandes). El código siguiente muestra como se programó el campo de distancia esférico y únicamente se manda a mapear los vertices antes de dibujarlos

```
bool CampoDeDistanciaEsferico(float x,float y,float z)
{
	float v_Modulo;
	float error;

	v_Modulo=sqrt(pow(x,2)+pow(y,2)+pow(z,2));
	error=(v_Modulo-r)/Rango;

	error=error+.5;

	if(error<0)	{ glTexCoord2f(0.1,0);	return true;}
	else if(error>1)	{ glTexCoord2f(0.1,1);	return true;}
	else		{ glTexCoord2f(0.1,error); return true;}
}
```

2.5.- Emulación de cómo visualizar un campo de distancia

Aunque se cuenta con un algoritmo muy eficiente para calcular EDTs, el tiempo de cálculo es muy elevado para su uso durante el desarrollo de las aplicaciones gráficas, que requieren muchas pruebas.

Se trató de emular cómo se vería un campo de distancia utilizando una paleta de colores y además que sea en objetos 3D y en tiempo real. Esto no es un campo de distancia como tal pero siguiendo el concepto de campo de distancia se emuló gráficamente como una aproximación razonable y sobretodo muy rápida. Para esto lo que se hizo fue tomar cualquier figura geométrica tridimencional y mandarla a dibujar un gran número de veces escalándola. Para mandar a dibujar el campo se hizo uso de una sucesión aritmética multiplicando cada término de la sucesión por una matriz unitaria 4 por 4 restandole una matriz de corrección para obtener así una matriz de escalamiento:

d_{gama} : *gama de propagación del campo*

$k_{resolución}$: *resolución*

$$s = i\frac{d_{gama}}{k_{resolucion}} \quad \forall\, i = 1,2,3,4,...,k_{resolucion} \mid i \in \square \ \wedge d_{gama} \in \square$$

$$I = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$ *Matriz identidad*

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$ *Matriz de corrección*

$$P_{4\times m} = \begin{pmatrix} x_0 & \cdots & x_m \\ y_0 & \cdots & y_m \\ z_0 & \cdots & z_m \\ 1 & \cdots & 1 \end{pmatrix}$$ *Matriz de vértices de la figura*

$$S_i = s(I - M) + M$$

$$C_i = S_i P_{4\times m}$$ *Conjunto de figuras emulando el campo*

Ahora la matriz C *i*-esima es una superficie que dista en una factor de $i\frac{d_{gama}}{k_{resolucion}}$ en forma radial a la superficie de la figura original.Mandando a dibujar todas las nuevas superficies obtiene una emulación del campo de distancia como lo muestra la figura

Figura 17: Aproximación de un corte transversal del campo de distancia

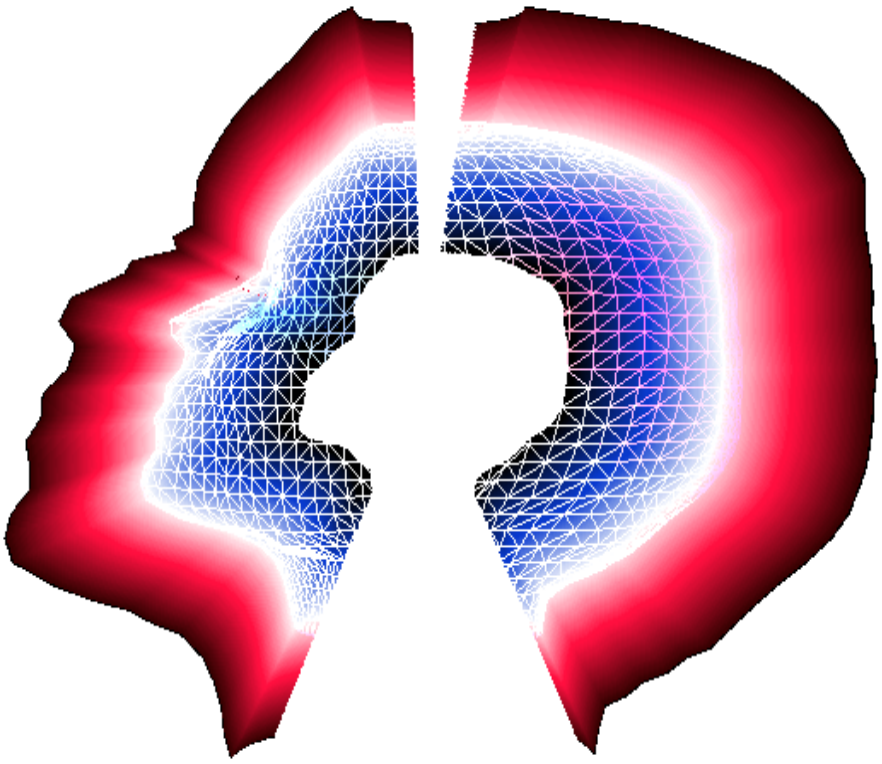

Figura 18: Aproximación de un corte transversal del campo de distancia con su respectivo modelo

El campo de distancia no se podría observar de esta manera porque la superficie más alejada hacia exterior taparía a las menos alejadas entonces para poder ver el campo se requiere hacer un corte transversal y dependiendo de la posición de la cámara, se hace dicho corte. Si se tiene la posición de la cámara y siempre está observando al origen, utilizamos ese vector, que es el vector de superficie de donde hacemos el corte además tenemos el vector de posición de cada vértice y deben ser operados de la siguiente forma:

$\bar{c} = c_x \hat{i} + c_y j + c_z k$ *Vector de posición de la cámara*

$\bar{v} = v_x \hat{i} + v_y j + v_z k$ *Vector de posición del vértice*

$\bar{p} = \bar{v} - \bar{c}$ *Vector de la cámara al vértice*

si:

$$\bar{c} \cdot \bar{p} = c_x p_x + c_y p_y + c_z p_z \quad (23)$$

$$\bar{c} \cdot \bar{p} = |\bar{c}||\bar{p}|\cos(\alpha) \quad (24)$$

y si la proyección del vector $\bar{p}$ *en* $\bar{c}$ *es*:

$$|\bar{p}|\cos(\alpha) \quad (25)$$

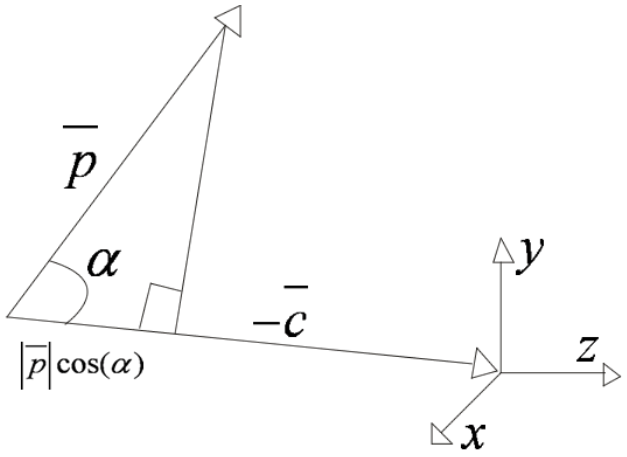

Figura 19: Gráfica de un plano de la región de corte

Entoces usando el producto punto se hacer un recorte entre 2 planos y son ello solo se manden a dibujar los vértices que pertenecen a ese intervalo entre los planos

$$f_{recorte}(\bar{p},\bar{c}) = \begin{cases} 1 & si \quad |\bar{c}| + t_{recorte} < \dfrac{|-\bar{c}\cdot\bar{p}|}{|\bar{c}|} \\ 0 & para\ otro\ caso \\ 1 & si \quad |\bar{c}| - t_{recorte} > \dfrac{|-\bar{c}\cdot\bar{p}|}{|\bar{c}|} \end{cases} \quad (26)$$

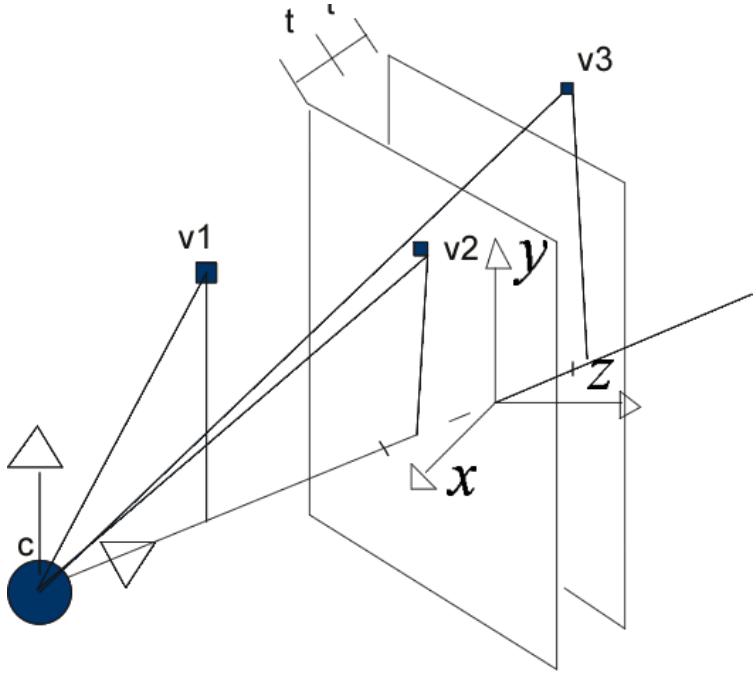

Figura 20: Modelo gráfico del algoritmo de recorte

La función resultante es una función booleana que sólo, devuelve valores verdaderos o falsos (0,1) con los cuales se va a determinar cuales vértices se mandan a dibujar y cuales no. Se muestra el código para hacer el recorte entre los dos planos.

```
V1[0]=((VertexArray[a][0]-cx)-Camara[0]/.1);
V1[1]=((VertexArray[a][1]-cy)-Camara[1]/.1);
V1[2]=((VertexArray[a][2]-cz)-Camara[2]/.1);

V2[0]=(VertexArray[b][0]-cx-Camara[0]/.1);
V2[1]=(VertexArray[b][1]-cy-Camara[1]/.1);
V2[2]=(VertexArray[b][2]-cz-Camara[2]/.1);

V3[0]=(VertexArray[c][0]-cx-Camara[0]/.1);
V3[1]=(VertexArray[c][1]-cy-Camara[1]/.1);
V3[2]=(VertexArray[c][2]-cz-Camara[2]/.1);

p1=(  V1[0]*Camara[0]*(-1)+
      V1[1]*Camara[1]*(-1)+
      V1[2]*Camara[2]*(-1))/modulo;

p2=(  V2[0]*Camara[0]*(-1)+
      V2[1]*Camara[1]*(-1)+
      V2[2]*Camara[2]*(-1))/modulo;
```

```
p3=(  V3[0]*Camara[0]*(-1)+
      V3[1]*Camara[1]*(-1)+
      V3[2]*Camara[2]*(-1))/modulo;


if(   p1>=modulo/.1-10 &&
      p2>=modulo/.1-10 &&
      p3>=modulo/.1-10 &&
      p1<=modulo/.1+10 &&
      p2<=modulo/.1+10 &&
      p3<=modulo/.1+10)
{
glVertex3f (VertexArray[a][0], VertexArray[a][1],
VertexArray[a][2]);
glVertex3f (VertexArray[b][0], VertexArray[b][1],
VertexArray[b][2]);

glVertex3f (VertexArray[c][0], VertexArray[c][1],
VertexArray[c][2]);

}
```

Todo esto da una idea de cómo se observa el campo de distancia de cualquier objeto y además se muestra en 3D y en tiempo real (el algoritmo exacto, aunque es muy eficiente, puede tardar más de media hora para una escena de 400×400×400).

2.6.- Alineación

2.6.1.- Análisis de componentes principales para alineación

El Análisis de Componentes Principales (ACP) en general es una técnica estadística de síntesis de la información, o reducción de la dimensión (número de variables). Es decir, ante un banco de datos con muchas variables, el objetivo será reducirlas a un menor número perdiendo la menor cantidad de información posible. Para ello, el resultado de aplicar ACP da eigenvalores en orden de importancia, y en una aplicación de reducción de información, se conservan aquellos cuya magnitud es mayor.

Los nuevos componentes principales o factores serán una combinación lineal de las variables originales, y además serán independientes entre sí. Un aspecto clave en ACP es la interpretación de los factores, ya que ésta no viene dada a priori, sino que se dedujo tras observar la relación de los factores con las variables iniciales.

El Análisis de Componentes Principales, en nuestro caso se utiliza para la alineación de dos modelos disminuyendo de esta forma el error simétrico. Obteniendo los eigenvalores y los eigenvectores de la distribución de vértices del modelo y estos representan los semiejes de un elipsoide ajustado al modelo de la cabeza. Por ejemplo se tiene las siguientes nubes de puntos como lo muestra la figura 30

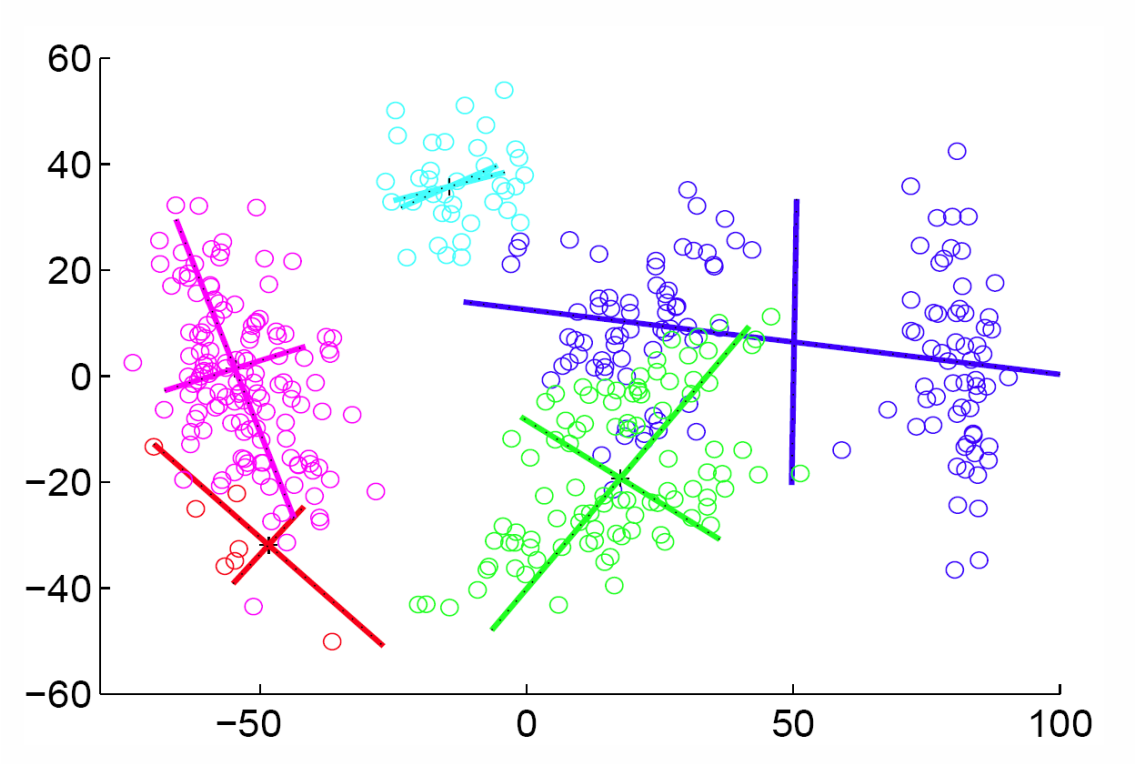

Figura 21: nubes de puntos y sus componentes principales

De la figura 30 se extraen una nube de puntos y se traza la elipsoide que representaría o ajustaría a la nube de puntos

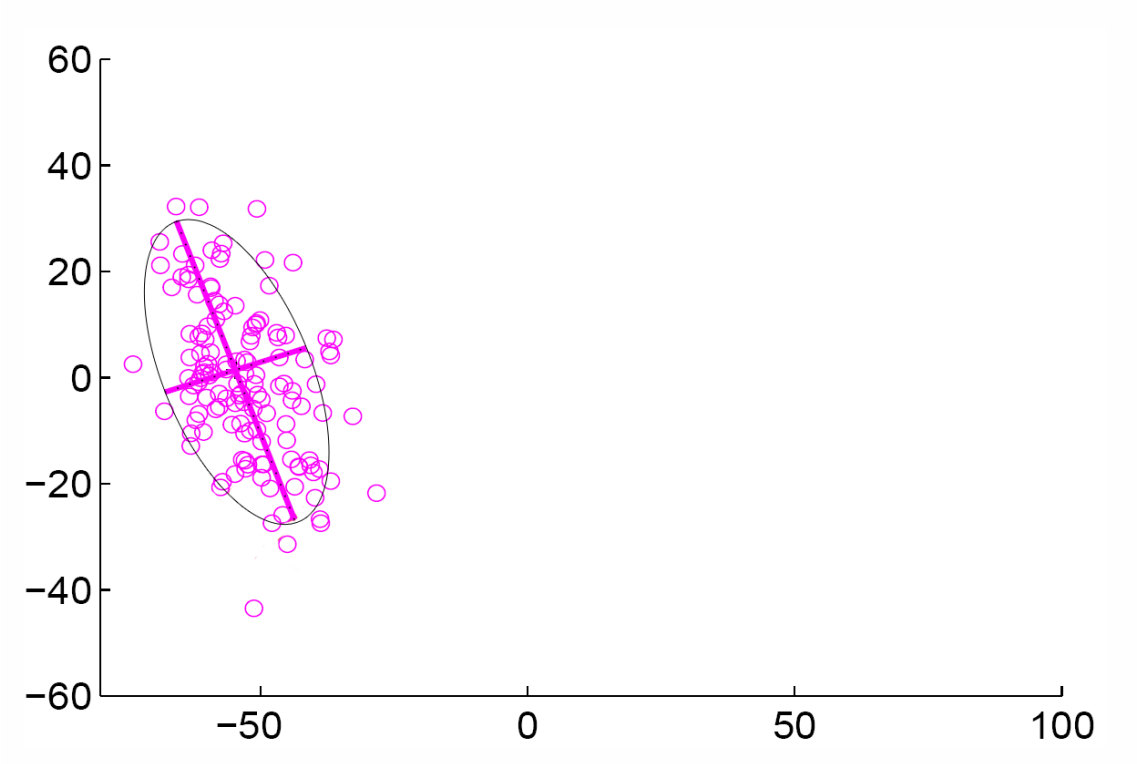

Figura 22

Se realiza lo mismo con el modelo de la cabeza con la diferencia de que será de 3 dimensiones y en vez de ser una elipse será una elipsoide como se muestra en la figura 32.

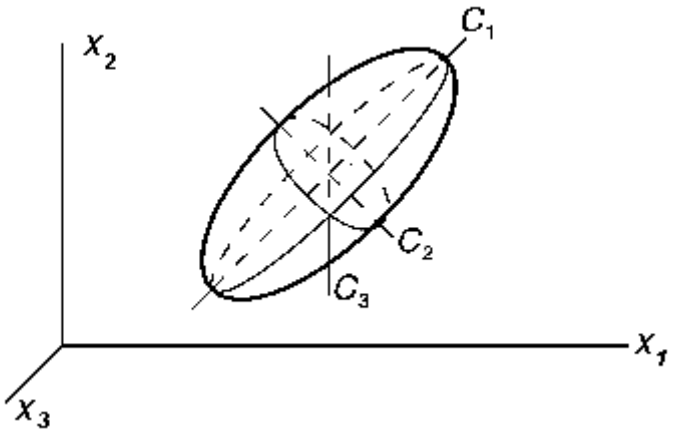

Figura 23: interpretación geométrica de las componentes principales

En la figura 33 se muestra la alineación entre la elipsoide y el modelo de la cabeza

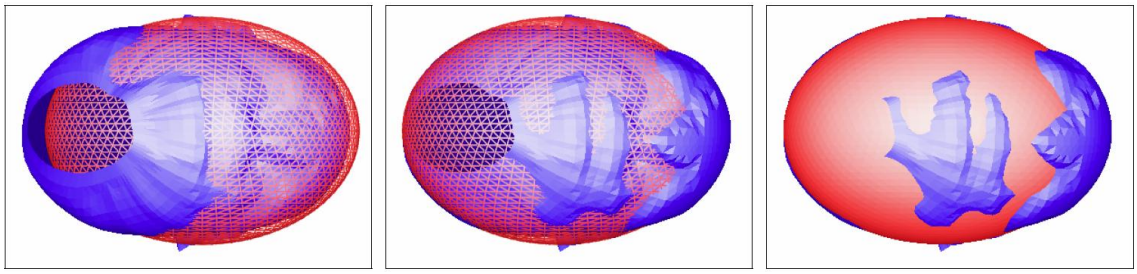

Figura 24: elipsoide que ajusta a un modelo de la cabeza humana. Figura extraída de [RAMIREZ2005].

Sea la siguiente matriz de vértices:

$$V = \begin{pmatrix} x_0 & \dots & x_n \\ y_0 & \dots & y_n \\ z_0 & \cdots & z_n \end{pmatrix}$$ Matriz de vértices

$$C = \begin{pmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \end{pmatrix}$$ Centroide

El centroide se puede representar como el punto medio de una nube de vértices

$$C = E\ V \qquad (27)$$

$$E\ V\ = V \cdot \frac{1}{n} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}_{1 \times n} \qquad (28)$$

Para aplicar el ACP necesitamos la matriz de covariancia de los vértices

$$\mathrm{cov}(x, y, z) = \begin{pmatrix} \mathrm{cov}(x,x) & \mathrm{cov}(x,y) & \mathrm{cov}(x,z) \\ \mathrm{cov}(y,x) & \mathrm{cov}(y,y) & \mathrm{cov}(y,z) \\ \mathrm{cov}(z,x) & \mathrm{cov}(z,y) & \mathrm{cov}(z,z) \end{pmatrix} \qquad (29)$$

La covarianza se define como:

$$\mathrm{cov}(x, y) = \sum_{i=0}^{n} \frac{(x_i - \bar{x})(y_i - \bar{y})}{n-1} \qquad (30)$$

$$Q = \begin{pmatrix} \bar{x} & 0 & 0 \\ 0 & \bar{y} & 0 \\ 0 & 0 & \bar{z} \end{pmatrix} \qquad (31)$$

$$M = Q\begin{pmatrix} 1 & \dots & 1 \\ 1 & \dots & 1 \\ 1 & \cdots & 1 \end{pmatrix} \qquad (33)$$

$$V - M = \begin{pmatrix} x_0 - \bar{x} & \dots & x_n - \bar{x} \\ y_0 - \bar{y} & \dots & y_n - \bar{y} \\ z_0 - \bar{z} & \cdots & z_n - \bar{z} \end{pmatrix} \qquad (34)$$

$$\text{cov}(x, y, z) = \frac{1}{n-1}(V - M)(V - M)^T \qquad (35)$$

Obtenemos los eigenvalores y los eigenvectores que serán las componentes principales tales que representarán los semiejes de la elipsoide que ajuste al modelo

$$\text{cov}(x, y, z)v = \lambda v \qquad (36)$$

$$(\text{cov}(x, y, z) - I\lambda)v = 0 \qquad (37)$$

$$\det(\text{cov}(x, y, z) - I\lambda) = 0 \qquad (38)$$