CCADET

CENTRO DE CIENCIAS APLICADAS Y DESARROLLO TECNOLÓGICO
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Informe Técnico
Sistema de Estéreo – Visión Múltiple para la Adquisición de Rostros Humanos

Alfonso Gastélum Strozzi Autores

Leticia Del Pilar López Tabares

Miguel A. Padilla Castañeda

Jorge Alberto Márquez Flores

Mayo 2015

Desarrollo

Financiamiento interno

Título: Sistema de Estéreo – Visión Múltiple para la Adquisición de Rostros Humanos

Autores: Alfonso Gastélum Strozzi, Leticia Del Pilar López Tabares, Miguel A. Padilla Castañeda y Jorge Alberto Márquez Flores

Afiliación: Centro de Ciencias Aplicadas y Desarrollo Tecnológico UNAM

Resumen

Entre los intereses del laboratorio de Análisis de Imágenes y Visualización tenemos el estudio de la forma del rostro humano y de la variabilidad que existe entre las diversas componentes de este entre diversos sujetos. Para llevar a cabo este estudio es necesaria la adquisición de la superficie tridimensional del rostro. Existen diversas técnicas para lograr esto, entre ellas escáneres laser, escáneres de luz estructurada y escáneres de estéreo-visión. Desde el punto de vista del diseño existen condiciones que debemos de cumplir, la primera es la adquisición del rostro de oreja a oreja (180º grados), la segunda es que la adquisición debe realizarse en el menor tiempo posible ya que el movimiento del sujeto arruinaría el registro de la posición y, tercero tenemos que construir un sistema con un diseño de calidad/costo escalable. Tomando en cuenta estos puntos decidimos construir un sistema de seis cámaras que forman tres pares de estéreo-visión capaces de adquirir el rostro humano de oreja a oreja. Para lidiar con el tiempo de adquisición trabajamos con cámaras con capacidad de sesenta cuadros por segundo y las sincronizamos utilizando un generador de pulsos externo. La ventaja del sistema también radica en que los algoritmos y hardware desarrollados para controlarlo pueden utilizarse en otros modelos de cámaras de mejor calidad, haciendo nuestro diseño escalable, de ser requerida una mayor resolución espacial. El sistema construido es capaz de adquirir el rostro completo utilizando tres puntos de vista diferentes, a 60 cuadros por segundo, con una resolución de 640 x 480 pixeles en estéreo por par de cámaras. El generador de ondas basado en un microcontrolador Arduino externo permite adquirir seis imágenes al mismo tiempo y un algoritmo de control en lenguaje C++ utiliza un sistema de buffers en cadena (memoria de trabajo dinámica) para procesamiento en tiempo real.

Índice

Nomenclatura

Símbolo (orden alfabético)	*Significado*
f	*Distancia focal*
O	*Centro de la camara*
P	Matrices de proyección
E	Matriz esencial
M	Matriz Parámetros intrínsecos
F	Matriz fundamental
kc	Distorsión
d	Disparidad
l	Landmark
$indice$	Índices antropometricos
d_k	Distancia entre dos landmarks
TEM	Error técnico de medición

Introducción

El trabajo presentado consiste en el desarrollo de un sistema para la adquisición de superficie faciales que permitan la construcción de modelos de rostros humanos en tres dimensiones (en adelante 3D). A continuación se reproducen actualizadas la investigación bibliográfica realizada en la tesis (Pilar, 2015) donde se presentan las necesidades en el área de la antropología por obtener datos de poblaciones y la posibilidad de hacerlo por métodos indirectos como los de visión por computadora.

La antropometría facial consiste en estudiar las dimensiones de las diferentes estructuras anatómicas como la nariz, boca, orejas, labios y ojos, así como sus proporciones relativas. Dichas mediciones proveen métodos para describir objetivamente el fenotipo de un sujeto y una lista individual de anomalías, además de ser útiles para definir una métrica en la detección de nuevos síndromes de acuerdo a (Farkas et al, 1996).

El estudio clásico de la antropometría se hace de manera directa con ayuda de herramientas táctiles, siendo actualmente este es el método preferido por los antropólogos, ya que permite palpar el rostro en la búsqueda de los puntos somatométricos, que en algunos de los casos están ocultos por el tejido suave y bajo el cabello. A pesar de que requiere un largo tiempo para la examinación y personal entrenado, sigue siendo el método comúnmente usado en la práctica clínica.

Además de este método se han utilizado métodos de imagenología, como fotografías y radiografías que proveen información limitada a una proyección de interés. Las fotografías son métodos indirectos, no invasivos y de bajo costo para la evaluación de tejido suave, sin embargo se utilizan más como algo ilustrativo y pocos son los que las utilizan como técnica de medición (Sforza et al, 2006). Sin embargo, es por esto que se han desarrollado métodos para crear y analizar modelos computacionales en 3D que permitan una manipulación flexible y precisa; las condiciones deseadas para estos métodos es que sean no invasivos, rápidos, lo más simples posible, con instrumentos de bajo costo y que permitan la creación de bases de datos digitales.

Entre los métodos recientes encontramos la Tomografía Computarizada (o CT, por sus siglas en inglés), sobre todo para la planeación quirúrgica (Enciso et al, 2003). Sin embargo, no contiene información precisa sobre los tejidos suaves del rostro, es costosa y debido a la dosis de radiación de rayos X necesaria para obtener cientos de cortes transversales, es una técnica invasiva y riesgosa, por lo que no es considerada como un método a elegir en casos de valoración antropológica.

Las imágenes de Resonancia Magnética no son invasivas pero su costo y tiempo de obtención son aún más elevados que los métodos de CT. El escáner láser facial que registra la textura del rostro utilizando láser y técnicas de luz visible, requiere que el sujeto permanezca quieto durante varios segundos o minutos mientras el escáner recorre el rostro. La estéreo fotogrametría digital es un método con visión estereoscópica, con un algoritmo de estéreo-reconstrucción que permite medir posiciones de rasgos, sus tamaños y las distancias entre ellos (Galantucci et al, 2009). Hay también sistemas de imágenes digitales de luz visible, que utilizan un sistema de mapas de profundidad creado por estéreo, fotogrametría y cámaras de luz estructurada.

Se han realizado estudios utilizando los diferentes métodos mencionados, obteniendo resultados similares a los obtenidos por antropometría directa, ya sea realizando un marcaje previo de los puntos somatométricos o no. (Wong et al, 2008) encuentra que 17 de las 18 medidas realizadas entre puntos somatométricos (en inglés landmarks) son precisas comparadas con la antropometría directa, no existe una diferencia estadística en 15 de éstas, por lo que Wong concluye que la antropometría digital es excelente y comparable a la antropometría directa. Utilizando imágenes digitales fotogramétricas en 3D, obtienen una precisión de 1 mm, concluyendo que la confiabilidad de las medidas en 3D es mejor o al menos comparable con la antropometría directa.

(Kook et al, 2014) comparan 5 métodos de análisis: cámara estereoscópica 3D, CT 3D, escáner láser, antropometría directa y un digitalizador, localizando 15 landmarks por el mismo operador, concluyendo que todos los métodos presentan alta precisión, error menor a 0.9 mm, y reproducibilidad para el uso clínico.

(Ozsoy et al, 2009) comparan fotografías 2D, antropometría directa y un digitalizador, concluyendo que las medidas en imágenes digitales minimizan los errores y consumen menos tiempo que los otros dos métodos.

(Metzler et al, 2013) utilizan un equipo de fotogrametría con dos observadores y marcando los landmarks en un maniquí y medidas directas, no encontraron diferencias significativas y el error inter-observador no difiere significativamente, excluyendo los errores sistemáticos cometidos; además el error entre calibraciones no es significativo, recomendando que se tomen varias imágenes del sujeto debido al movimiento que pueda presentar. Metzler y compañía, concluyen que los errores presentados no son clínicamente relevantes y los errores sistemáticos son insignificantes y pueden descartarse.

(Bush et al, 1996) utilizando un escáner láser concluye que las desviaciones estándar en las medidas digitales son menores a 0.5 mm, de modo que los landmarks de tejido suave pueden ser localizados de manera confiable.

(Menezes et al, 2009) marcaron los landmarks previamente y utilizando tres fotografías (una frontal y dos laterales), evaluaron el error aleatorio en las mediciones y ángulos entre landmarks, encontrando diferencias de 3 mm y 3 grados con métodos reproducibles, por lo que se considera que las coordenadas de los landmarks se obtienen con precisión y son adecuadas para obtener medidas confiables. Los errores de localización de landmarks menores a 0.5 mm son despreciables (Ghoddousi et al 2007), y en mediciones menores a 1 mm son aceptables clínicamente (Mendanca et al 2013).

Finalmente, (Woodward et al, 2012) reportan un sistema que es el antecedente del que nosotros diseñamos y construimos; su objetivo fue la reconstrucción facial y el modelado de expresiones. Los algoritmos desarrollados durante este trabajo fueron compartidos con la UNAM en una colaboración de desarrollo establecida con la Universidad de Auckland

Los sistemas de adquisición digital permiten el estudio de las propiedades de forma faciales pero además de esto pueden ser utilizados en aplicaciones novedosas que usan realidad virtual y aumentada, para las cuáles se generan "avatares" o modelos dinámicos de personas reales que requieren de información antropométrica precisa y representativa de una o más poblaciones (Sifakis et al, 2006).

El objetivo de este trabajo fue el desarrollo de un sistema de visión por computadora y los algoritmos necesarios para la obtención de superficies facial de oreja a oreja a 60 cuadros por segundo.

El sistema fue probado adquiriendo superficies faciales de una población sana para su análisis. Para lograr este objetivo se desarrollaron los siguientes pasos en el laboratorio de Análisis de Imágenes y Visualización del CCADET-UNAM:

- Diseñar un sistema de estereovisión compuesto por 6 cámaras que permitiera la adquisición de imágenes frontal y laterales para posteriormente crear un modelo 3D del rostro.

- Implementación computacional de algunos algoritmos e integración de otros, proporcionados por la U de Auckland o bien de dominio público o inclusive comercial.

- Aplicar en una muestra de pacientes sanos el sistema de adquisición elaborado.

- Obtener valores promedios y examinar cuáles parámetros, derivados de los rasgos, son los que presentan mayor y menor variación.

1 Sistema de visión-estéreo múltiple

El sistema de visión-estéreo múltiple desarrollado debe, a la distancia donde se encuentre el objeto de interés, ser capaz de obtener un campo de adquisición de 180 grados por cuadro de adquisición. Para esto es necesario construir un sistema con 3 puntos de vista. En la Figura 1 se muestra un esquema de la configuración inicial.

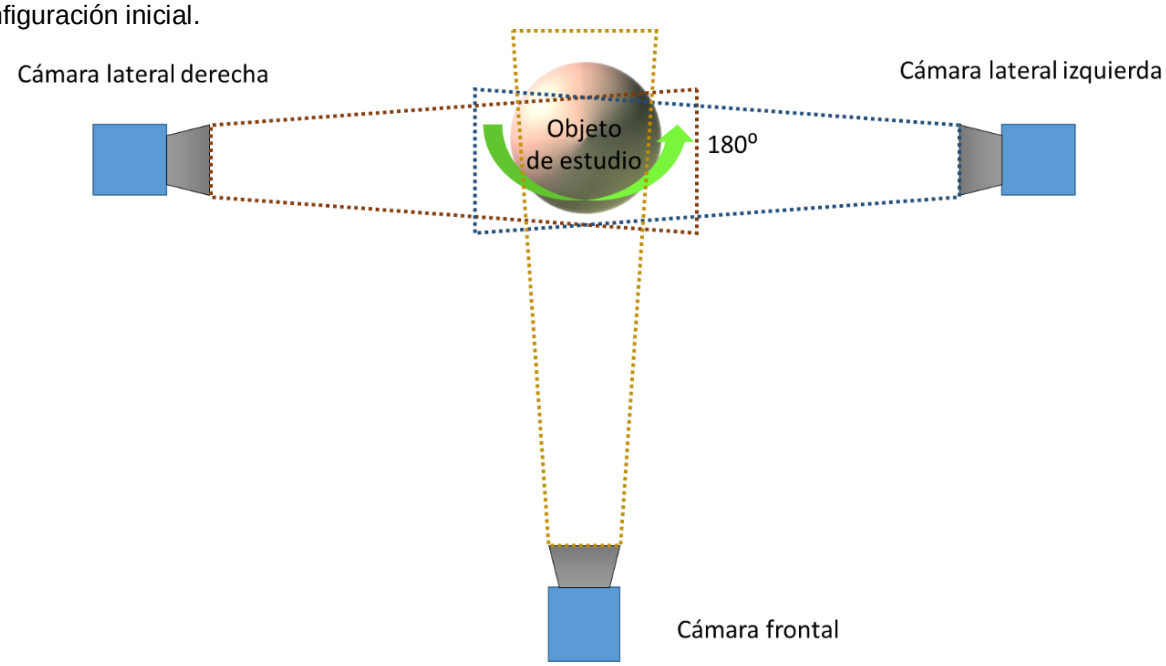

Figura 1. El sistema de adquisición está formado por tres puntos de vista que permiten la adquisición de un campo de visión de 180°.

Cada uno de estos puntos de vista debe estar compuesto por dos cámaras, con las cuales se puede obtener el campo de profundidad por cada punto de vista. El objetivo de este arreglo es adquirir una nube de puntos final del rostro humano de oreja a oreja, construida a partir de la superposición de las tres tomas (Figura 2).

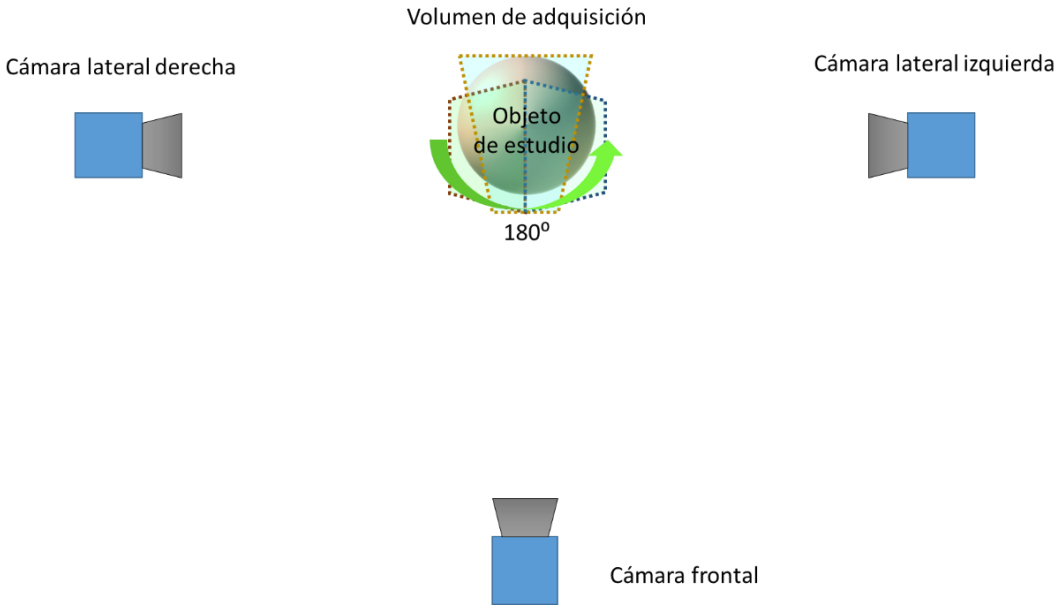

Figura 2. El sistema de adquisición está formado por tres puntos de vista que permiten la adquisición de un campo de visión de 180°.

Para el desarrollo del sistema prototipo se seleccionó la cámara PS3-EYE, la cual utiliza el CMOS modelo Omnivision OV7720/OV7721. Éstas son cámaras comerciales de bajo costo que tienen la capacidad de grabar video VGA a 60 cuadros por segundo y que pueden ser modificadas para ser sincronizadas por hardware.

1.1 Propiedades de la cámara PS3-EYE

Las cámaras utilizadas para construir el sistema estéreo son las PS3-EYE de la marca SONY®, las cuales tienen las siguientes características de fábrica:

- Resolución de 640 x 480 pixeles a 60 Hz
- Resolución de 320 x 240 pixeles a 120 Hz
- Distancia focal fija
- 56° de campo visual
- 75° de campo visual

El sistema final es funcional en cualquiera de las opciones, pero para la adquisición de rostros en este trabajo, se seleccionó la resolución de 640 x 480 pixeles a 60 Hz con un campo visual de 75°. En la Figura 3 se muestra una imagen de la cámara utilizada.

Figura 3. Cámara PS3-EYE de la compañía SONY®.

El sensor interno de la cámara PS3-EYE es un CMOS marca Omnivision® modelo OV7720/Ov7721 (Omnivision, 2006). Este modelo de sensor tiene una entrada de generación de tiempo de video de nombre FSIN el cual permite la sincronización de la toma de video utilizando un generador de pulsos externos, como se muestra en la Figura 4.

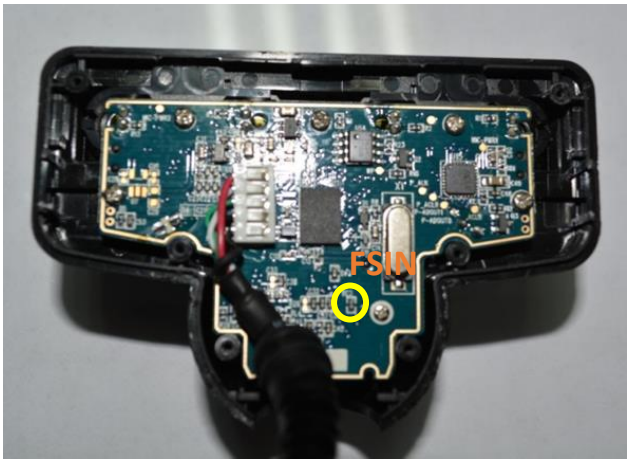

Figura 4. Pin de control FSIN. Este Pin permite el control de la adquisición de la cámara utilizando una señal externa.

Para reducir la línea base entre las cámaras que formaran el par estéreo, tanto los sensores CMOS junto con sus lentes se desmontaron de la caja original y se adaptaron a una nueva caja de embalaje diseñado en el software OpenSCAD (Kintel, 2014). El diseño del embalaje se muestra en la Figura 5 y el código para su construcción se puede ver en el Anexo A. Además de la reducción de la distancia entre los centros de las cámaras, la nueva caja se diseñó con límites planos para permitir la unión de las dos cámaras.

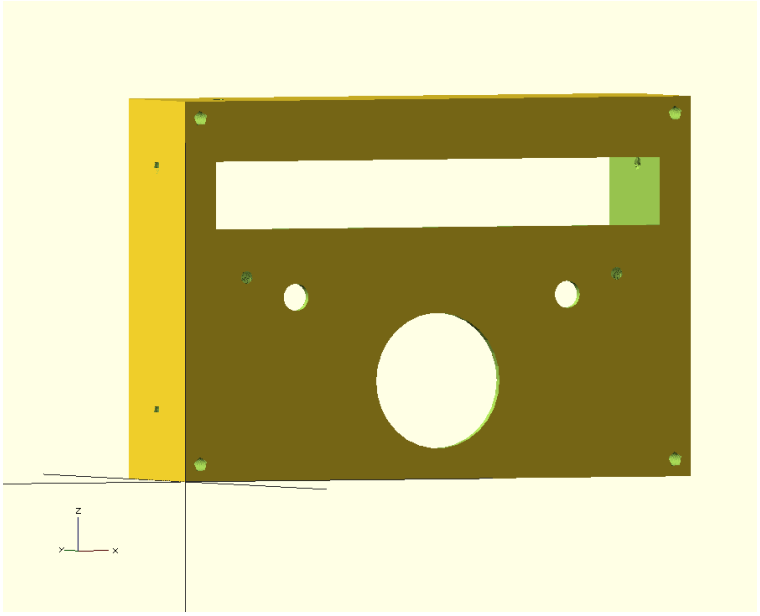

Figura 5. Modelo volumétrico de la caja que contendrá el sensor CMOS. El código de OpenSCAD se incluye en el apéndice.

El modelo se maquinó utilizando impresión 3D mediante una máquina de manufactura aditiva, con material PLA. En la Figura 6 se muestra una cámara en la nueva caja.

Figura 6. Imagen de la forma final de la caja con el CMOS y la lente montados en ella.

1.2 Sistema estéreo múltiple.

El sistema de estéreo múltiple está formado por seis cámaras, con las cuales podemos definir tres puntos de vista estéreo, donde cada punto de vista está formado por un par de cámaras. La configuración de las cámaras arriba/abajo, como se ve en la Figura 7, se debe a que el rostro humano tiene una alta simetría si lo estudiamos a partir de la parte derecha e izquierda de la línea media (Figura 8).

Figura 7. Arreglo de un par de cámaras para la obtención del mapa de profundidad. Para reducir la distancia entre las cámaras una de ellas se gira 180º.

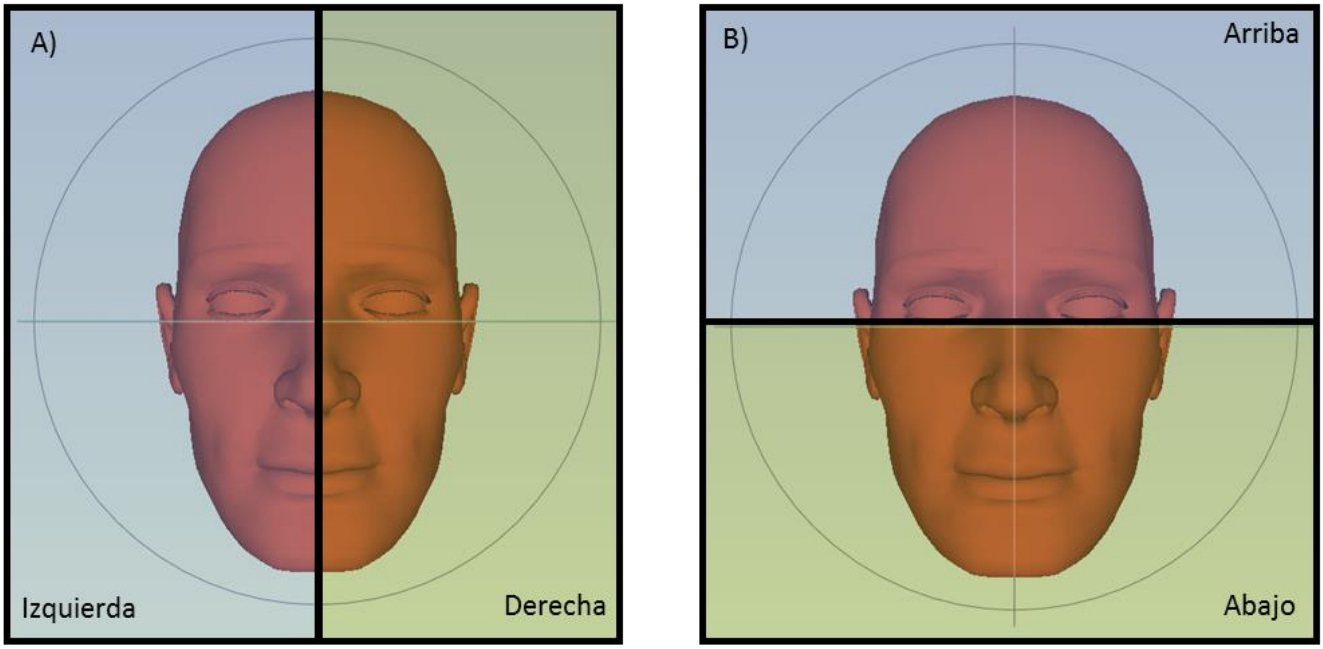

Figura 8. Los métodos de estéreo visión pueden tener problemas en encontrar la solución correcta cuando se trabaja con objetos simétricos. A) Configuración Derecha/Izquierda la simetría del rostro en esta configuración es alta. B) Configuración Arriba/Abajo

La configuración final de los tres pares de cámaras se muestra en la Figura 9. Las cámaras se montan en un sistema rígido que permite mantener la relación de posición entre los pares de cámaras obtenidas con la calibración del sistema.

Figura 9. Sistema de estéreo visión múltiple. La relación espacial entre las cámaras se mantiene con ayuda de un sistema rígido.

Un problema a resolver en los sistemas estéreos es la sincronización de la toma de cuadros: trabajando a 60 Hz no fue posible obtener una solución donde la sincronización por software de cámaras generara cuadros perfectamente alineados en el tiempo y fue por esto que se decidió trabajar con cámaras que permitieran la sincronización por hardware.

La sincronización por hardware se hace utilizando el pin FSIN el cual es controlado por una señal externa de modulación por ancho de pulsos (PWM). La PWM se realiza con un controlador Arduino One, que provee una PWM de 8-bit. Con esta señal se sincronizan las seis cámaras para que tomen cuadros de imágenes de manera simultánea, con una velocidad de 60 cuadros por segundo. Con esta configuración es posible sincronizar las seis cámaras a 60Hz y el algoritmo de adquisición se encarga de capturar los cuadros obtenidos de las CMOS a imágenes. El código de Arduino se presenta en el Anexo B.

El sistema de adquisición final se muestra en la Figura 10, las cámaras están posicionadas en forma de U, un par está al frente del sujeto y los otros dos pares al extremo lateral, uno frente al otro. El cosigo de control del sistema de cámaras se muestra en el Anexo C.

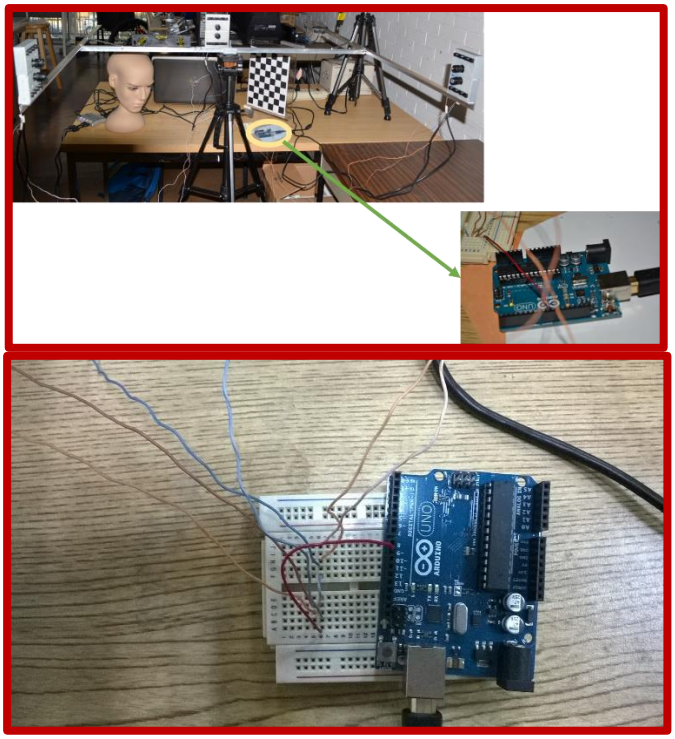

Figura 10. Sistema de estéreo visión múltiple. La relación espacial entre las cámaras se mantiene con ayuda de un soporte rígido.

1.3 Sistema de iluminación.

Terminado el sistema de adquisición se prosiguió a diseñar la iluminación apropiada que permitiera obtener la mejor posible representación de la superficie del rostro.

La homogeneidad en los tonos de la piel y los cambios de las zonas de iluminación durante la adquisición del rostro en condiciones de luz ambiental hace necesario el diseño de un sistema de iluminación que mejore el resultado obtenido.

Para aumentar la textura del rostro y reducir la homogeneidad, se utilizan un par de proyectores LCD para crear un patrón de color sobre el rostro, reduciendo con esto los errores del algoritmo de estéreo - visión. Los proyectores utilizados son el BenQ modelo MP515ST y el BenQ, modelo MP670XGA. En la Figura 11 se muestra el patrón utilizado para mejorar la textura del rostro.

Figura 11. Patrón de líneas proyectado sobre el rostro.

Para reducir las sombras y puntos brillantes en el rostro del sujeto se utilizan tres fuentes de iluminación difusa, con esto se buscó tener una iluminación continua sobre todo el rostro y evitar que se produzcan puntos calientes o sombras sobre el rostro. En la Figura 12 se muestra el sistema completo de adquisición junto con las fuentes de iluminación.

Figura 12. Sistema de estéreo visión múltiple junto a la iluminación externa utilizada.

2 Descripción matemática del modelo de estéreo – visión

La estereovisión se basa en la correspondencia de puntos en dos imágenes 2D que permitan estimar la profundidad de los objetos a partir de estas (Prince, 2012).

Esta describe las cámaras utilizando el modelo de cámara pinhole, cuya descripción geométrica se muestra en la Figura 13. En este modelo la imagen en el plano de imagen I es formada debido a una proyección de perspectiva formada por los rayos emitidos o reflejados por el objeto. Estos rayos entran por el centro óptico C el cual está a una distancia focal f del plano de la imagen I.

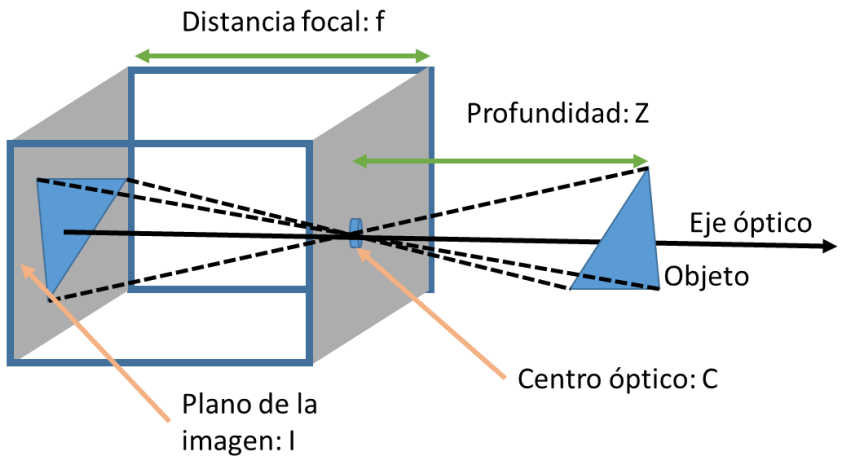

Figura 13. Modelo de la cámara de pinhole.

En la Figura 14 se muestra un esquema simplificado de las relaciones espaciales entre la posición del objeto en el plano de la imagen y su posición en el espacio.

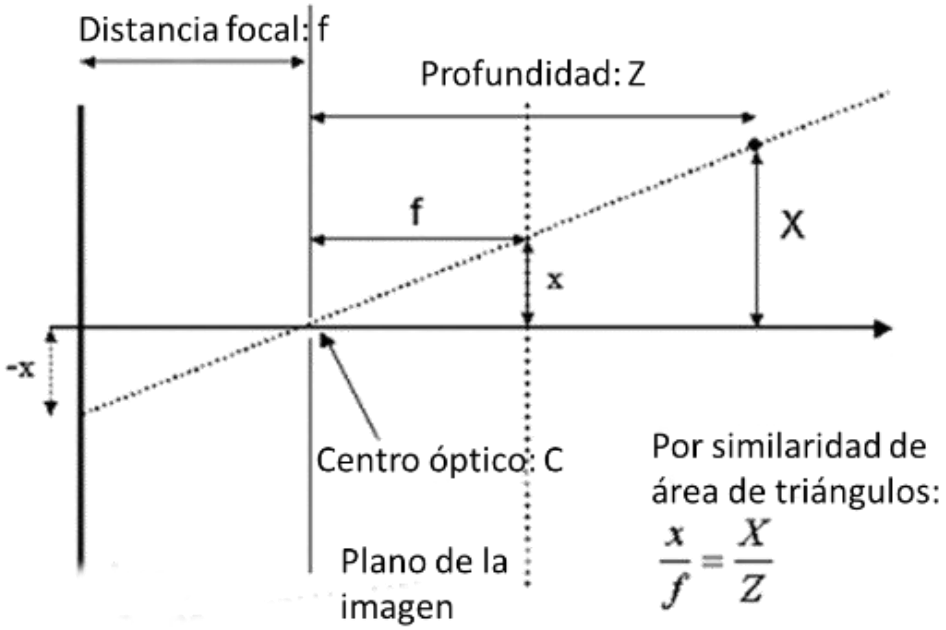

Figura 14. La relación entre la coordenada x de la proyección del objeto en el plano de la imagen y su posición en el espacio X está dada por la relación entre f y Z.

En la Figura 14 se puede observar que cualquier punto X′ que se encuentre en el mismo rayo dado por las coordenadas homogéneas de X dará como resultado la misma proyección x en el plano de la imagen. Así con solo un plano de la imagen y un solo rayo no es posible diferenciar entre Z y Z′. En las Ecuaciones 1 y 2 se muestran las relaciones entre x, X, y y Y.

$$\frac{x}{f} = \frac{\mathrm{X}}{Z} \tag{1}$$

$$\frac{y}{f} = \frac{Y}{Z} \tag{2}$$

Así podemos tener la relación lineal de las ecuaciones como una matriz de proyección:

$$\begin{bmatrix} U \\ V \\ S \end{bmatrix} = \begin{bmatrix} fX \\ fY \\ fZ \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3}$$

De la Ecuación 3 podemos definir a la matriz $\boldsymbol{P}$, una matriz de 3×4, llamada matriz de proyección en perspectiva.

$$\boldsymbol{P} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{4}$$

Para poder encontrar la profundidad a la que se encuentra un punto es necesario definir una nueva geometría basada en dos puntos de vista (Figura 15). Para ello se define una nueva geometría epipolar que permite relacionar el centro de la cámara de referencia O_R con la cámara objetivo O_T. Siguiendo la nomenclatura usual en la literatura, el desarrollo de las ecuaciones que definen la geometría epipolar se hará con respecto a una cámara izquierda L y una derecha R.

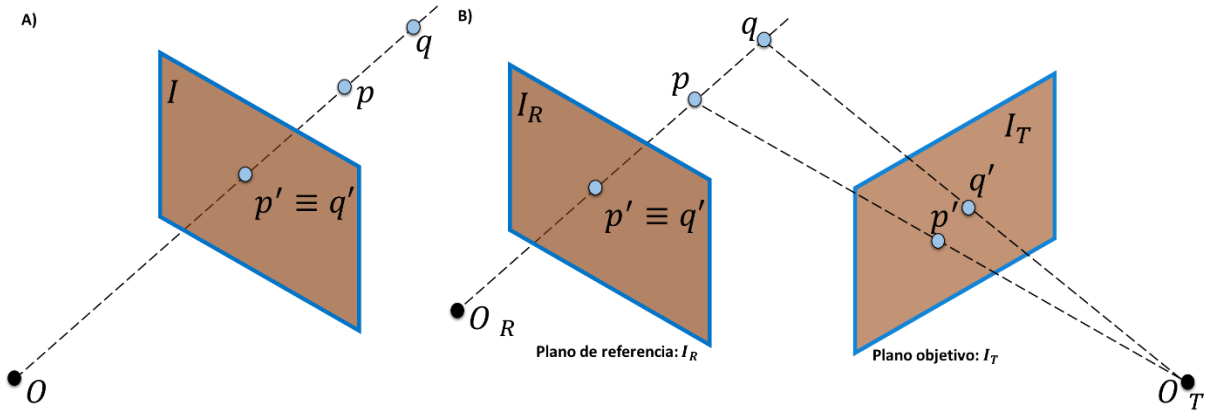

Figura 15. A) Con un solo punto de vista los puntos proyectados por p y q son iguales $p' \equiv q'$. B) En el Segundo punto de vista los puntos p y q proyectan a dos puntos diferentes p' y q'.

En la Figura 15 se puede ver que cuando se tienen dos cámaras se pueden definir una serie de relaciones geométricas entre el punto en el espacio real 3D y sus proyecciones a los dos planos de imagen en 2D. Estas relaciones se establecen siguiendo el modelo de la cámara de pinhole. En la Figura 16 se describen las principales relaciones geométricas entre las dos cámaras.

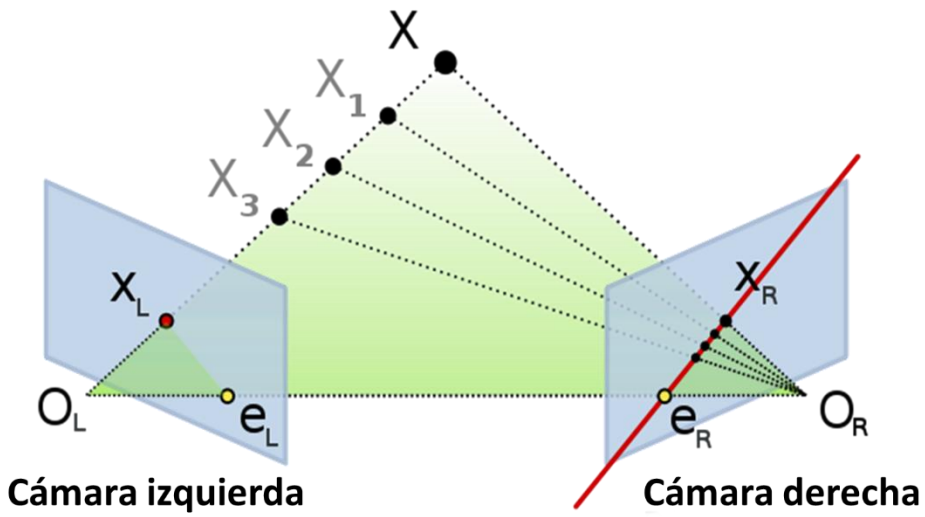

Figura 16. Relaciones geométricas que definen un plano epipolar (triangulo verde).

Donde O_L y O_R son los centros de las camaras izquierda y derecha respectivamente. e_L es el epipolo izquierdo el cual se define como la proyección del centro O_R en el plano de imagen de la cámara izquierda. e_R es el epipolo derecho el cual se define como la proyección del centro O_L en el plano de imagen de la cámara derecha. El plano epipolar se define con los puntos X, O_L y O_R. Una línea epipolar se define como la intersección entre el plano epipolar con el plano de la imagen.

La línea formada entre los puntos $O_L - X$ los ve la cámara izquierda como un punto, pero la cámara derecha lo ve como una línea. Esta línea sería una línea epipolar $(e_R - x_R)$ definida en el plano de la imagen de la cámara derecha.

Si se tiene el punto de proyección X_L entonces la línea epipolar $(e_R - x_R)$ es conocida. Por otro lado el punto real X proyectará en la cámara derecha en algún punto x'_R que estará contenido en la línea epipolar $e_R - x_R$.

Si podemos conocer ambos puntos X_L y X_R, sus líneas de proyección también son conocidas en su punto de intersección X. Así con una geometría epipolar simplificada, donde las dos cámaras se encuentren en el mismo plano, los algoritmos de estéreo – visión, desarrollados para encontrar correspondencias entre las dos imágenes, pueden idealmente reducir su búsqueda a un problema unidimensional.

Dada esta geometría podemos relacionar las matrices de proyección de la cámara derecha e izquierda, P_L y P_R ya que podemos definir dos transformaciones que relacionen sus centros. Las cámaras están relacionadas por un vector de translación $T = (O_R - O_L)$ y una matriz de rotación R. Así tenemos que:

$$P_R = R(P_L - T) \tag{5}$$

Y cualquier punto real p puede ser definido con respecto a las matrices de proyección de ambas cámaras como:

$$p_R = \frac{f_R}{z_R} P_R \tag{6}$$

$$p_L = \frac{f_L}{Z_L} P_L \tag{7}$$

El siguiente punto a desarrollar es la estimación del mapeo de los puntos a líneas para un par de imágenes en estéreo. Si tenemos un punto en el espacio p podemos definir una relación coplanaria entre los vectores P_R, P_L y T, donde de la ecuación 5 tenemos que de P_R podemos definir la relación $R^T P_R = (P_L - T)$. Estas relaciones se muestran en la Figura 17.

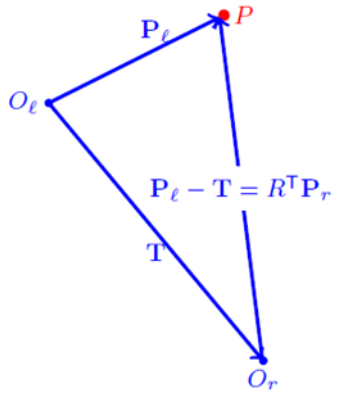

Figura 17. Relación entre los puntos que forman el plano dado por O_L, O_R y p.

Tenemos que cualquier vector $z = x \times y$ es ortogonal a x y a y: $x^T z = y^T z = 0$. Así, podemos definir las siguientes relaciones obtenidas del plano descrito en la Figura 17.

$$\begin{aligned}
& (P_L - T)^T (T \times P_L) = 0 \\
\Rightarrow \quad & (R^T P_R)^T (T \times P_L) = 0 \\
\Rightarrow \quad & P_R^T R (T \times P_L) = 0 \\
\Rightarrow \quad & P_R^T R \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} P_L = 0 \\
\Rightarrow \quad & P_R^T (RS) P_L = 0 \\
\Rightarrow \quad & P_R^T E P_L = 0
\end{aligned} \tag{8}$$

Donde la matriz S es de rango 2, R es de rango completo y la matriz resultante E llamada matriz esencial dada por $E = RS$ es de rango 2.

La matriz esencial relaciona las restricciones epipolares y las matrices de proyección, $P_R^T E P_L = 0$, con las propiedades extrínsecas del sistema estéreo de cámaras:

$$P_L = \frac{Z_L}{f_L} p_L ; P_R = \frac{Z_R}{f_R} p_R \Rightarrow \frac{Z_L Z_R}{f_L f_R} p_R^T p_L = 0 \Rightarrow p_R^T E p_L = 0 \tag{9}$$

E relaciona las propiedades extrínsecas de las cámaras, relación de posición de una cámara con respecto a la otra, pero no da información alguna de las componentes intrínsecas de cada cámara y por lo general esta información no es conocida de antemano. Es por esto que solo podemos obtener las mapeos de puntos a líneas epipolares solo para el punto correspondiente.

Para cada punto p tenemos dos puntos $\widetilde{p_L}$ y $\widetilde{p_R}$ en coordenadas pixel y p_L y p_R en coordenadas de cámara. Estos puntos están relacionados por dos matrices M_L y M_R de parámetros intrínsecos desconocidos:

$$\begin{aligned} & \widetilde{p_L} \equiv \begin{bmatrix} \widetilde{x_L} \\ \widetilde{y_L} \\ 1 \end{bmatrix} = M_L p_L; \widetilde{p_R} \equiv \begin{bmatrix} \widetilde{x_R} \\ \widetilde{y_R} \\ 1 \end{bmatrix} = M_R p_R \\ \Leftrightarrow \quad & p_L = M_L^{-1} \widetilde{p_L}; p_R = M_R^{-1} \widetilde{p_R} \\ \Leftrightarrow \quad & \widetilde{p_R}^T M_R^{-1} E M_L^{-1} \widetilde{p_L} = \widetilde{p_R}^T F \widetilde{p_L} \end{aligned} \tag{10}$$

F es la matriz fundamental que considera los parámetros intrínsecos y extrínsecos del sistema estéreo. Las matrices M_L y M_R pueden ser obtenidas por medios de calibración experimental y así podemos tener una solución del sistema sin conocimiento previo de los parámetros de las cámaras utilizadas. El método para obtener estas matrices es conocido como el algoritmo de los ocho puntos.

2.1 Calibración del sistema y parámetros de las cámaras

De la ecuación 10 se puede observar que todas las características de una cámara pueden ser expresadas por una matriz de 3 x 4, esta matriz transforma un punto en 3D $(x, y, z)^T$ a un punto 2D en la imagen $(u, v)^T$:

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} s \neq 0 \tag{11}$$

Donde s es un factor de escala de incertidumbre, este factor es resultante de errores al momento de capturar un cuadro, donde el capturador de cuadros puede realizar muestreo de pixeles para resolver errores de tiempo de adquisición o escaneo.

Los coeficientes de la matriz son desconocidos y depende de parámetros intrínsecos y extrínsecos del sistema, así tenemos que resolver el sistema de ecuaciones dado por:

$$\begin{cases} su = ax + by + cz + d \\ sv = ex + fy + gz + h \\ s = ix + jy + kz + 1 \end{cases} \tag{12}$$

Resolviendo para u y v:

$$\begin{cases} u = ax + by + cz + d - ixu - jyu - kzu \\ v = ex + fy + gz + h - ixv - jyv - kzv \end{cases} \tag{13}$$

Para obtener el valor de los coeficientes se puede utilizar patrones con objetos de tamaño conocidos, los cuales nos permitan saber qué puntos en la imagen $(u, v)^T$ corresponden con un punto en el espacio $(x, y, z)^T$ y a su vez establezca las relaciones en distancia entre pixeles y valores reales de medición, en este caso metros.

Para esto se realiza una calibración basada en el método de Zhang (Zhang, 1999). La técnica utilizada hace uso de un patrón de ajedrezado plano como objeto de referencia.

Con esto se obtiene un sistema lineal para un grupo de puntos P_i, $i \in \{1, \dots, N\}$ con coordenadas 3D espaciales $(x_i, y_i, z_i)^T$ y sus puntos correspondientes en pixeles $(u_i, v_i)^T$ conocidos:

$$\begin{pmatrix} x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -x_1u_1 & -y_1u_1 & -z_1u_1 \\ 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & -x_1v_1 & -y_1v_1 & -z_1v_1 \\ x_2 & y_2 & z_2 & 1 & 0 & 0 & 0 & 0 & -x_2u_2 & -y_2u_2 & -z_2u_2 \\ 0 & 0 & 0 & 0 & x_2 & y_2 & z_2 & 1 & -x_2v_2 & -y_2v_2 & -z_2v_2 \\ x_3 & y_3 & z_3 & 1 & 0 & 0 & 0 & 0 & -x_3u_3 & -y_3u_3 & -z_3u_3 \\ 0 & 0 & 0 & 0 & x_3 & y_3 & z_3 & 1 & -x_3v_3 & -y_3v_3 & -z_3v_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \\ j \\ k \end{pmatrix} = \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \end{pmatrix} \tag{14}$$

Para llevar a cabo la calibración se debe seguir el siguiente procedimiento:

- Imprimir un patrón en una superficie plana. Se imprimió un patrón ajedrezado con cuadros de 3 cm x 3 cm, que se pegó en una superficie de acrílico.
- Tomar imágenes en diferentes orientaciones, esto con la finalidad de corregir los cambios puntuales en la distorsión de la lente.
- Detectar los puntos de las imágenes, se le indica al programa las orillas del patrón y el tamaño de los cuadros, marcando los puntos de cada cuadro.
- Estimar los cinco parámetros extrínsecos e intrínsecos.
- Estimar los coeficientes de distorsión radial.
- Refinar todos los parámetros.

Para cada cámara se tomaron 18 fotografías del patrón en diferentes posiciones como se muestra en la Figura 18.

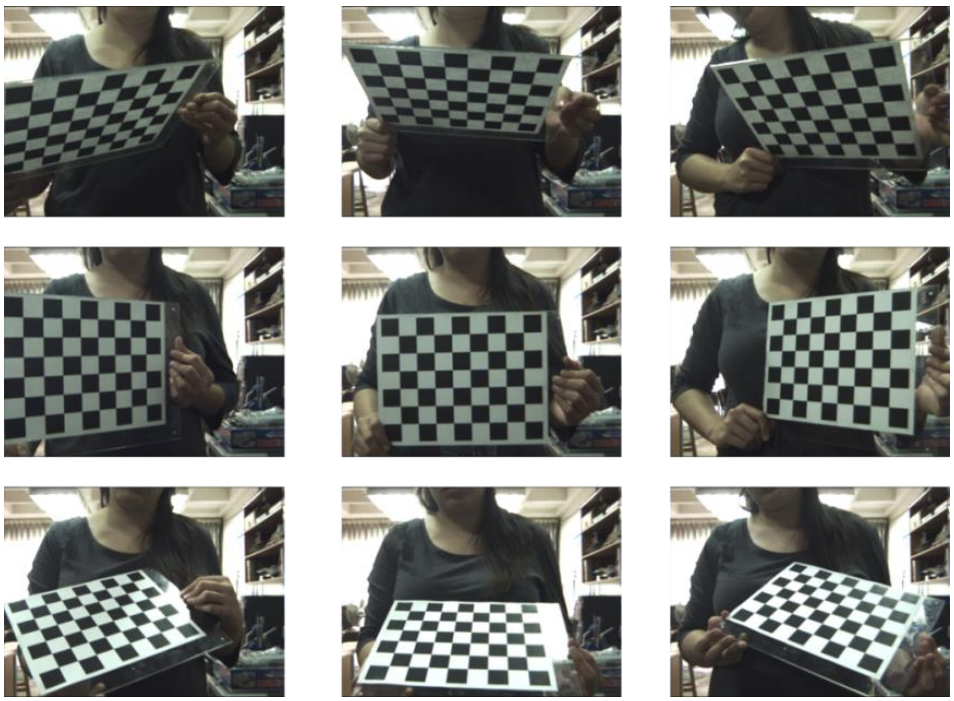

Figura 18. Adquisición de las imágenes con el patrón de calibración, los resultados de las tomas se utilizan para resolver el sistema linear presentado en la Ecuación 14.

La matriz se resuelve con una implementación del método de Zhang (Zhang, 1999) en C++ y con ayuda de la librería OpenCV (Bradski, 2000). Para la presentación grafica de resultados en este trabajo se utilizó el "*Camera Calibration Toolbox for Matlab*®".

Las esquinas que definen los puntos son extraídas para cada imagen (Figura 19).

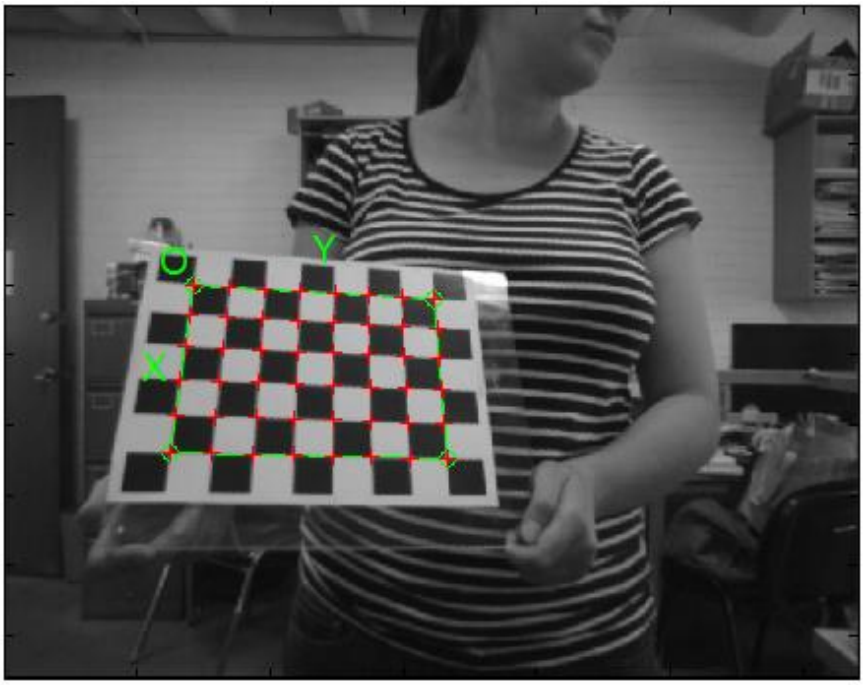

Figura 19. Segmentación de las esquinas del patrón de ajedrezado.

Inicialmente se obtienen los parámetros intrínsecos ignorando la distorsión y se obtienen los coeficientes de distorsión radial para refinar los parámetros resolviendo la proyección hasta que converja o se minimice. Los parámetros intrínsecos se presentan en la Tabla 1 y una representación gráfica de los parámetros extrínsecos se puede observar en la Figura 20 para uno de los pares de cámaras, el proceso se realizó para los tres pares de cámaras.

Distancia focal	f = [53.256132 53.187342] ± [0.201987 0.216526]
Punto principal	cc = [33.090400 24.366590] ± [0.327252 0.307594]
Sesgo	alpha_c = [0.00000] ± [0.00000] => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distorsión	kc = [-0.01141 0.02279 -0.00032 0.00004 0.00000] ± [0.00082 0.00231 0.00015 0.00016 0.00000]
Error de píxel	err = [0.01204 0.01339]

Tabla 1. Parámetros intrínsecos de la cámara 1 del primer par de cámaras.

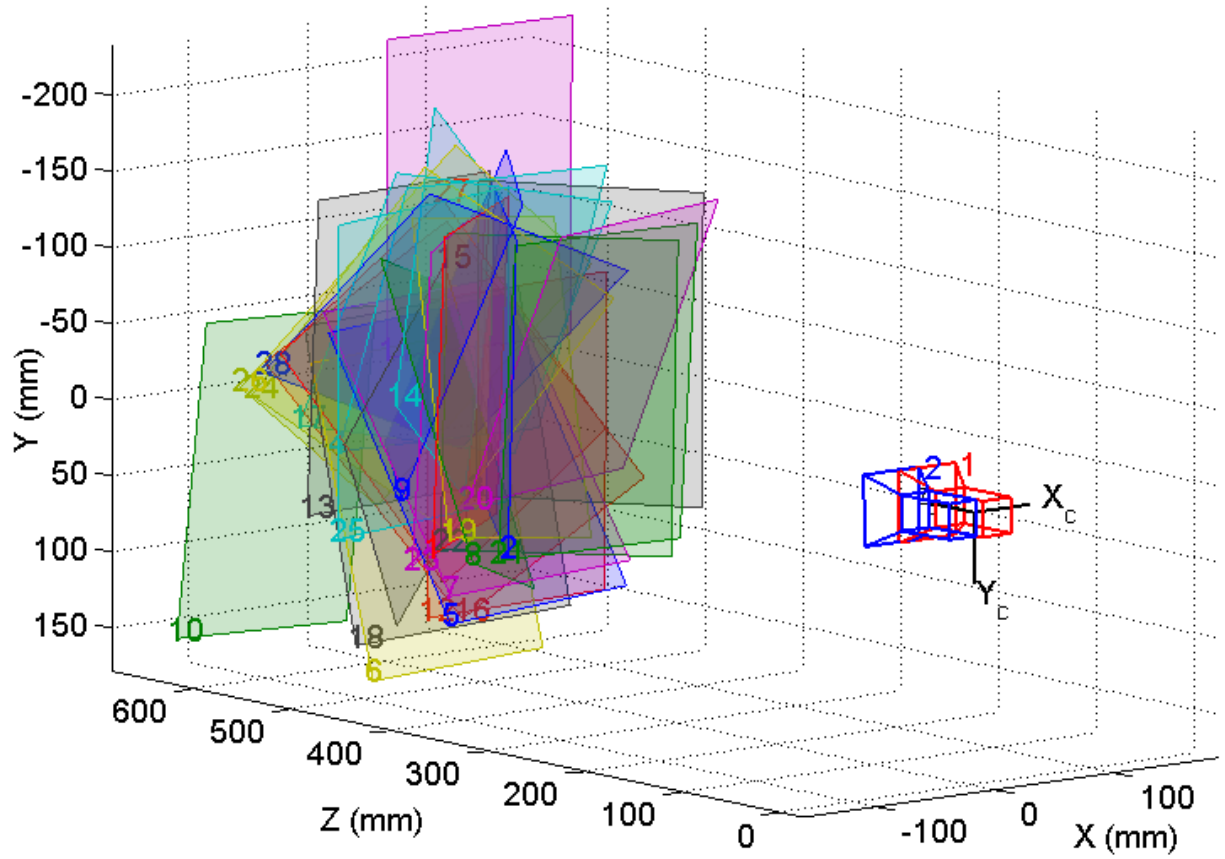

Figura 19. Relación espacial entre dos cámaras, una de ellos se coloca como el origen del sistema y la segunda tiene parámetros extrínsecos con respecto a esta. Los planos de colores muestran la posición del patrón de calibración en cada toma.

Utilizando los parámetros obtenidos se genera una matriz que permite rectificar las imágenes de cada cámara, logrando que ambas imágenes estén en el mismo sistema coordenado y las imágenes se corrigen para eliminar la distorsión introducida por las lentes, con esto obtenemos imágenes donde se cumple la restricción epipolar y la solución de la correspondencia entre pixeles en las dos imágenes se encuentra sobre una recta.

En la Figura 20 se puede ver arriba las imágenes sin corregir y abajo el resultado de aplicar la matriz obtenida en el proceso de calibración.

Figura 20. Arriba) Par de imágenes sin rectificar. Abajo) Imágenes rectificadas obtenidas, el pixel de una imagen y su par en la otra se encuentran sobre la misma línea epipolar.

El resultado de una geometría rectificada se utiliza en el algoritmo de estereovisión para describir la relación inversa entre los mapas de profundidad 3D Z y las disparidades d,

$$d = f\,{}^{B}/_{Z} \tag{15}$$

Donde $B = M^{-T}M^{-1}$ y f es la distancia focal obtenida del proceso de calibración.

Adicionalmente se realizaron dos calibraciones “cruzadas” con el fin de conocer la traslación y rotación de la cámara frontal respecto a las cámaras laterales; dicha calibración consistió en tomar la cámara frontal y lateral superior (derecha en un caso e izquierda en el otro), de manera tal que ambas cámaras vean el

patrón de calibración, lo que limitó el número de ángulos posibles. La localización de las cámaras entre sí se presenta en la Figura 21, donde la cámara central se muestra en rojo y las cámaras laterales en azul.

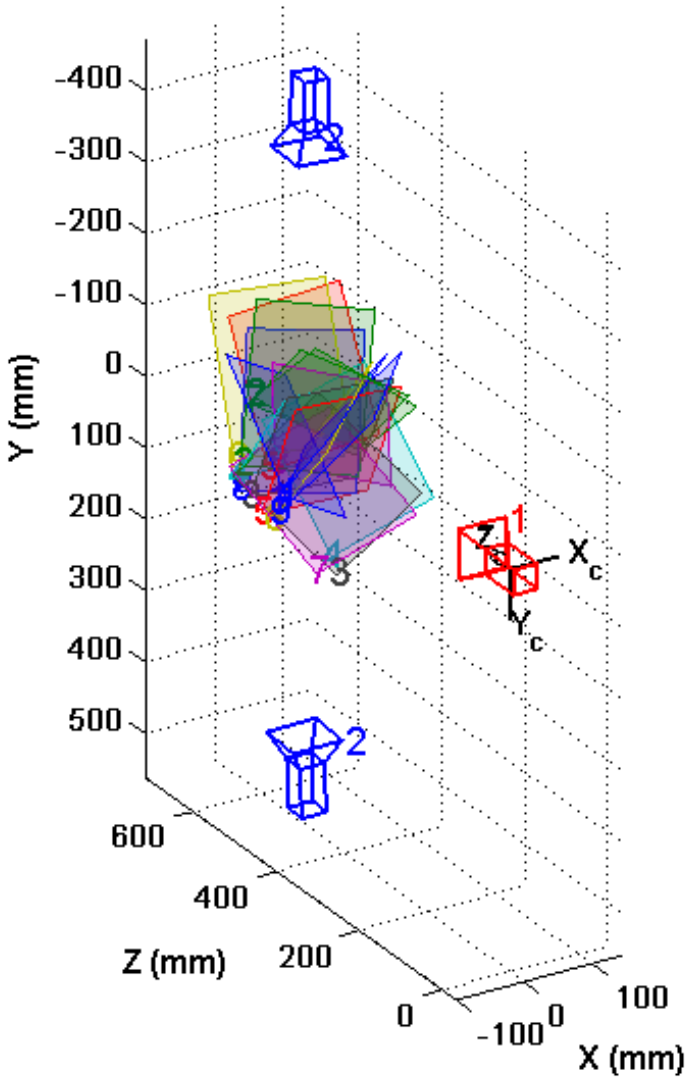

Figura 20. Posición de las cámaras respecto a la cámara frontal central.

Debido al ángulo de 90° entre las cámaras frontal y laterales no es posible cubrir todos los ángulos posibles para la calibración, por ello se tiene un error reproducible en cada calibración cruzada que fue corregido manualmente, es decir, añadiendo el valor del error al algoritmo de reconstrucción.

2.2 Mapa de disparidad

Las imágenes rectificadas son utilizadas para obtener el mapa de disparidad, el cual nos da la información de profundidad de cada punto. Para esto se buscan los pixeles que correspondan en ambas imágenes y la distancia resultante da el valor de disparidad para ese par de pixeles. La búsqueda de la correspondencia de pixeles se realiza con una implementación del algoritmo “Block matching + dynamic programing” (Delmas. 2013) desarrollado por el laboratorio IVZ de Nueva Zelanda. En la Figura 21 se presenta una imagen de la interface de usuario del programa en línea.

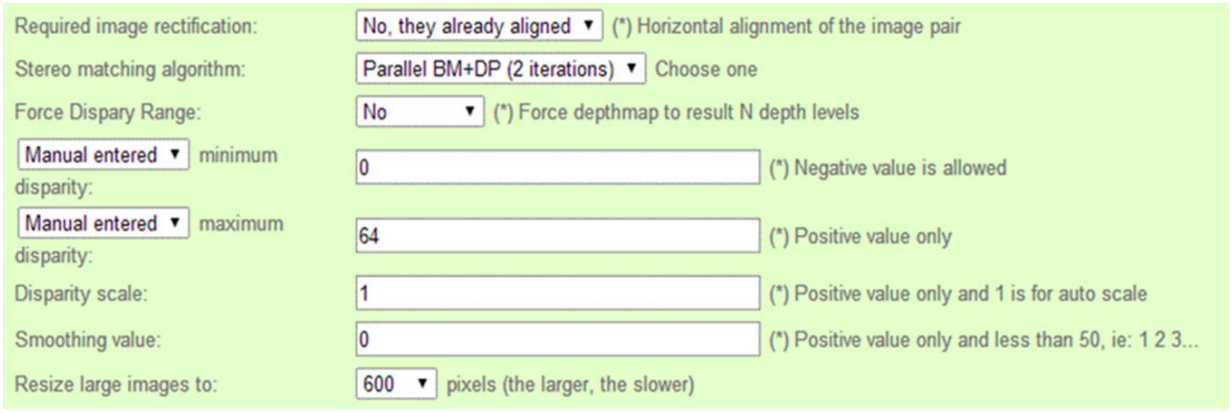

Figura 21. Interface de usuario de la página web con la que se obtienen los mapas de profundidad de las imágenes.

En la Figura 22 un ejemplo de un mapa de profundidad de un rostro es mostrado.

Figura 22. Mapa de disparidad resultante del sistema. Se utiliza pseudocolor para mostrar valores de profundidad.

3 Construcción del modelo de superficie del rostro

La superficie final del sistema se construye a partir de las tres imágenes de mapas de disparidad obtenidas por los tres puntos de vistas del sistema de adquisición.

El resultado final del sistema de adquisición es una malla tridimensional de superficie formada por triángulos como primitiva de representación.

3.1 Adquisición de los sujetos

Durante la adquisición de un sujeto las siguientes consideraciones se toman para disminuir las posibles oclusiones del rostro:

- Se les pide que retiren el cabello de su frente y orejas;
- Se retiran lentes u otros objetos como joyas que ocluyen los rasgos;
- Se sienten con la espalda recta y mirando al par de cámaras frontales;
- Se alinea la altura del rostro y se busca que las orejas queden a la altura de los pares de cámaras laterales.

Los sujetos son adquiridos con su rostro en posición neutral (no sonreír o realizar otro gesto). El programa de adquisición se configura para obtener 15 tomas en un minuto, donde el tiempo de integración de cada toma es de 0.16 segundos. Con esto se busca obtener para cada sujeto suficientes tomas de las cuales se pueda seleccionar tomas donde el sujeto no este parpadeando o presente movimientos durante la integración de la toma. En la Figura 23 se muestran la toma de una adquisición.

Figura 23. Adquisición de un sujeto con el sistema de estéreo – visión múltiple.

3.2 Reconstrucción de la superficie

Utilizando los mapas de disparidad de cada toma es posible definir una nube de puntos tridimensionales. Utilizando las propiedades extrínsecas obtenidas de las cámaras laterales con respecto a la central (Figura 20) se definen todas las nubes de puntos con respecto a las cámaras centrales, como se ilustra en la Figura 24.

Figura 24. Nube de puntos inicial obtenida de los mapas de disparidad.

La superficie final del sistema se construye a partir de las tres imágenes de mapas de disparidad obtenidas por los tres puntos de vistas del sistema de adquisición.

Posteriormente, para construir la malla triangular a partir de las nubes de puntos se utiliza un script desarrollado en Matlab y scripts que salvan los valores obtenidos en formato Ply. El algoritmo reconstruye las normales de la nube de puntos utilizando un proceso de vecindad y la malla triangular a través de una implementación del algoritmo de reconstrucción de Poisson. En la Figura 25 se muestran los pasos desde la nube de puntos a la malla.

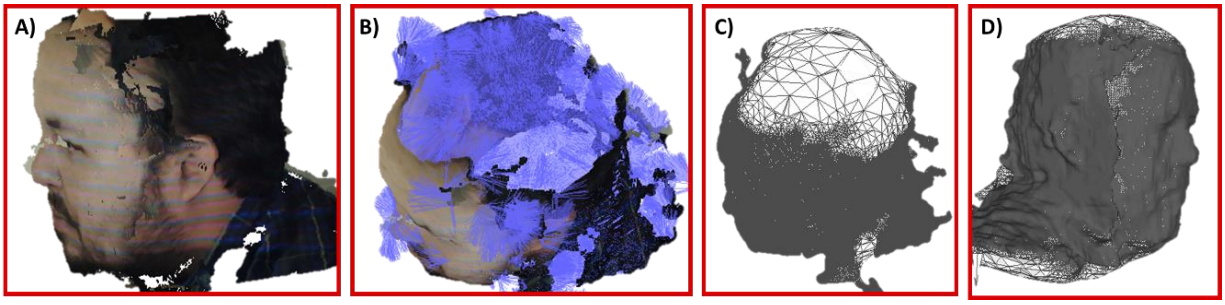

Figura 25. Pasos para la construcción de la malla triangular. A) obtención de la nube de puntos. B) cálculo de las normales por punto. C) reconstrucción de la malla con el algoritmo de Poisson. C) Filtrado de la malla obtenida por Poisson.

En la Figura 26 se muestra el resultado final para un sujeto, la textura obtenida por las cámaras se puede utilizar para darle textura a los triángulos de la malla. El código para construir las mallas se muestra en el Anexo D.

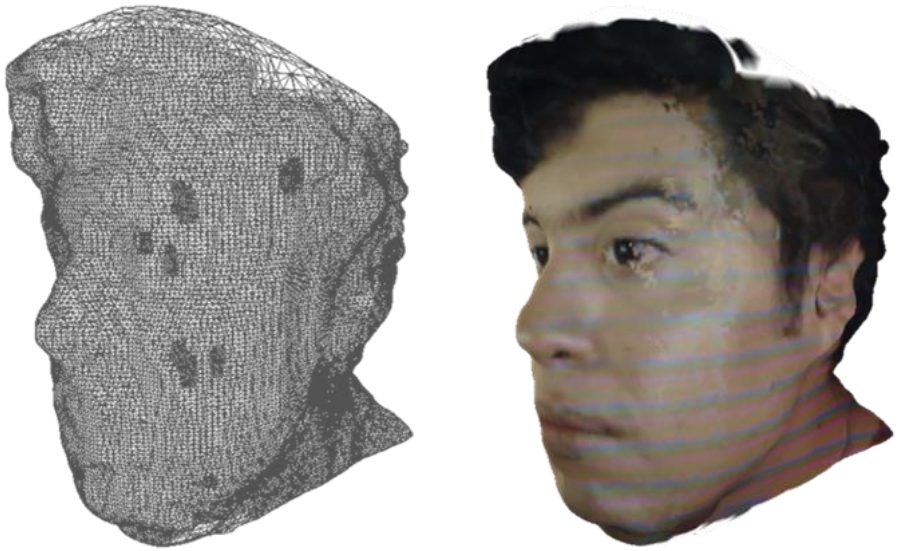

Figura 26. Malla triangular de un sujeto con mapeo de textura.

4 Análisis de datos

El estudio de las mallas obtenidas se realiza desde el punto de vista antropológico. Para poder obtener las medidas se realizó una selección manual de los puntos de interés “landmarks” sobre la malla triangular utilizando el programa Amira (Stalling et al, 2005). En la Figura 27 se presenta una imagen mostrando este proceso y las landmarks obtenidas.

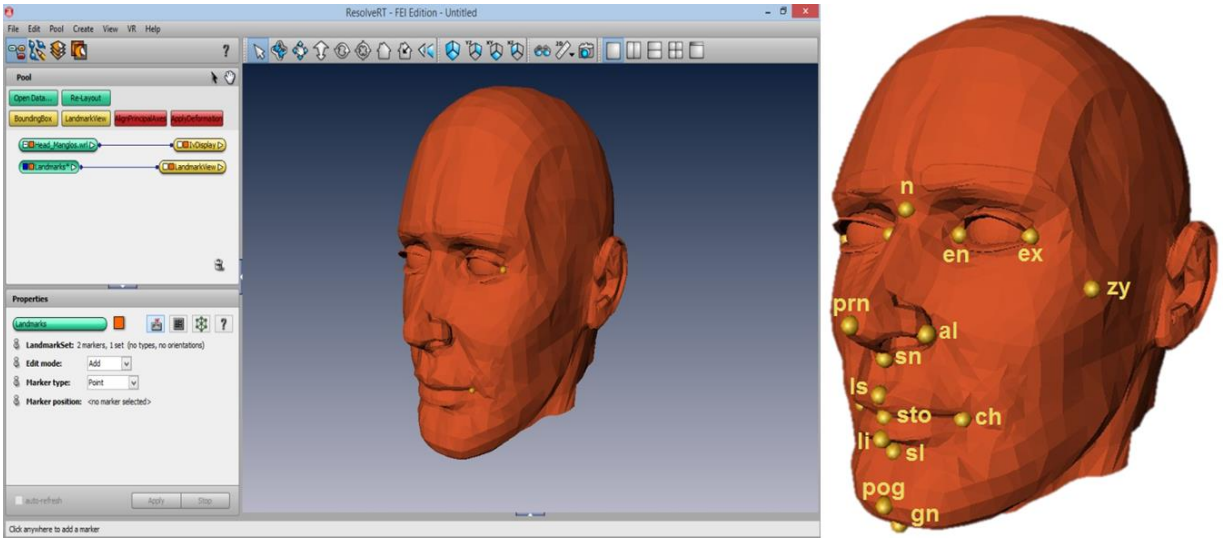

Figura 27. Marcaje de dos puntos somatométricos (landmarks) en el software Amira; estos aparecen como esferas amarillas.

4.1 Landmarks, medidas e índices

En La Tabla 2 se presenta la lista de los landmarks obtenidos junto con su nomenclatura y donde se localizan en el rostro.

	Landmark	Nom.	Localización
1	Nasion	n	El punto medio más profundo en la naciente de la nariz.
2	Endocatio*	en	Comisura interna de la hendidura de los ojos donde cierran los párpados.
3	Exocatio*	ex	Comisura externa de la hendidura de los ojos donde cierran los párpados.
4	Alar*	al	El punto más lateral del ala nasal.
5	Subnasal	sn	Punto de intersección del plano medio sagital con la línea que une los bordes inferiores de la nariz.
6	Labial superior	ls	Punto de la línea media sobre el borde del labio superior.
7	Stomion	sto	Punto de la línea media, entre ambos labios cuando están cerrados de forma natural.
8	Labial inferior	li	Punto de la línea media, sobre borde del labio inferior.
9	Gnation	gn	El punto más bajo en la línea media en el borde inferior del mentón.
10	Chelion*	ch	Comisura externa de la boca, donde los bordes exteriores de los labios superior e inferior se encuentran.
11	Sublabial	sl	El punto medio del surco mentolabial.
12	Pogonio	pog	El punto más sobresaliente en la mitad del mentón.
13	Pronasal	prn	El punto más sobresaliente en la punta nasal.
14	Zygion*	zy	El punto más lateral en el arco cigomático.

Tabla 2. Landmarks utilizados en este estudio.

A partir de los landmarks obtenidos se calculan medidas que cumplen con lo esperado en los estudios de antropometría: las mediciones deben ser precisas, es decir, de errores menores a 1 mm (Kook et al, 2014) y simples, lo que influye en la decisión de qué mediciones tomar, sin embargo deben ser suficientes para describir el rostro y permitan comparaciones.

Las medidas calculadas y la relación de los landmarks utilizados para cada una se muestran en la Tabla 3.

	Nombre	Distancia
d1	Altura facial superior	n-sto
d2	Altura facial total	n-gn
d3	Altura facial inferior	sn-gn
d4	Altura mandibular	sto-gn
d5	Anchura nasal	al^i-al^d
d6	Altura nasal	n-sn
d7	Altura labial superior	sn-sto
d8	Anchura bucal	ch^i-ch^d
d9	Altura cutánea labial superior	sn-ls
d10	Altura del labio superior	ls-sto
d11	Altura del labio inferior	sto-li
d12	Anchura binocular	ex^i-ex^d
d13	Anchura intercantal	en^i-en^d
d14	Altura de la boca	ls-li
d15	Distancia stomion a sublabial	sto-sl
d16	Altura del mentón	sl-gn
d17	Distancia exocantio izquierdo a labial inferior	ex^i-li
d18	Distancia alar izquierdo a pronasal	al^i-prn
d19	Anchura facial	zy^i-zy^d
d20	Distancia de exocantio derecho a subnasal	ex^d-sn
d21	Distancia de exocantio izquierdo a subnasal	ex^i-sn
d22	Distancia de chelion derecho a subnasal	ch^d-sn
d23	Distancia de chelion izquierdo a subnasal	ch^i-sn
d24	Distancia de exocantio derecho a labial inferior	ex^d-li
d26	Distancia de pogonio a nasion	pog-n
d27	Distancia de nasion a pronasal	prn-n
d28	Distancia de pogonio a pronasal	pog-prn

Tabla 3. Mediciones de distancias entre los landmarks. (i) izquierda, (d) derecha del sujeto.

Cada una de las mediciones presentadas en la Tabla 3 es simplemente la distancia entre dos landmarks:

$$d_k = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{16}$$

Donde d_k es cada uno de las medidas obtenidas en la Tabla 3 y (x, y, z) es la posición espacial de cada landmark l.

Finalmente de las medidas obtenidas se definen índices antropométricos (en adelante, simplemente "índices") que están definidos como un cociente entre dos distancias, que indican la proporcionalidad de una región del rostro que permite una evaluación objetiva y cuantitativa de la morfología facial y sus proporciones. Además, como se consideran distancias entre puntos coordenados, los índices no dependen de la orientación o posición de cada rostro en la base de datos de rostros; al ser también proporciones, dependen poco del tamaño absoluto de cada individuo.

Los índices se dividen en dos categorías: 1) regionales, de una sola área o 2) interregionales que son medidas de áreas distintas, por ejemplo, de la nariz y el rostro. Cada índice cuenta con un valor promedio y un intervalo de variación según el grupo, basado en edad, sexo y etnicidad. Dada la desviación estándar (SD) aquellos valores que se encuentran entre ± 2 SD son valores normales y entre ± 1 SD son valores óptimos, los que se encuentran fuera se puede considerar desproporcionados, o en algunos casos outliers. Sin embargo dos medidas desproporcionadas pueden compensarse y dar un índice balanceado, por lo tanto para determinar una desproporción es necesario analizar qué medidas la están causando (Naini, 2011).

Los ángulos medidos, propuestos por (George, 2007), se muestran en la Tabla 4 y los índices utilizados en este estudio son algunos de los propuestos por (Edler et al, 2006) y George, se muestran en la Tabla 5.

	Ángulo	Descripción
a1	Labial-orbital	ex^i-li-ex^d
a2	Nasal-orbital	ex^i-sn-ex^d
a3	Nasal-chelial	ch^i-sn-ch^d
a4	Nasal-facial	pog-n-prn
a5	Nasal-mentón	n-prn-pog
a6	Mentón-facial	n-pog-prn

Tabla 4. Ángulos calculados. (i) izquierda, (d) derecha del sujeto.

	Índice	Descripción
i1	Altura facial-facial superior	(n-sto)/(n-gn)
i2	Altura facial-facial inferior	(sn-gn)/(n-gn)
i3	Altura mandibular-facial	(sto-gn)/(n-gn)
i4	Altura mandibular-facial superior	(sto-gn)/(n-sto)
i5	Altura mandibular-facial inferior	(sto-gn)/(sn-gn)
i6	Altura labial superior-ancho bucal	(sn-sto)/(ch^i-ch^d)
i7	Altura cutánea-total labial superior	(sn-ls)/(sn-sto)
i8	Altura bermellón-total labial superior	(ls-sto)/(sn-sto)
i9	Altura bermellón-cutánea labial superior	(ls-sto)/(sn-ls)
i10	Contorno vertical del labial superior	(sn-sto)/((sn-ls)+(ls-sto))
i11	Bermellón	(ls-sto)/(sto-li)

i12	Altura facial superior-ancho binocular	(n-sto)/(ex^i-ex^d)
i13	Ancho intercantal-nasal	(en^i-en^d)/(al^i-al^d)
i14	Altura nasal-facial	(n-sn)/(n-gn)
i15	Ancho nasal-bucal	(al^i-al^d)/(ch^i-ch^d)
i16	Altura labial superior-facial superior	(sn-sto)/(n-sto)
i17	Altura labial superior-mandibular	(sn-sto)/(sto-gn)
i18	Altura labial superior-nasal	(sn-sto)/(n-sn)
i19	Nasal	(al^i-al^d)/(n-sn)
i20	Bucal	(ls-li)/(ch^i-ch^d)
i21	Intercantal	(en^i-en^d)/(ex^i-ex^d)
i22	Altura mandibular-mentón	(sl-gn)/(sto-gn)
i23	Longitud labial superior	(sn-sto)/(sn-gn)
i24	Longitud labial inferior	(sto-sl)/(sn-gn)
i25	Longitud mentón	(sl-gn)/(sn-gn)
i26	Altura labial inferior-mandibular	(sto-sl)/(sto-gn)
i27	Altura labial inferior-mentón	(sto-sl)/(sl-gn)
i28	Triángulo labial orbital	(ex^i-li)/(ex^i-ex^d)

Tabla 5. Índices calculados en el estudio.

Los índices son obtenidos como una relación entre dos mediciones:

$$indice_r = \frac{d_{k_1}}{d_{k_2}} \tag{17}$$

4.2 Evaluación de los errores en las medidas manuales

Para evaluar el error en las medidas antropométricas se utiliza el error técnico de medición (TEM, por sus siglas en inglés), que se define por la ecuación (18):

$$TEM = \sqrt{\frac{\sum_{k=1}^{n} l_k^2}{2n}} \tag{18}$$

Donde l_k es la diferencia entre la misma medición realizada por el mismo observador, y n es el número de sujetos evaluados, (Frisancho, 1990).

4.3 Análisis de Componentes Principales

El Análisis de Componentes Principales (PCA, del inglés) (Jolliffe, 1986) es un método que permite identificar patrones en una serie de datos y expresar ésta de acuerdo a sus similitudes y diferencias. A, l encontrar los patrones se reduce el número de dimensiones sin una gran pérdida de información.

La idea central del PCA es reducir la dimensionalidad de una serie de datos en los que hay un gran número de variables interrelacionadas, conservando lo más posible la variación presente en los datos. La reducción se logra transformando a una nueva serie de datos, llamados componentes principales, que no tienen correlación entre sí, o es mínima, y están ordenados de tal forma que los primeros contengan la mayor variación presente en todas las variables originales.

Se utiliza un script programado en Matlab utilizando sus las librerías propietarias para obtener el cálculo de PCA de los datos y el estudio de los factores con mayor y menor varianza en la población.

5 Resultados

El sistema de adquisición debe de ser capaz de adquirir rostros humanos de oreja a oreja y obtener de las tres tomas una superficie triangular de la cual se puedan obtener medidas características. Así para probar la capacidad de adquisición el sistema y evaluar la capacidad de este como método de medición antropológica se adquirieron las mallas triangulares de 35 sujetos (20 hombres y 15 mujeres).

El marcaje de los landmarks fue realizado en dos diferentes ocasiones, por el mismo sujeto.

Durante el proceso de adquisición existen posturas de los sujetos que complican la selección manual de los landmarks en la malla reconstruida:

- Sujetos con su rostro ladeado hacia alguno de sus hombros, debido a que la fuente de luz utilizada puede causar puntos calientes que producen un desplazamiento en la posición del tabique de la nariz en la malla final.
- Sujetos con mejillas muy prominentes provocaban sombras en las comisuras de sus labios dificultando la localización del chelion.

Así mismo se tienen posiciones de landmarks que debido al arreglo de las cámaras no se pueden adquirir con confianza en todos los sujetos:

- El gnation se vio afectado debido a que la posición de las cámaras provoca en el frente una oclusión de la parte más sobresaliente del mentón sobre su parte inferior, y las imágenes laterales no son suficientes para reconstruir adecuadamente dicha región.
- El nasion ya que las imágenes de las laterales, a 90°, no logran adquirir la región del nacimiento de la nariz pues se ve ocluida por los arcos superciliares y las cejas.

A pesar de estos problemas el valor del TEM para todas las mediciones fue menor a 1 mm, por lo tanto se consideran aceptables,

Las medidas que presentan mayor error son:

- Distancia de chelion derecho a subnasal (0.71 mm)
- Anchura bucal (0.69 mm)
- Altura nasal (0.53 mm)

Las que presentan menor error son:

- Anchura nasal (0.30 mm)
- Anchura facial (0.34 mm)
- Altura de la boca (0.37 mm).

En las Tabla 6, 7 y 8 se muestran los valores promedio de las mediciones de los índices y de los ángulos.

Medida	Valor promedio	Valor promedio Mujeres	Valor promedio Hombres

	(mm)	± (mm)	(mm)	± (mm)	(mm)	± (mm)
1	74	6	73	7	75	6
2	122	10	116	8	126	9
3	72	9	67	6	76	9
4	50	8	46	5	54	8
5	37	4	35	3	38	4
6	53	6	52	6	53	6
7	22	3	21	2	23	3
8	54	5	51	4	55	5
9	15	2	15	2	16	3
10	7	2	7	1	7	2
11	9	2	9	2	9	2
12	96	5	95	5	97	5
13	38	3	37	3	38	3
14	16	3	16	3	16	3
15	20	3	18	3	21	3
16	31	7	28	5	33	8
17	95	6	93	6	97	5
18	25	3	24	3	26	3
19	130	8	127	7	133	8
20	72	4	71	3	73	4
21	74	4	72	4	75	4
22	37	3	36	3	38	3
23	40	4	38	4	41	3
24	94	6	92	5	96	6
25	109	8	105	8	112	7
26	42	6	42	6	42	6
27	75	7	71	5	78	7

Tabla 6. Valores de distancia promedio.

Índice	Valor promedio		Valor promedio Mujeres		Valor promedio Hombres	
		±		±		±
1	0.61	0.04	0.63	0.04	0.59	0.04
2	0.59	0.05	0.58	0.04	0.61	0.05
3	0.41	0.04	0.39	0.03	0.43	0.04
4	0.69	0.12	0.63	0.09	0.73	0.13
5	0.70	0.04	0.68	0.03	0.70	0.04
6	0.42	0.06	0.42	0.04	0.41	0.07
7	0.69	0.05	0.68	0.04	0.69	0.06
8	0.32	0.06	0.33	0.04	0.31	0.07
9	0.48	0.12	0.49	0.09	0.47	0.14
10	0.99	0.01	0.99	0.01	0.99	0.01
11	0.85	0.19	0.88	0.24	0.83	0.16
12	0.77	0.06	0.77	0.06	0.77	0.07
13	1.03	0.11	1.07	0.13	1.01	0.08
14	0.43	0.05	0.45	0.04	0.42	0.05
15	0.69	0.07	0.69	0.06	0.69	0.08

16	0.30	0.04	0.30	0.04	0.31	0.05
17	0.45	0.08	0.47	0.06	0.43	0.08
18	0.43	0.08	0.41	0.08	0.44	0.09
19	0.71	0.11	0.68	0.08	0.74	0.13
20	0.29	0.06	0.31	0.06	0.29	0.07
21	0.40	0.03	0.39	0.03	0.40	0.03
22	0.60	0.07	0.61	0.07	0.60	0.07
23	0.31	0.04	0.32	0.03	0.30	0.04
24	0.28	0.04	0.27	0.05	0.28	0.04
25	0.42	0.06	0.41	0.06	0.43	0.07
26	0.40	0.07	0.40	0.07	0.40	0.07
27	0.69	0.20	0.68	0.22	0.69	0.20
28	0.99	0.05	0.98	0.04	1.00	0.05

Tabla 7. Valores de índices promedio.

Ángulo	Valor promedio		Valor promedio Mujeres		Valor promedio Hombres	
	(°)	±(°)	(°)	±(°)	(°)	±(°)
1	82	4	82	3	81	5
2	88	6	88	4	89	6
3	61	3	62	3	60	4
4	29	3	29	3	29	2
5	135	4	135	4	135	4
6	16	3	16	2	15	3

Tabla 8. Valores de ángulos promedio.

Los valores promedios obtenidos no dan información clara del comportamiento general de la población ya que el número de medidas obtenidas crean un problema de alta dimensionalidad. Es necesario presentar las medidas en una forma que permita estudiar cuales de estas son representativas de la población y cuales en la población obtenida tienen una varianza alta.

Para reducir la dimensionalidad del problema se utiliza el método de Análisis de Componentes Principales (PCA). Con esto podemos obtener las variables (distancias, índices y ángulos) que tienen una mayor varianza en la muestra de sujetos en el nuevo espacio producido por PCA.

En la Figura 28 se muestran los diagramas de Pareto con el porcentaje de varianza que cada componente principal explica para el resultante del método de PCA aplicado a distancias, índices, ángulos y todos los grupos concatenados.

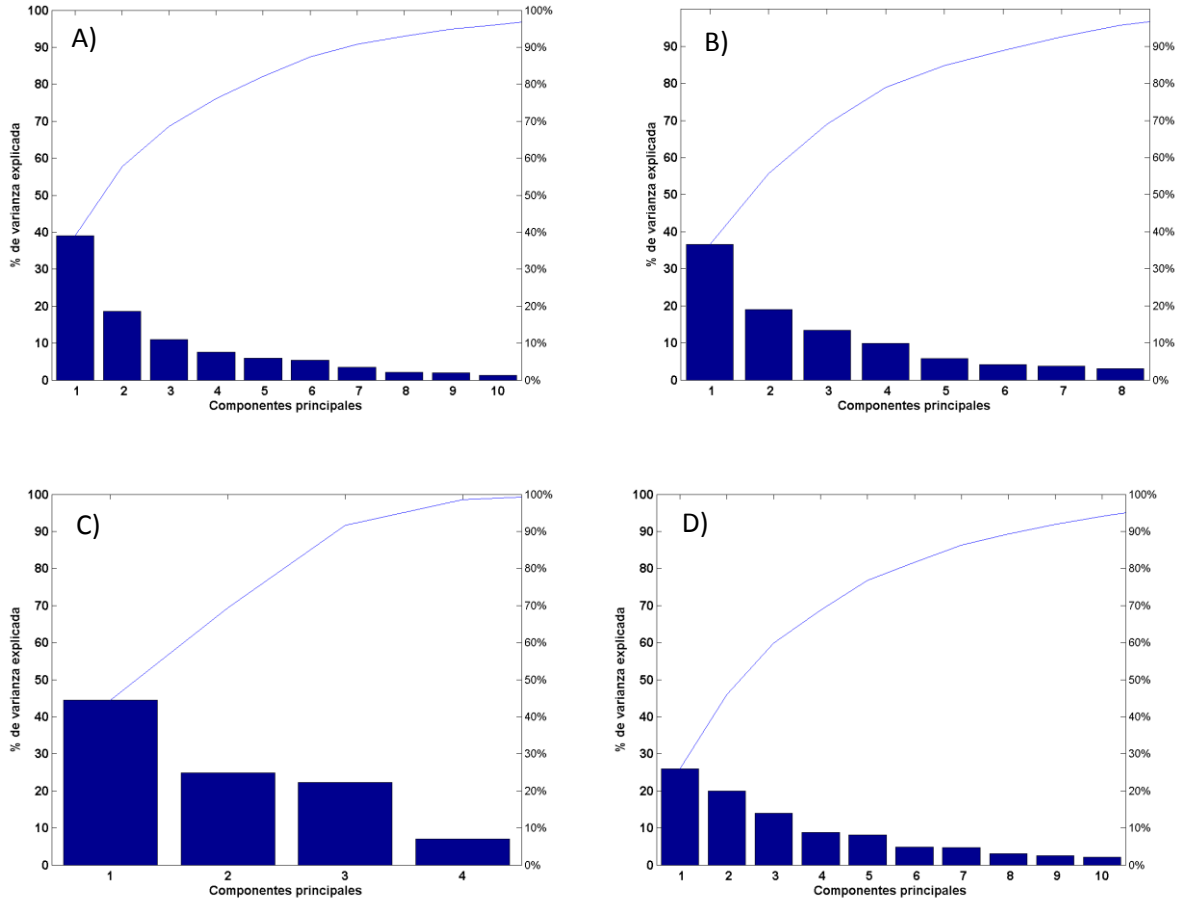

Figura 28. Diagrama de Pareto mostrando el porcentaje de varianza explicado por cada componente principal para cada grupo de medida. A) Las tres primeras componentes principales explican un 68.56% de la varianza en las distancias. B) Para los índices explican el 69.03%. C) Para los ángulos el 91.65%. D) Para la concatenación de todas las medidas explican el 59.94%.

Utilizando las tres primeras componentes principales se encuentran la proyección de cada una de las medidas en el espacio de componentes principales. Con esto se encuentran las medidas que son más características de la poblaciones y aquellas que varían en mayor mediad a lo largo de todos los sujetos.

En la Figura 29 se muestran los resultados para las distancias.

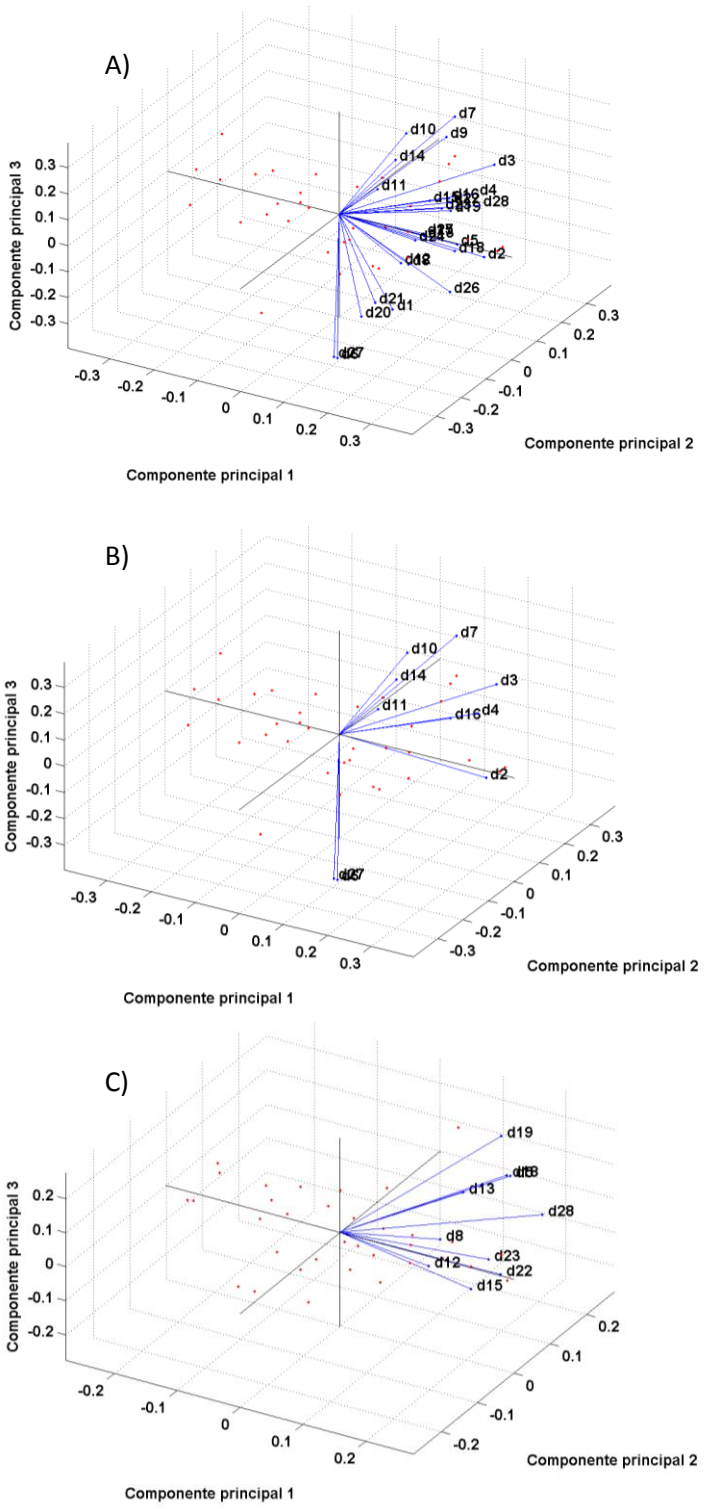

Figura 29. Resultado de la proyección de las medidas de distancia. A) Representación vectorial de las medidas de distancia en el espacio de componentes principales. B) Medidas de distancia con mayor variación. C) Medidas de distancia con menor variación.

En la Figura 30 se muestran los resultados para los índices.

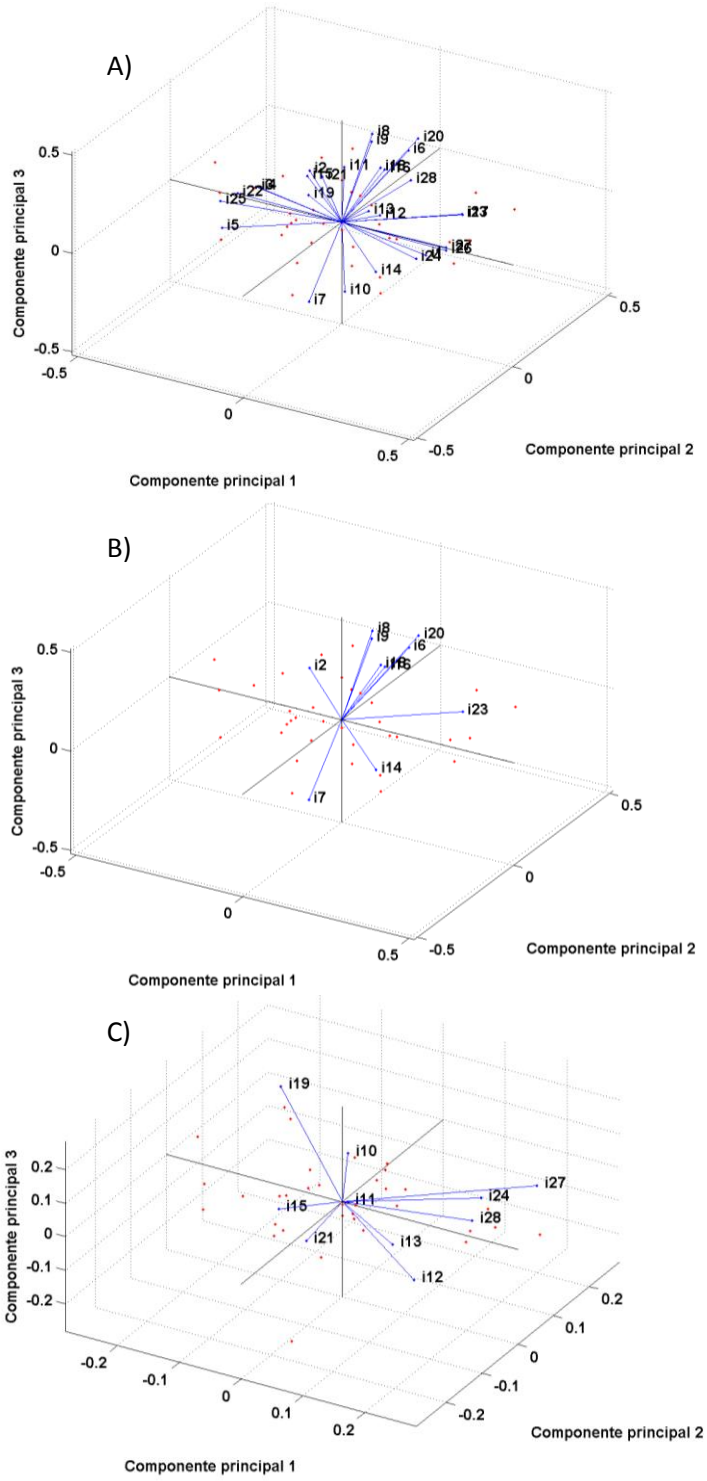

Figura 30. Resultado de la proyección de las medidas de los índices. A) Representación vectorial de las medidas de índices en el espacio de componentes principales. B) Medidas de índices con mayor variación. C) Medidas de índices con menor variación.

En la Figura 31 se muestran los resultados para los ángulos.

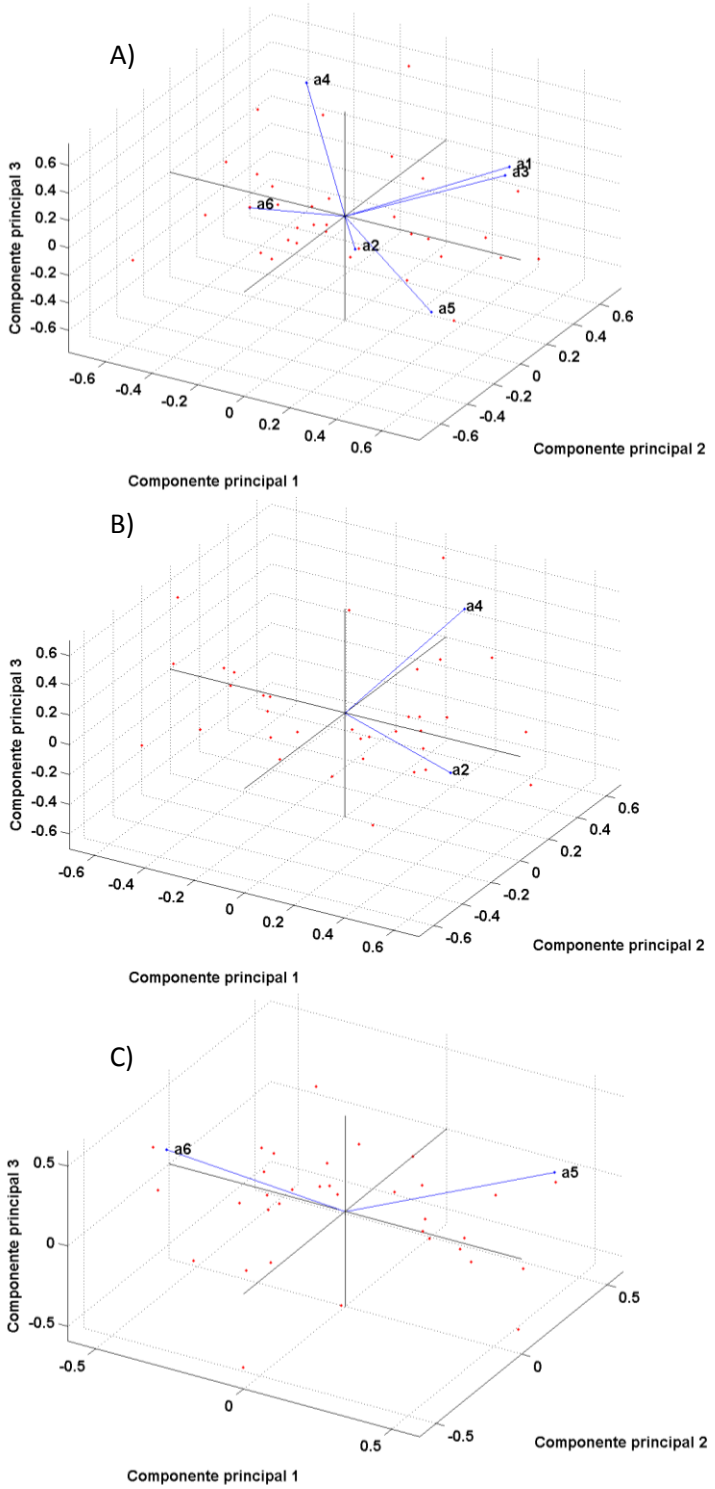

Figura 31. Resultado de la proyección de las medidas de los ángulos. A) Representación vectorial de las medidas de ángulos en el espacio de componentes principales. B) Medidas de ángulos con mayor variación. C) Medidas de ángulos con menor variación.

En la Figura 32 se muestran los resultados para las medidas concatenadas.

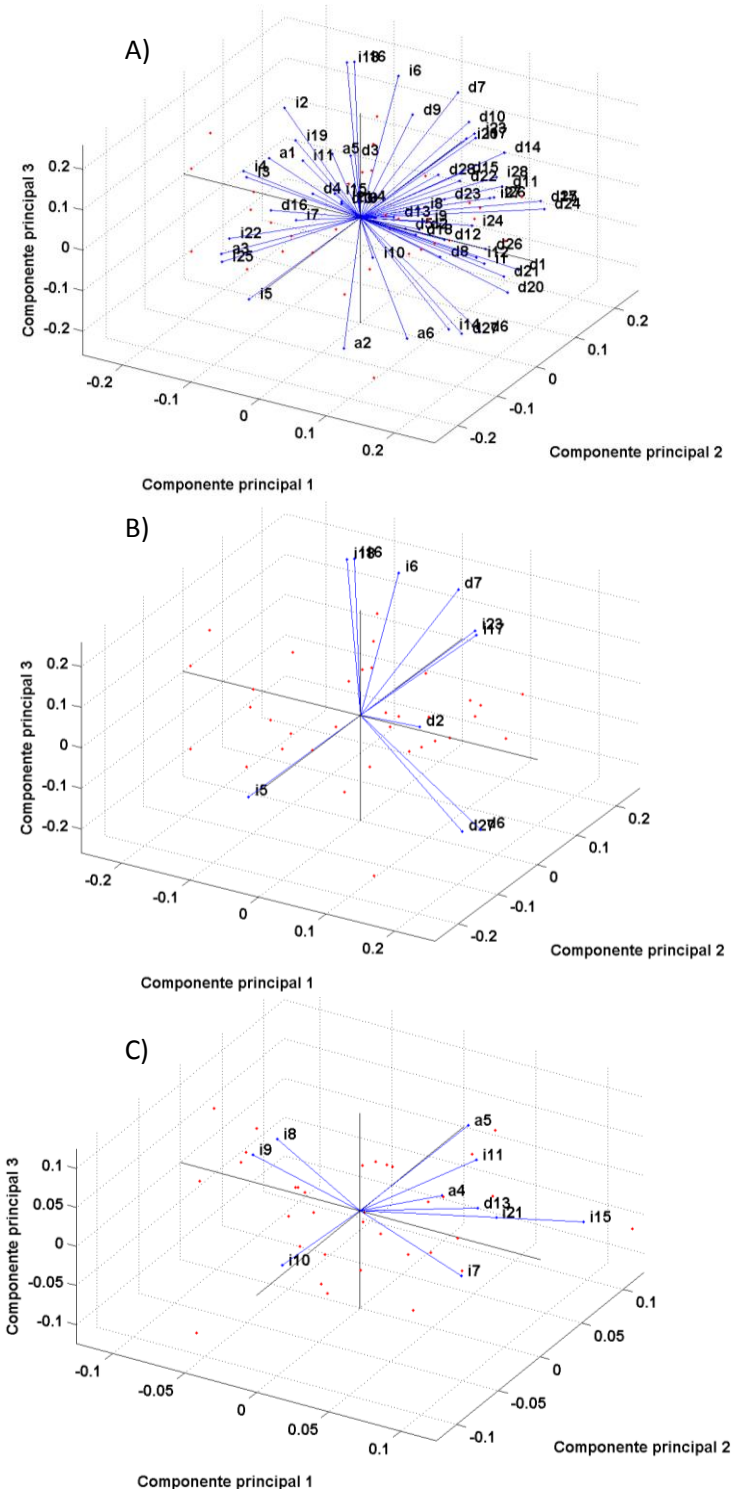

Figura 32. Resultado de la proyección de las medidas concatenadas. A) Representación vectorial de las medidas concatenadas en el espacio de componentes principales. B) Medidas concatenadas con mayor variación. C) Medidas concatenadas con menor variación.

El resultado del método de PCA sobre las medidas obtenidas es presentado en la Tabla 9.

Distancia		Índice		Angulo		Concatenado	
Mayor varianza	Menor varianza	Mayor varianza	Menor varianza	Mayor varianza	Menor varianza	Mayor varianza	Menor varianza
d11	d13	i23	i13	a2	a5	d27	a4
d2	d8	i2	i12	a4	a6	i5	i10
d10	d12	i14	i21			i17	i8
d3	d23	i6	i15			i23	i9
d4	d15	i20	i11			d6	i7
d14	d22	i18	i24			i6	i21
d16	d5	i16	i28			d7	a5
d7	d18	i7	i10			d2	i11
d27	d28	i9	i19			i18	d13
d6	d19	i8	i27			i16	i15

Tabla 8. Medidas con mayor y menos varianza en la población estudiada. Des estos valores podemos saber qué medidas son características de la población.

En la Tabla 9 se presentan los sujetos más extremos (alejados de la población media) para cada una de las medidas.

Sujetos más extremos			
Distancia	Índice	Angulo	Concatenado
s18	s32	s27	s10
s24	s33	s32	s15
s26	s16	s15	s21
s2	s21	s3	s7
s23	s17	s33	s18
s13	s11	s19	s29
s3	s19	s17	s6

s17	s13	s29	s14
s29	s26	s26	s31
s28	s20	s16	s5

Tabla 9. Sujetos que se alejan más de la población media. Se observa que esto es altamente dependiente de qué medidas se utilicen. Utilizando todas las medidas el sujeto 10 es el más extremo.

Utilizando el resultado obtenido de PCA es posible agrupar a los sujetos con respecto a la varianza de sus medidas. En la Figura 33 se muestra el resultado de aplicar el método de agrupado k-means en los datos obtenidos de PCA.

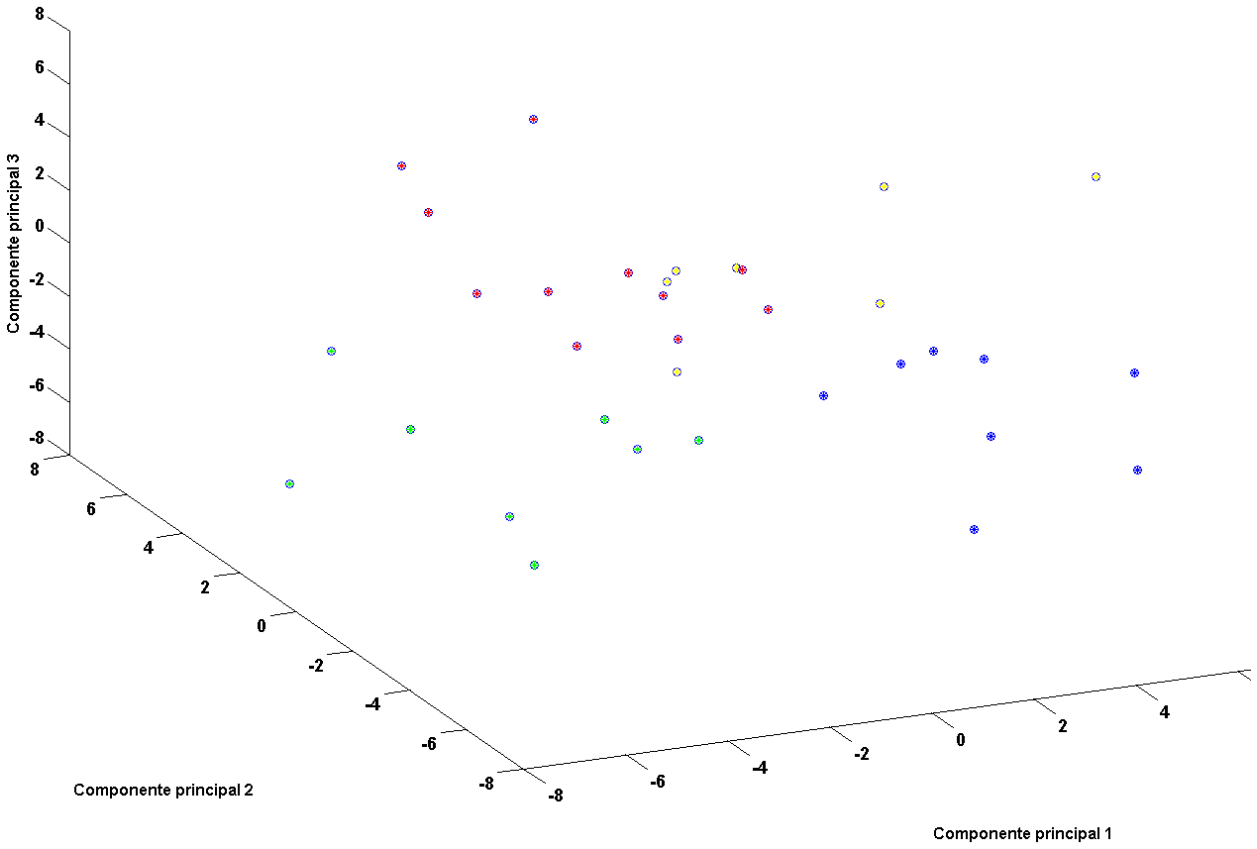

Figura 33. Separación de los individuos en grupos utilizando el método de k-means.

El código para el análisis estadístico de los datos obtenidos se muestra en el Anexo E.

Conclusiones

El sistema presentado es capaz de adquirir tres mapas de disparidad de rostros humanos de oreja a oreja a una velocidad máxima de 60 cuadros por segundo y reconstruir automáticamente la malla triangular que lo represente.

Debido a la calidad de las cámaras y su baja ganancia es necesario utilizar un sistema de iluminación difusa para mejorar los datos adquiridos, en un nuevo prototipo se propone utilizar cámaras de grado científico que ayuden a reducir el número de fuentes de luz externa necesarias y mejore con esto la portabilidad del sistema final.

Para resolver los problemas de oclusión y áreas no adquiridas debidos al diseño del sistema se sugiere un arreglo que involucre más cámaras, un nuevo par frontal bajo que logre adquirir la parte inferior del mentón y experimentar en cuál es la mejor relación espacial con las laterales para obtener un ángulo menor a 90° que aun sea capaz de adquirir el rostro de oreja a oreja con la finalidad de mejorar el registro de todas las cámaras del sistema.

También es importante trabajar en la manipulación de los volúmenes tridimensionales desarrollando algoritmos que mejoren la selección de los puntos de interés sobre la superficie, si es posible de forma automática o semiautomática, proponiendo un punto cercano que pueda ser corregido por el usuario.

Agradecimientos

Doctor Patrice Delmas, Edwin y los que consideres de Auckland, La Dra Lilia Escorcia del Instituto de Estudios Antropológicos, Dr. Luis Benítez Bribiesca por alguna tutoría en la tesis de Leticia, Beca de maestría de Conacyt 2012-2014 a Leticia López y apoyo de proyectos PAEP 2012.

Anexo A

Código de OpenSCAD para construir el modelo por manufactura aditiva del contenedor de la cámara:

```
radio_tornillo_cam = 1.1;
module screw(type, r1, r2, n, h, t)
{
	linear_extrude(height = h, twist = 360*t/n, convexity = t)
	difference() {
		circle(r2);
		for (i = [0:n-1]) {
			if (type == 1) rotate(i*360/n) polygon([
				[ 2*r2, 0 ],
				[ r2, 0 ],
				[ r1*cos(180/n), r1*sin(180/n) ],
				[ r2*cos(360/n), r2*sin(360/n) ],
				[ 2*r2*cos(360/n), 2*r2*sin(360/n) ],
			]);
			if (type == 2) rotate(i*360/n) polygon([
				[ 2*r2, 0 ],
				[ r2, 0 ],
				[ r1*cos(90/n), r1*sin(90/n) ],
				[ r1*cos(180/n), r1*sin(180/n) ],
				[ r2*cos(270/n), r2*sin(270/n) ],
				[ 2*r2*cos(270/n), 2*r2*sin(270/n) ],
			]);
		}
	}
}

module nut(type = 3, r1 = 1.5, r2 = 1.0, r3 = 3, s = 6, n = 1, h = 30/10, t =
8/10)
{
	difference() {
		cylinder($fn = s, r = r3, h = h);
		translate([ 0, 0, -h/2 ]) screw(type, r1, r2, n, h*2, t*2);
	}
}


//caja frontal
difference(){
	union(){
	//caja original
		difference(){
			translate ([0,0,0]) cube(size = [82,20,56]);
			translate ([4,0,4]) cube(size = [74,20,48]);
		}
	   //cara
		difference(){
		  translate ([0,0,0]) cube(size = [82,1,56]);
		  //diferencia para crear la salida de la lente de la camara
		  //como esta rotado 90 grados y y z cambian de orden en translate
		  translate ([41,2.5,14.5]) rotate ([90,0,0]) cylinder(h=5,r=10);
		  //salidas del tornillo que fija la camara
```

```
              translate ([10.0,2.5,30]) rotate ([90,0,0]) screw(3, 1.5,
radio_tornillo_cam, 1, 30, 8);
              translate ([70.0,2.5,30]) rotate ([90,0,0]) screw(3, 1.5,
radio_tornillo_cam, 1, 30, 8);
              //espacio para microfonos
              translate ([5,0,37.1])  cube(size = [72,1,10]);
              //DIODOS
              translate ([18,2.5,27]) rotate ([90,0,0])
cylinder(h=5,r=2.0,$fn=100);
              translate ([62,2.5,27]) rotate ([90,0,0])
cylinder(h=5,r=2.0,$fn=100);
            }
      //sujetadores camara
            translate ([10.0,5,30]) rotate ([90,0,0])
            difference(){
                  cylinder(h=5,r=2.0,$fn=100);
                  screw(3, 1.5, radio_tornillo_cam, 1, 5, 8);
            }
            translate ([70.0,5,30]) rotate ([90,0,0])
            difference(){
                  cylinder(h=5,r=2.0,$fn=100);
                  screw(3, 1.5, radio_tornillo_cam, 1, 5, 8);
            }
      }
        //tornillos caja
        translate ([2.5,-1,2.5]) rotate ([-90,0,0]) screw(3, 1.5,
radio_tornillo_cam, 1, 30, 8);
        translate ([2.5,-1,53.5]) rotate ([-90,0,0]) screw(3, 1.5,
radio_tornillo_cam, 1, 30, 8);
        translate ([79.5,-1,53.5]) rotate ([-90,0,0]) screw(3, 1.5,
radio_tornillo_cam, 1, 30, 8);
        translate ([79.5,-1,2.5]) rotate ([-90,0,0]) screw(3, 1.5,
radio_tornillo_cam, 1, 30, 8);
        //conectores
        translate ([0,10,10]) rotate ([0,90,0]) screw(3, 1.5,
radio_tornillo_cam, 1, 30, 8);
     translate ([0,10,46]) rotate ([0,90,0]) screw(3, 1.5, radio_tornillo_cam,
1, 30, 8);
        translate ([76,10,10]) rotate ([0,90,0]) screw(3, 1.5,
radio_tornillo_cam, 1, 30, 8);
     translate ([76,10,46]) rotate ([0,90,0]) screw(3, 1.5,
radio_tornillo_cam, 1, 30, 8);
        translate ([10,10,46]) rotate ([0,0,0]) screw(3, 1.5,
radio_tornillo_cam, 1, 30, 8);
        translate ([72,10,46]) rotate ([0,0,0]) screw(3, 1.5,
radio_tornillo_cam, 1, 30, 8);
     translate ([10,10,-10]) rotate ([0,0,0]) screw(3, 1.5,
radio_tornillo_cam, 1, 30, 8);
     translate ([72,10,-10]) rotate ([0,0,0]) screw(3, 1.5,
radio_tornillo_cam, 1, 30, 8);
}
```

Anexo B

El controlador Arduino se utiliza para generar el pulso de sincronización del sistema de cámaras. El código desarrollado para esto se presenta a continuación:

```
int Pinnumber = 9;      // PIN de salida de la señal cuadrada
int val = 0;            // Variable de control de voltaje de salida
int time = 0;           // Variable de control temporal

void setup(){
      pinMode(Pinnumber, OUTPUT);   // Se selecciona al PIN como uno de salida
}

void loop(){
      if (val == 0) {
            val = 168;
            time = 2000;
      }
      else {
            val = 0;
            time = 160;
      }
      // Escritura del valor de voltaje al pin de salida
      analogWrite(Pinnumber, val);
      // Tiempo de retraso entre cambios de la señal cuadrada
      delay(time);
}
```

Anexo C

La adquisición de imágenes se hace utilizando un código de captura de memoria en anillo utilizando la librería de CLEyeMulticam para controlar cada cámara, esto solo funciona para las cámaras PS3 – EYE cada cámara tiene su propio controlador, y la librería OpenCV para crear los contenedores de imágenes y la calibración de las cámaras. En la Figura C.1. Se muestra el diagrama de relaciones del algoritmo.

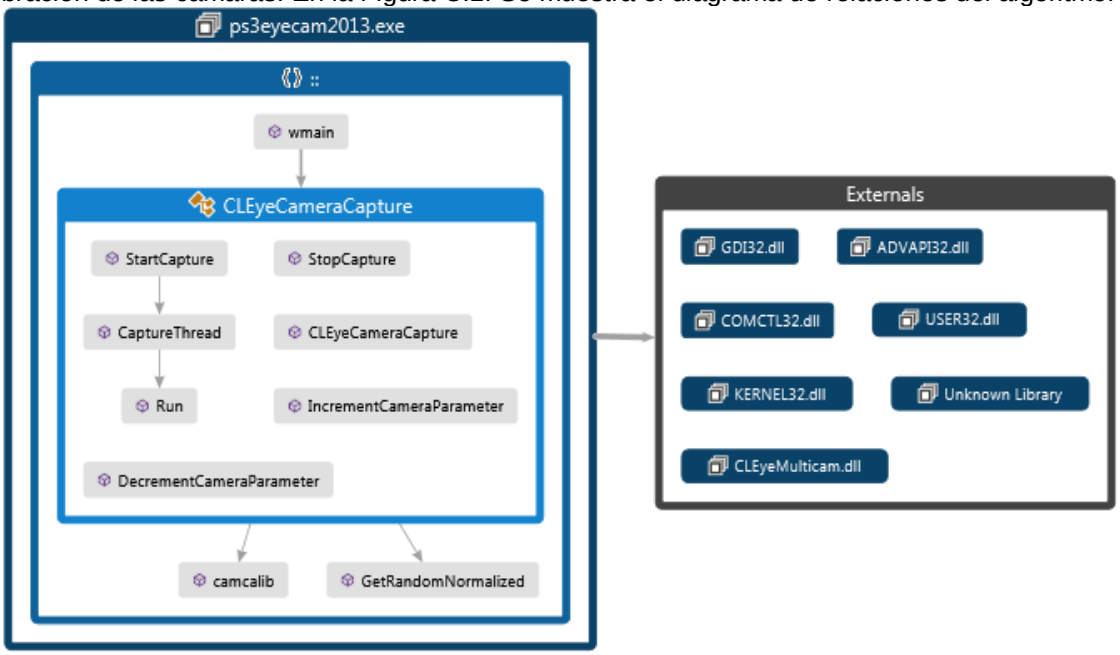

Figura C.1. Las cámaras son controlados por un sistema de OpenCV con memoria de anillo, si la calibración es necesaria se puede utilizar la función camcalib.

El código de la cámara es mostrado a continuación:

```
#ifdef _MSC_VER
```

```
#define _CRT_SECURE_NO_WARNINGS
#endif
//c api
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
#include <vector>
#include <string>
#include <sstream>
#include <windows.h>
#include <tchar.h>
//cl eye api
#include <CLEyeMulticam.h>
//opencv
#include "opencv2\highgui\highgui_c.h"
#include "opencv2\imgproc\imgproc_c.h"
#include "opencv2\legacy\compat.hpp"
#include "opencv2\core\core_c.h"
#include "opencv2\opencv.hpp"
#include "opencv2\calib3d\calib3d.hpp"

void camcalib(IplImage *cap){
	int numBoards; // número de tomas
	int board_w;   // número de esquinas w
	int board_h;   // número de esquinas h

	cv::Size board_sz = cvSize(board_w, board_h);
	int board_n = board_w*board_h;
	//localidad real de los puntos
	std::vector<std::vector<CvPoint3D32f> > objectPoints;
	//localidad en imagen de los puntos
	std::vector<std::vector<CvPoint2D32f> > imagePoints;
	std::vector<CvPoint2D32f> corners;

	cv::Mat gray, img;
	IplImage *VideoCapture;
	int success = 0;
	int k = 0;
	bool found = false;

	std::vector<CvPoint3D32f> obj;
	CvPoint3D32f objtemp;
	for (int j = 0; j<board_n; j++){
		objtemp.x = j / board_w;
		objtemp.y = j % board_w;
		objtemp.z = 0.0f;
		obj.push_back(objtemp);
	}
	//información de la calibración
	cv::cvtColor(img, gray, CV_BGR2GRAY);
	found = cv::findChessboardCorners(gray, board_sz, corners,
		cv::CALIB_CB_ADAPTIVE_THRESH + cv::CALIB_CB_NORMALIZE_IMAGE
		+ cv::CALIB_CB_FAST_CHECK);

	//Si los puntos son encontrados crea la información
	if (found){
```

```
			imagePoints.push_back(corners);
			objectPoints.push_back(obj);
			printf("Corners stored\n");
			success++;
		}

		//calibrar la imagen
		cv::Mat CM = cv::Mat(3, 3, CV_32FC1);
		cv::Mat D;
		std::vector<cv::Mat> rvecs, tvecs;

		CM.at<float>(0, 0) = 1;
		CM.at<float>(1, 1) = 1;

		cv::calibrateCamera(objectPoints, imagePoints, img.size(), CM, D, rvecs,
tvecs);

}
double GetRandomNormalized(){
		return (double)(rand() - (RAND_MAX >> 1)) / (double)(RAND_MAX >> 1);
}

// Modulo de captura
class CLEyeCameraCapture{
		int _counter;
		CHAR _windowName[256];
		CHAR _dir[256];
		GUID _cameraGUID;
		CLEyeCameraInstance _cam;
		CLEyeCameraColorMode _mode;
		CLEyeCameraResolution _resolution;
		float _fps;
		HANDLE _hThread;
		bool _running;
		IplImage * _pCapImageOLD;

		public:
			CLEyeCameraCapture(LPSTR windowName, LPSTR dirName, GUID
cameraGUID, CLEyeCameraColorMode mode, CLEyeCameraResolution resolution, float
fps) :
				_cameraGUID(cameraGUID), _cam(NULL), _mode(mode),
_resolution(resolution), _fps(fps), _running(false)
			{
				strcpy(_windowName, windowName);
				strcpy(_dir, dirName);
				_counter = 0;
				_pCapImageOLD = NULL;
			}

			bool StartCapture()
			{
				_running = true;
				cvNamedWindow(_windowName, CV_WINDOW_AUTOSIZE);
				// Inicio de la captura
				_hThread = CreateThread(NULL, 0,
&CLEyeCameraCapture::CaptureThread, this, 0, 0);
```

```
                if (_hThread == NULL)
                {
                        MessageBoxA(NULL, "Could not create capture thread",
"CLEyeMulticamTest", MB_ICONEXCLAMATION);
                        return false;
                }
                return true;
        }
        void StopCapture()
        {
                if (!_running)    return;
                _running = false;
                WaitForSingleObject(_hThread, 1000);
                cvDestroyWindow(_windowName);
        }
        void IncrementCameraParameter(int param)
        {
                if (!_cam)  return;
                printf("CLEyeGetCameraParameter %d\n",
CLEyeGetCameraParameter(_cam, (CLEyeCameraParameter)param));
                CLEyeSetCameraParameter(_cam, (CLEyeCameraParameter)param,
CLEyeGetCameraParameter(_cam, (CLEyeCameraParameter)param) + 10);
        }
        void DecrementCameraParameter(int param)
        {
                if (!_cam)  return;
                printf("CLEyeGetCameraParameter %d\n",
CLEyeGetCameraParameter(_cam, (CLEyeCameraParameter)param));
                CLEyeSetCameraParameter(_cam, (CLEyeCameraParameter)param,
CLEyeGetCameraParameter(_cam, (CLEyeCameraParameter)param) - 10);
        }
        void Run()
        {
                int w, h;
                IplImage *pCapImage;
                IplImage *pCapImageDest;
                IplImage *pCapImageDestGrey;
                CvScalar average;
                CvScalar std_dev;

                PBYTE pCapBuffer = NULL;
                // Crear instancia de la cámara
                _cam = CLEyeCreateCamera(_cameraGUID, _mode, _resolution,
_fps);
                if (_cam == NULL)       return;
                // Obtener dimensiones de la cámara
                CLEyeCameraGetFrameDimensions(_cam, w, h);
                // Crear el contenedor de OpenCV
      if (_mode == CLEYE_COLOR_PROCESSED || _mode == CLEYE_COLOR_RAW)
                {
                        pCapImage = cvCreateImage(cvSize(w, h), IPL_DEPTH_8U,
4);
                        _pCapImageOLD = cvCreateImage(cvGetSize(pCapImage),
IPL_DEPTH_8U, 4);
                        pCapImageDest = cvCreateImage(cvSize(w, h),
IPL_DEPTH_8U, 4);
```

```
                        pCapImageDestGrey = cvCreateImage(cvSize(w, h),
IPL_DEPTH_8U, 1);
                  }

                  else
                  {
                        pCapImage = cvCreateImage(cvSize(w, h), IPL_DEPTH_8U,
1);
                        _pCapImageOLD = cvCreateImage(cvGetSize(pCapImage),
IPL_DEPTH_8U, 1);
                        pCapImageDest = cvCreateImage(cvSize(w, h),
IPL_DEPTH_8U, 1);
                        pCapImageDestGrey = cvCreateImage(cvSize(w, h),
IPL_DEPTH_8U, 1);
                  }


                  // Parámetros de la cámara
                  CLEyeSetCameraParameter(_cam, CLEYE_GAIN, 0);
                  CLEyeSetCameraParameter(_cam, CLEYE_EXPOSURE, 511);
                  CLEyeSetCameraParameter(_cam, CLEYE_ZOOM,
(int)(GetRandomNormalized()*100.0));
                  CLEyeSetCameraParameter(_cam, CLEYE_ROTATION,
(int)(GetRandomNormalized()*300.0));

                  // Inicio de la captura
                  CLEyeCameraStart(_cam);
                  cvGetImageRawData(pCapImage, &pCapBuffer);
                  CLEyeCameraGetFrame(_cam, pCapBuffer);

                  _pCapImageOLD = cvCloneImage(pCapImage);
                  // loop de la captura
                  while (_running)
                  {
                        //guardar la imagen
                        CLEyeCameraGetFrame(_cam, pCapBuffer);

                        camcalib(pCapImage);

                        cvAbsDiff(pCapImage, _pCapImageOLD, pCapImageDest);
                        if (pCapImage)
                        {
                              _pCapImageOLD = cvCloneImage(pCapImage);
                        }


                        cvCvtColor(pCapImageDest, pCapImageDestGrey,
CV_BGR2GRAY);

                        cvAvgSdv(pCapImageDestGrey, &average, &std_dev);

                        if (std_dev.val[0] > 0.75)
                        {
                              std::string filename;
                              std::stringstream ss;
                              _counter = _counter + 1;
```

```
                              ss << _counter;
                              filename = _dir + ss.str() + ".bmp";
                              cvShowImage(_windowName, pCapImage);
                              if (_counter > 10)
                              {
                                    cvSaveImage(filename.c_str(), pCapImage);
                              }
                        }


                  }
                  // Detener la cámara
                  CLEyeCameraStop(_cam);
                  CLEyeDestroyCamera(_cam);
                  cvReleaseImage(&pCapImage);
                  _cam = NULL;
            }
            static DWORD WINAPI CaptureThread(LPVOID instance)
            {

                  srand(GetTickCount() + GetCurrentThreadId());

                  CLEyeCameraCapture *pThis = (CLEyeCameraCapture *)instance;
                  pThis->Run();
                  return 0;
            }
};

int _tmain(int argc, _TCHAR* argv[]){
      CLEyeCameraCapture *cam[2] = { NULL };
      srand(GetTickCount());
      // Numero de cámaras conectadas
      int numCams = CLEyeGetCameraCount();

      if (numCams == 0)
      {
            printf("No PS3Eye cameras detected\n");
            return -1;
      }
      printf("Found %d cameras\n", numCams);
      for (int i = 0; i < numCams; i++)
      {
            char windowName[64];
            char dirname[64];
            // Id de la cámara
            GUID guid = CLEyeGetCameraUUID(i);
            printf("Camera %d GUID:
[%08x-%04x-%04x-%02x%02x-%02x%02x%02x%02x%02x%02x]\n",
                  i + 1, guid.Data1, guid.Data2, guid.Data3,
                  guid.Data4[0], guid.Data4[1], guid.Data4[2],
                  guid.Data4[3], guid.Data4[4], guid.Data4[5],
                  guid.Data4[6], guid.Data4[7]);
            sprintf(windowName, "Camera Window %d", i + 1);
            sprintf(dirname, "C://stereotest//%d//", i + 1);
            // Objeto de la cámara
```

```
            cam[i] = new CLEyeCameraCapture(windowName, dirname, guid,
CLEYE_COLOR_RAW,
                  CLEYE_VGA, 30);
            printf("Starting capture on camera %d\n", i + 1);
            cam[i]->StartCapture();
      }
      printf("Use the following keys to change camera parameters:\n"
            "\t'1' - select camera 1\n"
            "\t'2' - select camera 2\n"
            "\t'g' - select gain parameter\n"
            "\t'e' - select exposure parameter\n"
            "\t'z' - select zoom parameter\n"
            "\t'r' - select rotation parameter\n"
            "\t'+' - increment selected parameter\n"
            "\t'-' - decrement selected parameter\n");
      // Salir del programa con esc
      CLEyeCameraCapture *pCam = NULL;
      int param = -1, key;

      while ((key = cvWaitKey(0)) != 0x1b)
      {
            switch (key)
            {
            case 'g':   case 'G':   printf("Parameter Gain\n");         param =
CLEYE_GAIN;       break;
            case 'e':   case 'E':   printf("Parameter Exposure\n");     param =
CLEYE_EXPOSURE;   break;
            case 'z':   case 'Z':   printf("Parameter Zoom\n");         param =
CLEYE_ZOOM;       break;
            case 'r':   case 'R':   printf("Parameter Rotation\n");     param =
CLEYE_ROTATION;   break;
            case '1':                     printf("Selected camera 1\n");
      pCam = cam[0];                break;
            case '2':                     printf("Selected camera 2\n");
      pCam = cam[1];                break;
            case '+':   if (pCam)   pCam->IncrementCameraParameter(param);
      break;
            case '-':   if (pCam)   pCam->DecrementCameraParameter(param);
      break;
            }
      }

      for (int i = 0; i < numCams; i++)
      {
            printf("Stopping capture on camera %d\n", i + 1);
            cam[i]->StopCapture();
            delete cam[i];
      }
      return 0;
}
```

Anexo D

Las imágenes de profundidad obtenidas por el método de estéreo – visión se convierten en mallas triangulares utilizando el algoritmo siguiente:

```
0001 %Programa de generación de mallas de superficie triangulares.
0002
0003 %Entrada: Archivos mat con las matrices de calibración.
0004 %Imagenes de las cámaras de estéreo visión.
0005
0006 %Salida: Archivos Ply con la malla de salida izquierda, derecha, frontal
y
0007 %compuesta
0008
0009 clear
0010
0011 load ('stereoParams_f.mat');
0012 ft = stereoParams_f.TranslationOfCamera2;
0013 fxo = ft(1);
0014 fyo = ft(2);
0015 fzo = ft(3);
0016 fR = stereoParams_f.RotationOfCamera2;
0017
0018 load ('stereoParams_ld.mat');
0019 ldt = stereoParams_ld.TranslationOfCamera2;
0020 ldxo = ldt(1);
0021 ldyo = ldt(2);
0022 ldzo = ldt(3);
0023 ldR = stereoParams_ld.RotationOfCamera2;
0024
0025 load ('stereoParams_li.mat');
0026 lit = stereoParams_li.TranslationOfCamera2;
0027 lixo = lit(1);
0028 liyo = lit(2);
0029 lizo = lit(3);
0030 liR = stereoParams_li.RotationOfCamera2;
0031
0032 load ('stereoParams_fd.mat');
0033 fdt = stereoParams_fd.TranslationOfCamera2;
0034 fdxo = fdt(1);
0035 fdyo = fdt(2);
0036 fdzo = fdt(3);
0037 fdR = stereoParams_fd.RotationOfCamera2;
0038
0039 load ('stereoParams_if.mat');
0040 ift = stereoParams_if.TranslationOfCamera2;
0041 ifxo = ift(1);
0042 ifyo = ift(2);
0043 ifzo = ift(3);
0044 ifR = stereoParams_if.RotationOfCamera2;
0045
0046 %Frente
0047
0048 % lectura del estéreo par.
0049 I1 = imread('..\arriba_f01.bmp');
0050 I2 = imread('..\abajo_f01.bmp');
0051
0052 % Rectificación de la imagen.
0053 [J1_f, J2_f] = rectifyStereoImages(I1, I2, stereoParams_f);
0054
0055 disparityMap = disparity(rgb2gray(J1_f), rgb2gray(J2_f));
```

```
0056
0057 pointCloud_f = reconstructScene(disparityMap, stereoParams_f);
0058
0059 vector_f(:,1) = reshape(pointCloud_f(:,:,1),...
0060                  [size(pointCloud_f,1)*size(pointCloud_f,2), 1]);
0061 vector_f(:,2) = reshape(pointCloud_f(:,:,2),...
0062                  [size(pointCloud_f,1)*size(pointCloud_f,2), 1]);
0063 vector_f(:,3) = reshape(pointCloud_f(:,:,3),...
0064                  [size(pointCloud_f,1)*size(pointCloud_f,2), 1]);
0065 vector_f(:,4) = reshape(J1_f(:,:,1),...
0066                  [size(J1_f,1)*size(J1_f,2), 1]);
0067 vector_f(:,5) = reshape(J1_f(:,:,2),...
0068                  [size(J1_f,1)*size(J1_f,2), 1]);
0069 vector_f(:,6) = reshape(J1_f(:,:,3),...
0070                  [size(J1_f,1)*size(J1_f,2), 1]);
0071
0072 vector_f(any(isnan(vector_f),2),:)=[];
0073 vector_f(vector_f(:,3)<100 | vector_f(:,3)>450,:)=[];
0074
0075
0076 vector_f(:,1:3) = vector_f(:,1:3)*fR;
0077
0078 vector_f(:,1) = vector_f(:,1) - fxo;
0079 vector_f(:,2) = vector_f(:,2) + fyo;
0080 vector_f(:,3) = vector_f(:,3) + fzo;
0081
0082 % lectura del estéreo par.
0083 I1 = imread('..\arriba_ld01.bmp');
0084 I2 = imread('..\abajo_ld01.bmp');
0085
0086 % Rectificación de la imagen.
0087 [J1_d, J2_d] = rectifyStereoImages(I1, I2, stereoParams_ld);
0088 disparityMap_ld = disparity(rgb2gray(J1_d), rgb2gray(J2_d));
0089 pointCloud_ld = reconstructScene(disparityMap_ld, stereoParams_ld);
0090
0091 vector_ld(:,1) = reshape(pointCloud_ld(:,:,1),...
0092                  [size(pointCloud_ld,1)*size(pointCloud_ld,2), 1]);
0093 vector_ld(:,2) = reshape(pointCloud_ld(:,:,2),...
0094                  [size(pointCloud_ld,1)*size(pointCloud_ld,2), 1]);
0095 vector_ld(:,3) = reshape(pointCloud_ld(:,:,3),...
0096                  [size(pointCloud_ld,1)*size(pointCloud_ld,2), 1]);
0097 vector_ld(:,4) = reshape(J1_d(:,:,1),...
0098                  [size(J1_d,1)*size(J1_d,2), 1]);
0099 vector_ld(:,5) = reshape(J1_d(:,:,2),...
0100                  [size(J1_d,1)*size(J1_d,2), 1]);
0101 vector_ld(:,6) = reshape(J1_d(:,:,3),...
0102                  [size(J1_d,1)*size(J1_d,2), 1]);
0103
0104 vector_ld(any(isnan(vector_ld),2),:)=[];
0105 vector_ld(vector_ld(:,3)<100 | vector_ld(:,3)>480,:)=[];
0106 vector_ld(:,1:3) = vector_ld(:,1:3)*fdR;
0107
0108 vector_ld(:,1) = vector_ld(:,1) + fdxo;
0109 vector_ld(:,2) = vector_ld(:,2) + fdyo;
0110 vector_ld(:,3) = vector_ld(:,3) + fdzo;
0111
```

```
% Lateral izquierdo
% lectura del estéreo par.
I1 = imread('..\arriba_li01.bmp');
I2 = imread('..\abajo_li01.bmp');

% Rectificación de la imagen.
[J1_i, J2_i] = rectifyStereoImages(I1, I2, stereoParams_li);
disparityMap_li = disparity(rgb2gray(J1_i), rgb2gray(J2_i));
pointCloud_li = reconstructScene(disparityMap_li, stereoParams_li);

vector_li(:,1) = reshape(pointCloud_li(:,:,1),...
                 [size(pointCloud_li,1)*size(pointCloud_li,2), 1]);
vector_li(:,2) = reshape(pointCloud_li(:,:,2),...
                 [size(pointCloud_li,1)*size(pointCloud_li,2), 1]);
vector_li(:,3) = reshape(pointCloud_li(:,:,3),...
                 [size(pointCloud_li,1)*size(pointCloud_li,2), 1]);
vector_li(:,4) = reshape(J1_i(:,:,1),...
                 [size(J1_i,1)*size(J1_i,2), 1]);
vector_li(:,5) = reshape(J1_i(:,:,2),...
                 [size(J1_i,1)*size(J1_i,2), 1]);
vector_li(:,6) = reshape(J1_i(:,:,3),...
                 [size(J1_i,1)*size(J1_i,2), 1]);

vector_li(any(isnan(vector_li),2),:)=[];
vector_li(vector_li(:,3)<200 | vector_li(:,3)>500,:)=[];

vector_li(:,1) = vector_li(:,1) - lixo;
vector_li(:,2) = vector_li(:,2) - liyo;
vector_li(:,3) = vector_li(:,3) - lizo;

vector_li(:,1:3) = vector_li(:,1:3)*liR;

vector_li(:,1:3) = vector_li(:,1:3)*ifR;

vector_li(:,1) = vector_li(:,1) + ifxo;
vector_li(:,2) = vector_li(:,2) + ifyo;
vector_li(:,3) = vector_li(:,3) + ifzo;

vector_T = cat(1,vector_f,vector_ld,vector_li);

vector_T(:,1)=vector_T(:,1)-min(vector_T(:,1));
vector_T(:,2)=vector_T(:,2)-min(vector_T(:,2));
vector_T(:,3)=vector_T(:,3)-min(vector_T(:,3));

vector_ld(:,1)=vector_ld(:,1)-min(vector_T(:,1));
vector_ld(:,2)=vector_ld(:,2)-min(vector_T(:,2));
vector_ld(:,3)=vector_ld(:,3)-min(vector_T(:,3));

vector_li(:,1)=vector_li(:,1)-min(vector_T(:,1));
vector_li(:,2)=vector_li(:,2)-min(vector_T(:,2));
vector_li(:,3)=vector_li(:,3)-min(vector_T(:,3));

vector_f(:,1)=vector_f(:,1)-min(vector_T(:,1));
vector_f(:,2)=vector_f(:,2)-min(vector_T(:,2));
vector_f(:,3)=vector_f(:,3)-min(vector_T(:,3));

```

```
0168
0169 a=size(vector_ld,1);
0170 fl = fopen('malla_ld.ply', 'w');
0171 fprintf(fl, 'ply\n');
0172 fprintf(fl, 'format ascii 1.0\n');
0173 fprintf(fl, 'element vertex %f\n', a);
0174 fprintf(fl, 'property float x\n');
0175 fprintf(fl, 'property float y\n');
0176 fprintf(fl, 'property float z\n');
0177 fprintf(fl, 'property uchar red\n');
0178 fprintf(fl, 'property uchar green\n');
0179 fprintf(fl, 'property uchar blue\n');
0180 fprintf(fl, 'end_header\n');
0181 fprintf(fl,'%g %g %g %g %g %g \n',vector_ld');
0182 fclose(fl);
0183 a=size(vector_li,1);
0184 fl = fopen('malla_li.ply', 'w');
0185 fprintf(fl, 'ply\n');
0186 fprintf(fl, 'format ascii 1.0\n');
0187 fprintf(fl, 'element vertex %f\n', a);
0188 fprintf(fl, 'property float x\n');
0189 fprintf(fl, 'property float y\n');
0190 fprintf(fl, 'property float z\n');
0191 fprintf(fl, 'property uchar red\n');
0192 fprintf(fl, 'property uchar green\n');
0193 fprintf(fl, 'property uchar blue\n');
0194 fprintf(fl, 'end_header\n');
0195 fprintf(fl,'%g %g %g %g %g %g \n',vector_li');
0196 fclose(fl);
0197 a=size(vector_f,1);
0198 fl = fopen('malla_f.ply', 'w');
0199 fprintf(fl, 'ply\n');
0200 fprintf(fl, 'format ascii 1.0\n');
0201 fprintf(fl, 'element vertex %f\n', a);
0202 fprintf(fl, 'property float x\n');
0203 fprintf(fl, 'property float y\n');
0204 fprintf(fl, 'property float z\n');
0205 fprintf(fl, 'property uchar red\n');
0206 fprintf(fl, 'property uchar green\n');
0207 fprintf(fl, 'property uchar blue\n');
0208 fprintf(fl, 'end_header\n');
0209 fprintf(fl,'%g %g %g %g %g %g \n',vector_f');
0210 fclose(fl);
0211 a=size(vector_T,1);
0212 fl = fopen('malla_T.ply', 'w');
0213 fprintf(fl, 'ply\n');
0214 fprintf(fl, 'format ascii 1.0\n');
0215 fprintf(fl, 'element vertex %f\n', a);
0216 fprintf(fl, 'property float x\n');
0217 fprintf(fl, 'property float y\n');
0218 fprintf(fl, 'property float z\n');
0219 fprintf(fl, 'property uchar red\n');
0220 fprintf(fl, 'property uchar green\n');
0221 fprintf(fl, 'property uchar blue\n');
0222 fprintf(fl, 'end_header\n');
0223 fprintf(fl,'%g %g %g %g %g %g \n',vector_T');
```

```
0224 fclose(fl);
```

Anexo E

El análisis de los datos obtenidos por el sistema de estéreo - visión múltiple se obtienen utilizando un algoritmo programado en Matlab el cual tiene como entrada los archivos tipo ".landmarkAscii" y como salida las tablas de varianza así como las gráficas de las proyecciones en los ejes de proyección del método PCA:

```
%Programa de estudio estadístico de rostros humanos.
%El programa obtiene la reducción de dimensionalidad por PCA y
%la clasificación por grupos de cluster obtenidos del método k-menas.

0001 %Programa de estudio estadístico de rostros humanos.
0002 %El programa obtiene la reducción de dimensionalidad por PCA y
0003 %la clasificación por grupos de cluster obtenidos del método k-menas.
0004
0005 %Entrada: Folder con los archivos tipo .landmarkAscii con la información
de
0006 %los sujetos
0007
0008
0009 % Puntos localizados
0010 % 1 glabella    g
0011 % 2 nasion  n
0012 % 3 endocathion derecho en der
0013 % 4 endocathion izquierdo   en izq
0014 % 5 exocathion derecho  ex der
0015 % 6 exocathion izquierdo    ex izq
0016 % 7 alare derecho   al der
0017 % 8 alare izquierdo al izq
0018 % 9 subnasale   sn
0019 % 10    labiale superius    ls
0020 % 11    stomion sto
0021 % 12    labiale inferius    li
0022 % 13    gnathion    gn
0023 % 14    cheilion derecho    ch der
0024 % 15    cheilion izquierdo  ch izq
0025 % 16    sublabiale  sl
0026 % 17    pogonion pog
0027 % 18    pronasale prn
0028 % 19    zygion derecha      zy der
0029 % 20    zygion izquierda    zy izq
0030
0031 header0={'0';'n';'en der'; 'en izq'; 'ex der'; 'ex izq'; 'al der'; ...
0032         'al izq'; 'sn'; 'ls'; 'sto'; 'li'; 'gn'; 'ch der'; 'ch
izq';'sl'; ...
0033         'pog'; 'prn';'zy der'; 'zy izq'};
0034
0035 clear
0036 dirroot='D:\OneDrive\experiments\caras';
0037
0038 cd(dirroot);
0039 dirinfo=dir();
0040 dirinfo(~[dirinfo.isdir])=[];
0041 tf=ismember ({dirinfo.name},{'.','..'});
0042 dirinfo(tf)=[];
```

```
0043 subdirinfo=cell(size(dirinfo,1),1);
0044 size (dirinfo,1);
0045 for K=1 : size (dirinfo,1)
0046     thisdir=dirinfo(K).name;
0047     subdirinfo{K}=dir(fullfile(thisdir, '*.landmarkAscii'));
0048 end
0049 for K=1 : size (dirinfo,1)
0050     thisdir=dirinfo(K).name;
0051     filename=fullfile(thisdir,subdirinfo{K,1}.name);
0052     asciidata=importdata(filename,' ',14);
0053     facelandmarks{K,1}=asciidata.data;
0054 end
0055
0056 % Orden de las distancias por columna
0057 % 1 n-sto (2-11)
0058 % 2 n-gn (2-13)
0059 % 3 sn-gn (9-13)
0060 % 4 sto-gn (11-13)
0061 % 5 al-al (7-8)
0062 % 6 n-sn (2-9)
0063 % 7 sn-sto (9-11)
0064 % 8 ch-ch (14-15)
0065 % 9 sn-ls (9-10)
0066 % 10 ls-sto (10-11)
0067 % 11 sto-li (11-12)
0068 % 12 ex-ex (5-6)
0069 % 13 en-en (3-4)
0070 % 14 ls-li (10-12)
0071 % 15 sto-sl (11-16)
0072 % 16 sl-gn (16-13)
0073 % 17 ex-li (6-12)
0074 % 18 al-prn (8-18)
0075 % 19 zy-zy (19-20)
0076 % 20 ex(der)-sn (9-5)
0077 % 21 ex(izq)-sn (9-6)
0078 % 22 ch(der)-sn (9-14)
0079 % 23 ch(izq)-sn (9-15)
0080 % 24 ex(der)-li (12-5)
0081 % 25 ex(izq)-li (12-6)
0082 % 26 pog-n (2-17)
0083 % 27 prn-n (2-18)
0084 % 28 pog-prn (18-17)
0085
0086 for K=1 : size (dirinfo,1),
0087
0088     colnum=1;
0089
0090     %n-sto
0091     indice1=2;
0092     indice2=11;
0093     distancias(K,colnum)=sqrt(...
0094         (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0095         +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
```

```
0096          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0097      colnum=colnum+1;
0098
0099      %n-gn
0100      indice1=2;
0101      indice2=13;
0102      distancias(K,colnum)=sqrt(...
0103          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0104          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0105          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0106      colnum=colnum+1;
0107
0108      %sn-gn
0109      indice1=9;
0110      indice2=13;
0111      distancias(K,colnum)=sqrt(...
0112          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0113          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0114          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0115      colnum=colnum+1;
0116
0117      %sto-gn
0118      indice1=11;
0119      indice2=13;
0120      distancias(K,colnum)=sqrt(...
0121          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0122          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0123          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0124      colnum=colnum+1;
0125
0126      %al-al
0127      indice1=7;
0128      indice2=8;
0129      distancias(K,colnum)=sqrt(...
0130          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0131          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0132          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0133      colnum=colnum+1;
0134
0135      %n-sn
0136      indice1=2;
0137      indice2=9;
0138      distancias(K,colnum)=sqrt(...
```

```
0139          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0140          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0141          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0142      colnum=colnum+1;
0143
0144      %sn-sto
0145      indice1=9;
0146      indice2=11;
0147      distancias(K,colnum)=sqrt(...
0148          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0149          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0150          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0151      colnum=colnum+1;
0152
0153      %ch-ch
0154      indice1=14;
0155      indice2=15;
0156      distancias(K,colnum)=sqrt(...
0157          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0158          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0159          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0160      colnum=colnum+1;
0161
0162      %sn-ls
0163      indice1=9;
0164      indice2=10;
0165      distancias(K,colnum)=sqrt(...
0166          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0167          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0168          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0169      colnum=colnum+1;
0170
0171      %ls-sto
0172      indice1=10;
0173      indice2=11;
0174      distancias(K,colnum)=sqrt(...
0175          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0176          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0177          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0178      colnum=colnum+1;
0179
```

```
0180      %sto-li
0181      indice1=11;
0182      indice2=12;
0183      distancias(K,colnum)=sqrt(...
0184          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0185          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0186          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0187      colnum=colnum+1;
0188
0189      %ex-ex
0190      indice1=5;
0191      indice2=6;
0192      distancias(K,colnum)=sqrt(...
0193          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0194          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0195          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0196      colnum=colnum+1;
0197
0198      %en-en
0199      indice1=3;
0200      indice2=4;
0201      distancias(K,colnum)=sqrt(...
0202          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0203          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0204          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0205      colnum=colnum+1;
0206
0207      %ls-li
0208      indice1=10;
0209      indice2=12;
0210      distancias(K,colnum)=sqrt(...
0211          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0212          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0213          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0214      colnum=colnum+1;
0215
0216      %sto-sl
0217      indice1=11;
0218      indice2=16;
0219      distancias(K,colnum)=sqrt(...
0220          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0221          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
```

```
0222          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0223      colnum=colnum+1;
0224
0225      %sl-gn
0226      indice1=16;
0227      indice2=13;
0228      distancias(K,colnum)=sqrt(...
0229          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0230          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0231          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0232      colnum=colnum+1;
0233
0234      %ex-li
0235      indice1=6;
0236      indice2=12;
0237      distancias(K,colnum)=sqrt(...
0238          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0239          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0240          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0241      colnum=colnum+1;
0242
0243      %al-prn
0244      indice1=8;
0245      indice2=18;
0246      distancias(K,colnum)=sqrt(...
0247          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0248          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0249          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0250      colnum=colnum+1;
0251
0252      %zy-zy
0253      indice1=19;
0254      indice2=20;
0255      distancias(K,colnum)=sqrt(...
0256          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0257          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0258          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0259      colnum=colnum+1;
0260
0261      %ex(der)-sn
0262      indice1=9;
0263      indice2=5;
0264      distancias(K,colnum)=sqrt(...
```

```
0265          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0266          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0267          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0268      colnum=colnum+1;
0269
0270      %ex(izq)-sn
0271      indice1=9;
0272      indice2=6;
0273      distancias(K,colnum)=sqrt(...
0274          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0275          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0276          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0277      colnum=colnum+1;
0278
0279      %ch(der)-sn
0280      indice1=9;
0281      indice2=14;
0282      distancias(K,colnum)=sqrt(...
0283          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0284          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0285          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0286      colnum=colnum+1;
0287
0288      %ch(izq)-sn
0289      indice1=9;
0290      indice2=15;
0291      distancias(K,colnum)=sqrt(...
0292          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0293          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0294          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0295      colnum=colnum+1;
0296
0297      %ex(der)-li (12-5)
0298      indice1=12;
0299      indice2=5;
0300      distancias(K,colnum)=sqrt(...
0301          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0302          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0303          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0304      colnum=colnum+1;
0305
```

```
0306      %ex(izq)-li (12-6)
0307      indice1=12;
0308      indice2=6;
0309      distancias(K,colnum)=sqrt(...
0310          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0311          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0312          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0313      colnum=colnum+1;
0314
0315      %pog-n
0316      indice1=2;
0317      indice2=17;
0318      distancias(K,colnum)=sqrt(...
0319          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0320          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0321          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0322      colnum=colnum+1;
0323
0324      %prn-n
0325      indice1=2;
0326      indice2=18;
0327      distancias(K,colnum)=sqrt(...
0328          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0329          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0330          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0331      colnum=colnum+1;
0332
0333      %pog-prn
0334      indice1=18;
0335      indice2=17;
0336      distancias(K,colnum)=sqrt(...
0337          (facelandmarks{K,1}(indice1,1)-
facelandmarks{K,1}(indice2,1))^2 ...
0338          +(facelandmarks{K,1}(indice1,2)-
facelandmarks{K,1}(indice2,2))^2 ...
0339          +(facelandmarks{K,1}(indice1,3)-
facelandmarks{K,1}(indice2,3))^2);
0340      colnum=colnum+1;
0341
0342  end
0343  % Orden de los índices
0344  % 1 sujeto
0345  % 2 upper face - face height (1/2)
0346  % 3 lower face - face height (3/2)
0347  % 4 mandibulo face height (4/2)
0348  % 5 madibulo - upper face height (4/1)
0349  % 6 mandibulo - lower face height (4/3)
```

```
% 7 upper lip height - mouth width (7/8)
% 8 cutaneous - total upper lip height (9/7)
% 9 vermilion - total upper lip height (10/7)
% 10    vermilion - cutaneous upper lip height (10/9)
% 11    upper lip vertical contour (7/(9+10))
% 12    vemilion height (10/11)
% 13    upper face height - binocular width (1/12)
% 14    intercanthal - nasal width (13/5)
% 15    nose - face height (6/2)
% 16    nose - mouth width (5/8)
% 17    upper lip - upper face height (7/1)
% 18    upper lip - madible height (7/4)
% 19    upper lip - nose height (7/6)
% 20    nasal width (5/6)
% 21    buccal (14/8)
% 22    intercanthal (13/12)
% 23    chin-mandible height (16/4)
% 24    upper lip lenght (7/3)
% 25    lower lip lenght (15/3)
% 26    chin height (16/3)
% 27    lower-lip mandible height (15/4)
% 38    lower-lip chin height (15/16)
% 39    labio orbitale triangle (17/12)
% 30    nasal projection (18/6) revisarlo

% Para los ángulos
% 35    ex(der)-sn
% 36    ex(izq)-sn
% 37    ch(der)-sn
% 38    ch(izq)-sn
% 39    ex(der)-li
% 40    ex(izq)-li
for idx=1:28
    header2{idx,1}=strcat('d', num2str(idx));
end
for idx=1:28
    header3{idx,1}=strcat('i', num2str(idx));
end
for idx=1:6
    header4{idx,1}=strcat('a', num2str(idx));
end
for idx=1:34
    sujeto{idx,1}=strcat('s', num2str(idx));
end

j=1;

%upper face - face height
ind1=1;
ind2=2;
indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
j=j+1;

%lower face - face height
ind1=3;
ind2=2;
```

```
0406 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0407 j=j+1;
0408
0409 %mandibulo face height
0410 ind1=4;
0411 ind2=2;
0412 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0413 j=j+1;
0414
0415 %madibulo - upper face height
0416 ind1=4;
0417 ind2=1;
0418 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0419 j=j+1;
0420
0421 %mandibulo - lower face height
0422 ind1=4;
0423 ind2=3;
0424 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0425 j=j+1;
0426
0427 %upper lip height - mouth width
0428 ind1=7;
0429 ind2=8;
0430 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0431 j=j+1;
0432
0433 %cutaneous - total upper lip height
0434 ind1=9;
0435 ind2=7;
0436 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0437 j=j+1;
0438
0439 %vermilion - total upper lip height
0440 ind1=10;
0441 ind2=7;
0442 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0443 j=j+1;
0444
0445 %vermilion - cutaneous upper lip height
0446 ind1=10;
0447 ind2=9;
0448 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0449 j=j+1;
0450
0451 %upper lip vertical contour
0452 ind1=7;
0453 ind2=9;
0454 ind3=10;
0455 indice(:,j)=distancias(:,ind1)./(distancias(:,ind2)+distancias(:,ind3));
0456 j=j+1;
0457
0458 %vemilion height
0459 ind1=10;
0460 ind2=11;
0461 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
```

```
0462 j=j+1;
0463
0464 %upper face height - binocular width
0465 ind1=1;
0466 ind2=12;
0467 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0468 j=j+1;
0469
0470 %intercanthal - nasal width
0471 ind1=13;
0472 ind2=5;
0473 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0474 j=j+1;
0475
0476 %nose - face height
0477 ind1=6;
0478 ind2=2;
0479 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0480 j=j+1;
0481
0482 %nose - mouth height
0483 ind1=5;
0484 ind2=8;
0485 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0486 j=j+1;
0487
0488 %upper lip - upper face height
0489 ind1=7;
0490 ind2=1;
0491 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0492 j=j+1;
0493
0494 %upper lip - madible height
0495 ind1=7;
0496 ind2=4;
0497 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0498 j=j+1;
0499
0500 %upper lip - nose height
0501 ind1=7;
0502 ind2=6;
0503 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0504 j=j+1;
0505
0506 %nasal width
0507 ind1=5;
0508 ind2=6;
0509 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0510 j=j+1;
0511
0512 %buccal
0513 ind1=14;
0514 ind2=8;
0515 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0516 j=j+1;
0517
```

```
0518 %intercanthal
0519 ind1=13;
0520 ind2=12;
0521 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0522 j=j+1;
0523
0524 %chin-mandible height
0525 ind1=16;
0526 ind2=4;
0527 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0528 j=j+1;
0529
0530 %upper lip lenght
0531 ind1=7;
0532 ind2=3;
0533 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0534 j=j+1;
0535
0536 %lower lip lenght
0537 ind1=15;
0538 ind2=3;
0539 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0540 j=j+1;
0541
0542 %chin height
0543 ind1=16;
0544 ind2=3;
0545 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0546 j=j+1;
0547
0548 %lower-lip mandible height
0549 ind1=15;
0550 ind2=4;
0551 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0552 j=j+1;
0553
0554 %lower-lip chin height
0555 ind1=15;
0556 ind2=16;
0557 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0558 j=j+1;
0559
0560 %labio orbitale triangle
0561 ind1=17;
0562 ind2=12;
0563 indice(:,j)=distancias(:,ind1)./distancias(:,ind2);
0564
0565 j=1;
0566 %Ángulos
0567 for K=1 : size (dirinfo,1),
0568
0569     %labio-orbital
0570     indice1=9;
0571     indice2=5;
0572
0573     ux=facelandmarks{K,1}(indice2,1)-facelandmarks{K,1}(indice1,1);
```

```
0574      uy=facelandmarks{K,1}(indice2,2)-facelandmarks{K,1}(indice1,2);
0575      uz=facelandmarks{K,1}(indice2,3)-facelandmarks{K,1}(indice1,3);
0576
0577      u=[ux,uy,uz];
0578
0579      indice1=9;
0580      indice2=6;
0581
0582      vx=facelandmarks{K,1}(indice2,1)-facelandmarks{K,1}(indice1,1);
0583      vy=facelandmarks{K,1}(indice2,2)-facelandmarks{K,1}(indice1,2);
0584      vz=facelandmarks{K,1}(indice2,3)-facelandmarks{K,1}(indice1,3);
0585
0586      v=[vx,vy,vz];
0587
0588      costheta=dot(u,v)/(norm(u)*norm(v));
0589
0590      angulo(K,j)= acos(costheta)*180/pi;
0591  end
0592  j=j+1;
0593  for K=1 : size (dirinfo,1),
0594      %naso-orbital
0595      indice1=9;
0596      indice2=14;
0597
0598      ux=facelandmarks{K,1}(indice2,1)-facelandmarks{K,1}(indice1,1);
0599      uy=facelandmarks{K,1}(indice2,2)-facelandmarks{K,1}(indice1,2);
0600      uz=facelandmarks{K,1}(indice2,3)-facelandmarks{K,1}(indice1,3);
0601
0602      u=[ux,uy,uz];
0603
0604      indice1=9;
0605      indice2=15;
0606
0607      vx=facelandmarks{K,1}(indice2,1)-facelandmarks{K,1}(indice1,1);
0608      vy=facelandmarks{K,1}(indice2,2)-facelandmarks{K,1}(indice1,2);
0609      vz=facelandmarks{K,1}(indice2,3)-facelandmarks{K,1}(indice1,3);
0610
0611      v=[vx,vy,vz];
0612
0613      costheta=dot(u,v)/(norm(u)*norm(v));
0614
0615      angulo(K,j)= acos(costheta)*180/pi;
0616  end
0617  j=j+1;
0618  for K=1 : size (dirinfo,1),
0619      %naso-chelial
0620      indice1=12;
0621      indice2=5;
0622
0623      ux=facelandmarks{K,1}(indice2,1)-facelandmarks{K,1}(indice1,1);
0624      uy=facelandmarks{K,1}(indice2,2)-facelandmarks{K,1}(indice1,2);
0625      uz=facelandmarks{K,1}(indice2,3)-facelandmarks{K,1}(indice1,3);
0626
0627      u=[ux,uy,uz];
0628
0629      indice1=12;
```

```
0630      indice2=6;
0631
0632      vx=facelandmarks{K,1}(indice2,1)-facelandmarks{K,1}(indice1,1);
0633      vy=facelandmarks{K,1}(indice2,2)-facelandmarks{K,1}(indice1,2);
0634      vz=facelandmarks{K,1}(indice2,3)-facelandmarks{K,1}(indice1,3);
0635
0636      v=[vx,vy,vz];
0637
0638      costheta=dot(u,v)/(norm(u)*norm(v));
0639
0640      angulo(K,j)= acos(costheta)*180/pi;
0641  end
0642  j=j+1;
0643  for K=1 : size (dirinfo,1),
0644      %naso-facial
0645      indice1=2;
0646      indice2=17;
0647
0648      ux=facelandmarks{K,1}(indice2,1)-facelandmarks{K,1}(indice1,1);
0649      uy=facelandmarks{K,1}(indice2,2)-facelandmarks{K,1}(indice1,2);
0650      uz=facelandmarks{K,1}(indice2,3)-facelandmarks{K,1}(indice1,3);
0651
0652      u=[ux,uy,uz];
0653
0654      indice1=2;
0655      indice2=18;
0656
0657      vx=facelandmarks{K,1}(indice2,1)-facelandmarks{K,1}(indice1,1);
0658      vy=facelandmarks{K,1}(indice2,2)-facelandmarks{K,1}(indice1,2);
0659      vz=facelandmarks{K,1}(indice2,3)-facelandmarks{K,1}(indice1,3);
0660
0661      v=[vx,vy,vz];
0662
0663      costheta=dot(u,v)/(norm(u)*norm(v));
0664
0665      angulo(K,j)= acos(costheta)*180/pi;
0666  end
0667  j=j+1;
0668  for K=1 : size (dirinfo,1),
0669      %naso-mental
0670      indice1=18;
0671      indice2=2;
0672
0673      ux=facelandmarks{K,1}(indice2,1)-facelandmarks{K,1}(indice1,1);
0674      uy=facelandmarks{K,1}(indice2,2)-facelandmarks{K,1}(indice1,2);
0675      uz=facelandmarks{K,1}(indice2,3)-facelandmarks{K,1}(indice1,3);
0676
0677      u=[ux,uy,uz];
0678
0679      indice1=18;
0680      indice2=17;
0681
0682      vx=facelandmarks{K,1}(indice2,1)-facelandmarks{K,1}(indice1,1);
0683      vy=facelandmarks{K,1}(indice2,2)-facelandmarks{K,1}(indice1,2);
0684      vz=facelandmarks{K,1}(indice2,3)-facelandmarks{K,1}(indice1,3);
0685
```

```
0686      v=[vx,vy,vz];
0687
0688      costheta=dot(u,v)/(norm(u)*norm(v));
0689
0690      angulo(K,j)= acos(costheta)*180/pi;
0691  end
0692  j=j+1;
0693  for K=1 : size (dirinfo,1),
0694      %mento-facial
0695      indice1=17;
0696      indice2=2;
0697
0698      ux=facelandmarks{K,1}(indice2,1)-facelandmarks{K,1}(indice1,1);
0699      uy=facelandmarks{K,1}(indice2,2)-facelandmarks{K,1}(indice1,2);
0700      uz=facelandmarks{K,1}(indice2,3)-facelandmarks{K,1}(indice1,3);
0701
0702      u=[ux,uy,uz];
0703
0704      indice1=17;
0705      indice2=18;
0706
0707      vx=facelandmarks{K,1}(indice2,1)-facelandmarks{K,1}(indice1,1);
0708      vy=facelandmarks{K,1}(indice2,2)-facelandmarks{K,1}(indice1,2);
0709      vz=facelandmarks{K,1}(indice2,3)-facelandmarks{K,1}(indice1,3);
0710
0711      v=[vx,vy,vz];
0712
0713      costheta=dot(u,v)/(norm(u)*norm(v));
0714
0715      angulo(K,j)= acos(costheta)*180/pi;
0716  end
0717  j=j+1;
0718
0719  %PCA distancias
0720  figure()
0721  boxplot(distancias,'orientation','horizontal','labels',header2)
0722
0723  C=corr(distancias,distancias);
0724
0725  w = 1./var(distancias);
0726  [wcoeff,score,latent,tsquared,explained]=pca(distancias,...
0727      'VariableWeights',w);
0728  coefforth=diag(sqrt(w))*wcoeff;
0729
0730  figure('visible','off','units','normalized','outerposition',[0.05 0.05
0.95 0.95])
0731  pareto(explained)
0732  xlabel('Componentes principales')
0733  ylabel('% de varianza explicada')
0734  set(gca,'FontSize',12,'fontWeight','bold')
0735  set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
0736  fig = gcf;
0737  print(fig, '-dpng', ['paretodis','.png']);
0738  close
0739
0740  VectorVals = coefforth(:,1:3);
```

```
VectorValsdist = (sqrt(sum(VectorVals'.^2)))';
VectorValsdist(:,2)=1:size(VectorValsdist,1);
VectorValsdist = sortrows(VectorValsdist,1);

[st2,index] = sort(tsquared,'descend'); % sort in descending order
listdif(:,1)=sujeto(index(1:10,1),:);

figure('visible','off','units','normalized','outerposition',[0.05 0.05
0.95 0.95])
biplot(coefforth(:,1:3),'scores',score(:,1:3), 'varlabels',header2);
xlabel('Componente principal 1')
ylabel('Componente principal 2')
zlabel('Componente principal 3')
%axis([-.18 0.4 -.2 .23 -.22 .4]);
view([30 40]);
set(gca,'FontSize',12,'fontWeight','bold')
set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
fig = gcf;
print(fig, '-dpng', ['biplotPCAdis','.png']);
close

list=VectorValsdist(end-9:end,2);
figure('visible','off','units','normalized','outerposition',[0.05 0.05
0.95 0.95])

biplot(VectorVals(list,:),'scores',score(:,1:3),'varlabels',header2(list,1));
xlabel('Componente principal 1')
ylabel('Componente principal 2')
zlabel('Componente principal 3')
%axis([-.18 0.4 -.2 .23 -.22 .4]);
view([30 40]);
set(gca,'FontSize',12,'fontWeight','bold')
set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
fig = gcf;
print(fig, '-dpng', ['biplotPCATopdis','.png']);
close
topten(:,1) = header2(list,1);

list=VectorValsdist(1:10,2);
figure('visible','off','units','normalized','outerposition',[0.05 0.05
0.95 0.95])

biplot(VectorVals(list,:),'scores',score(:,1:3),'varlabels',header2(list,1));
xlabel('Componente principal 1')
ylabel('Componente principal 2')
zlabel('Componente principal 3')
%axis([-.002 4 -.007 0.012 -.013 .02]);
view([30 45]);
set(gca,'FontSize',12,'fontWeight','bold')
set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
fig = gcf;
print(fig, '-dpng', ['biplotPCABotdis','.png']);
close
lowten(:,1) = header2(list,1);


```

```
w = 1./var(indice);
[wcoeff,score,latent,tsquared,explained]=pca(indice,...
    'VariableWeights',w);
coefforth=diag(sqrt(w))*wcoeff;

figure('visible','off','units','normalized','outerposition',[0.05 0.05
0.95 0.95])
pareto(explained)
xlabel('Componentes principales')
ylabel('% de varianza explicada')
set(gca,'FontSize',12,'fontWeight','bold')
set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
fig = gcf;
print(fig, '-dpng', ['paretoind','.png']);
close

VectorVals = coefforth(:,1:3);
VectorValsdist = (sqrt(sum(VectorVals'.^2)))';
VectorValsdist(:,2)=1:size(VectorValsdist,1);
VectorValsdist = sortrows(VectorValsdist,1);

[st2,index] = sort(tsquared,'descend'); % sort in descending order
listdif(:,2)=sujeto(index(1:10,1),:);

figure('visible','off','units','normalized','outerposition',[0.05 0.05
0.95 0.95])
biplot(coefforth(:,1:3),'scores',score(:,1:3), 'varlabels',header3);
xlabel('Componente principal 1')
ylabel('Componente principal 2')
zlabel('Componente principal 3')
%axis([-.18 0.4 -.2 .23 -.22 .4]);
view([30 40]);
set(gca,'FontSize',12,'fontWeight','bold')
set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
fig = gcf;
print(fig, '-dpng', ['biplotPCAind','.png']);
close

list=VectorValsdist(end-9:end,2);
figure('visible','off','units','normalized','outerposition',[0.05 0.05
0.95 0.95])

biplot(VectorVals(list,:),'scores',score(:,1:3),'varlabels',header3(list,1));
xlabel('Componente principal 1')
ylabel('Componente principal 2')
zlabel('Componente principal 3')
%axis([-.18 0.4 -.2 .23 -.22 .4]);
view([30 40]);
set(gca,'FontSize',12,'fontWeight','bold')
set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
fig = gcf;
print(fig, '-dpng', ['biplotPCATopind','.png']);
close
topten(:,2) = header3(list,1);

list=VectorValsdist(1:10,2);
```

```
0844 figure('visible','off','units','normalized','outerposition',[0.05 0.05
0.95 0.95])
0845
biplot(VectorVals(list,:),'scores',score(:,1:3),'varlabels',header3(list,1));
0846 xlabel('Componente principal 1')
0847 ylabel('Componente principal 2')
0848 zlabel('Componente principal 3')
0849 %axis([-.002 4 -.007 0.012 -.013 .02]);
0850 view([30 45]);
0851 set(gca,'FontSize',12,'fontWeight','bold')
0852 set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
0853 fig = gcf;
0854 print(fig, '-dpng', ['biplotPCABotind','.png']);
0855 close
0856 lowten(:,2) = header3(list,1);
0857
0858 w = 1./var(angulo);
0859 [wcoeff,score,latent,tsquared,explained]=pca(angulo,...
0860     'VariableWeights',w);
0861 coefforth=diag(sqrt(w))*wcoeff;
0862
0863 figure('visible','off','units','normalized','outerposition',[0.05 0.05
0.95 0.95])
0864 pareto(explained)
0865 xlabel('Componentes principales')
0866 ylabel('% de varianza explicada')
0867 set(gca,'FontSize',12,'fontWeight','bold')
0868 set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
0869 fig = gcf;
0870 print(fig, '-dpng', ['paretoang','.png']);
0871 close
0872
0873 VectorVals = coefforth(:,1:3);
0874 VectorValsdist = (sqrt(sum(VectorVals'.^2)))';
0875 VectorValsdist(:,2)=1:size(VectorValsdist,1);
0876 VectorValsdist = sortrows(VectorValsdist,1);
0877
0878 [st2,index] = sort(tsquared,'descend'); % sort in descending order
0879 listdif(:,3)=sujeto(index(1:10,1),:);
0880
0881 figure('visible','off','units','normalized','outerposition',[0.05 0.05
0.95 0.95])
0882 biplot(coefforth(:,1:3),'scores',score(:,1:3), 'varlabels',header4);
0883 xlabel('Componente principal 1')
0884 ylabel('Componente principal 2')
0885 zlabel('Componente principal 3')
0886 %axis([-.18 0.4 -.2 .23 -.22 .4]);
0887 view([30 40]);
0888 set(gca,'FontSize',12,'fontWeight','bold')
0889 set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
0890 fig = gcf;
0891 print(fig, '-dpng', ['biplotPCAang','.png']);
0892 close
0893
0894 list=VectorValsdist(end-1:end,2);
```

```
0895 figure('visible','off','units','normalized','outerposition',[0.05 0.05
0.95 0.95])
0896
biplot(VectorVals(list,:),'scores',score(:,1:3),'varlabels',header4(list,1));
0897 xlabel('Componente principal 1')
0898 ylabel('Componente principal 2')
0899 zlabel('Componente principal 3')
0900 %axis([-.18 0.4 -.2 .23 -.22 .4]);
0901 view([30 40]);
0902 set(gca,'FontSize',12,'fontWeight','bold')
0903 set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
0904 fig = gcf;
0905 print(fig, '-dpng', ['biplotPCATopang','.png']);
0906 close
0907 %topten(:,3) = header2(list,1);
0908
0909 list=VectorValsdist(1:2,2);
0910 figure('visible','off','units','normalized','outerposition',[0.05 0.05
0.95 0.95])
0911
biplot(VectorVals(list,:),'scores',score(:,1:3),'varlabels',header4(list,1));
0912 xlabel('Componente principal 1')
0913 ylabel('Componente principal 2')
0914 zlabel('Componente principal 3')
0915 %axis([-.002 4 -.007 0.012 -.013 .02]);
0916 view([30 45]);
0917 set(gca,'FontSize',12,'fontWeight','bold')
0918 set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
0919 fig = gcf;
0920 print(fig, '-dpng', ['biplotPCABotang','.png']);
0921 close
0922 %lowten(:,3) = header2(list,1);
0923
0924
0925
0926 Total = [distancias indice angulo];
0927 HTotal = [header2; header3; header4];
0928
0929 w = 1./var(Total);
0930 [wcoeff,score,latent,tsquared,explained]=pca(Total,...
0931     'VariableWeights',w);
0932 coefforth=diag(sqrt(w))*wcoeff;
0933
0934 figure('visible','off','units','normalized','outerposition',[0.05 0.05
0.95 0.95])
0935 pareto(explained)
0936 xlabel('Componentes principales')
0937 ylabel('% de varianza explicada')
0938 set(gca,'FontSize',12,'fontWeight','bold')
0939 set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
0940 fig = gcf;
0941 print(fig, '-dpng', ['paretoT','.png']);
0942 close
0943
0944 VectorVals = coefforth(:,1:3);
0945 VectorValsdist = (sqrt(sum(VectorVals'.^2)))';
```

```
VectorValsdist(:,2)=1:size(VectorValsdist,1);
VectorValsdist = sortrows(VectorValsdist,1);

[st2,index] = sort(tsquared,'descend'); % sort in descending order
listdif(:,4)=sujeto(index(1:10,1),:);

figure('visible','off','units','normalized','outerposition',[0.05 0.05
0.95 0.95])
biplot(coefforth(:,1:3),'scores',score(:,1:3), 'varlabels',HTotal);
xlabel('Componente principal 1')
ylabel('Componente principal 2')
zlabel('Componente principal 3')
%axis([-.18 0.4 -.2 .23 -.22 .4]);
view([30 40]);
set(gca,'FontSize',12,'fontWeight','bold')
set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
fig = gcf;
print(fig, '-dpng', ['biplotPCAT','.png']);
close

list=VectorValsdist(end-9:end,2);
figure('visible','off','units','normalized','outerposition',[0.05 0.05
0.95 0.95])

biplot(VectorVals(list,:),'scores',score(:,1:3),'varlabels',HTotal(list,1));
xlabel('Componente principal 1')
ylabel('Componente principal 2')
zlabel('Componente principal 3')
%axis([-.18 0.4 -.2 .23 -.22 .4]);
view([30 40]);
set(gca,'FontSize',12,'fontWeight','bold')
set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
fig = gcf;
print(fig, '-dpng', ['biplotPCATopT','.png']);
close
topten(:,4) = HTotal(list,1);

list=VectorValsdist(1:10,2);
figure('visible','off','units','normalized','outerposition',[0.05 0.05
0.95 0.95])

biplot(VectorVals(list,:),'scores',score(:,1:3),'varlabels',HTotal(list,1));
xlabel('Componente principal 1')
ylabel('Componente principal 2')
zlabel('Componente principal 3')
%axis([-.002 4 -.007 0.012 -.013 .02]);
view([30 45]);
set(gca,'FontSize',12,'fontWeight','bold')
set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
fig = gcf;
print(fig, '-dpng', ['biplotPCABotT','.png']);
close
lowten(:,4) = HTotal(list,1);

E= evalclusters(score(:,1:3),'kmeans','silhouette','klist',[1:10]);
h=gscatter(score(:,1),score(:,2),E.OptimalY);
```

```
0997 UniqueVal = unique(E.OptimalY);
0998 for k = 1:numel(UniqueVal)
0999       set(h(k), 'ZData', score(E.OptimalY == UniqueVal(k),3));
1000 end
1001 xlabel('Componente principal 1')
1002 ylabel('Componente principal 2')
1003 zlabel('Componente principal 3')
1004 view(3)
1005 view([45 45]);
1006 set(gca,'FontSize',14,'fontWeight','bold')
1007 set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
1008
1009 colorclusters = {'r', 'g', 'b', 'y', 'm', 'c'};
1010 objcenters = {'ko', 'kx', 'k*', 'k+', 'ks', 'kd' , 'kv'};
1011 numcluster=4;
1012
1013 [center,U,objFcn] = fcm(score(:,1:3),numcluster);
1014 maxU = max(U);
1015 clear index
1016 for idx=1:numcluster
1017     index{idx,1} = find(U(idx,:) == maxU);
1018 end
1019 figure
1020 plot3(score(:,1),score(:,2),score(:,3),'o');
1021 hold on
1022 for idx=1:numcluster
1023     line(score(index{idx,1},1), score(index{idx,1},2),
score(index{idx,1},3), 'linestyle',...
1024                            'none','marker',
'*','color',colorclusters{1,idx});
1025 end
1026 xlabel('Componente principal 1')
1027 ylabel('Componente principal 2')
1028 zlabel('Componente principal 3')
1029 view(3)
1030 view([45 45]);
1031 set(gca,'FontSize',14,'fontWeight','bold')
1032 set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
1033
1034
1035
1036
1037
1038 [coeffppca,scoreppca,pcvarppca,muppca] = ppca(Total,3);
1039 E= evalclusters(scoreppca(:,1:3),'kmeans','silhouette','klist',[1:10]);
1040 figure
1041 h=gscatter(scoreppca(:,1),scoreppca(:,2),E.OptimalY);
1042
1043 UniqueVal = unique(E.OptimalY);
1044 for k = 1:numel(UniqueVal)
1045       set(h(k), 'ZData', scoreppca(E.OptimalY == UniqueVal(k),3));
1046 end
1047 xlabel('Componente principal 1')
1048 ylabel('Componente principal 2')
1049 zlabel('Componente principal 3')
1050 view(3)
```

```
1051 view([45 45]);
1052 set(gca,'FontSize',14,'fontWeight','bold')
1053 set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
1054
1055 colorclusters = {'r', 'g', 'b', 'y', 'm', 'c'};
1056 objcenters = {'ko', 'kx', 'k*', 'k+', 'ks', 'kd' , 'kv'};
1057 numcluster=4;
1058
1059 [center,U,objFcn] = fcm(scoreppca(:,1:3),numcluster);
1060 maxU = max(U);
1061 clear index
1062 for idx=1:numcluster
1063     index{idx,1} = find(U(idx,:) == maxU);
1064 end
1065 figure
1066 plot3(scoreppca(:,1),scoreppca(:,2),scoreppca(:,3),'o');
1067 hold on
1068 for idx=1:numcluster
1069     line(scoreppca(index{idx,1},1), scoreppca(index{idx,1},2),
scoreppca(index{idx,1},3), 'linestyle',...
1070                          'none','marker',
'*','color',colorclusters{1,idx});
1071 end
1072 xlabel('Componente principal 1')
1073 ylabel('Componente principal 2')
1074 zlabel('Componente principal 3')
1075 view(3)
1076 view([45 45]);
1077 set(gca,'FontSize',14,'fontWeight','bold')
1078 set(findall(gcf,'type','text'),'FontSize',12,'fontWeight','bold')
```

Referencias Bibliográficas

Bradski, G., *The opencv library*, Doctor Dobbs Journal, Vol. 25, 2000, pp. 120-126

Bush, K., and Antonyshyn, O., *Three-Dimensional Facial Anthropometry Using a Laser Surface Scanner: Validation of the Technique*, Plastic and Reconstructive Surgery, Vol. 98, No. 2, 1996, pp. 226-235

Delmas, P. (2013) *Online Computational Stereo Vision* [Internet], IVS LAB. Disponible desde <http://www.ivs.auckland.ac.nz/quick_stereo/index.php> Acceso 2 de febrero 2015].

Farkas, L. and Deutsch, C., *Anthropometric Determination of Craniofacial Morphology*, American Journal of Medical Genetics, Vol. 65, 1996, pp. 1-4

Frisancho, R., *Anthropometric Standards for the Assesment of Growth and Nutritional Status*, The University of Michigan Press, USA, 1990, pp. 29-30

Galantucci, L. M, Percoco, G., Di Gioia, E., *Photogrammetric 3D Digitization of Human Faces Based on Landmarks*, Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 Vol I, IMECS 2009, March 18 - 20, Hong Kong, 2009

George, R., *Facial Geometry: Graphic Analysis for Forensic Artists*, Charles C. Thomas Publisher, LTD., 2007, pp. 3-17, 38-60

Ghoddousi, H., Edler, R., Haers, P., Wertheim, D. and Greemhill,D., *Comparison of three methods of facial measurement*, International Journal of Oral and Maxillofacial Surgery, Vol. 36, 2007, pp. 250-258

Jolliffe, I., *Principal Component Analysis*, Springer Verlag, Berlin, 1986.

Kintel, M. (2014) *OpenSCAD The Programmers Solid 3D CAD Modeller* [Internet], OpenSCAD. Disponible desde <http://www.openscad.org/about.html> Acceso 2 de febrero 2015].

Menezes, M., Rosati, R., Allievi, C. and Sforza, C., *A Photographic System for the Three-Dimensional Study of Facial Morphology*, Angle Orthodontist, Vol. 79, No. 6, 2009, pp. 1070-1077

Metzler, P., Sun, Y., Zemann, W., Bartella, A., Lehner, M., Obwegeser, J., Kruse-Gujer, A. and Lübbers, H., *Validity of the 3D VECTRA photogrammetric surface imaging system for cranio-maxillofacial anthropometric measurements*, Oral and Maxillofacial Surgery, Vol. 18, 2014, pp. 297-304

Naini, F., *Facial Aesthetics: Concepts and Clinical Diagnosis*, John Wiley and Sons, UK 2011, pp. 127-128

Omnivision. (2006) *OV7720/OV7721 CMOS VGA (640x480) CameraChip sensor* [Internet], Omnivision Technologies, Inc. Disponible desde: <http://www.zhopper.narod.ru/mobile/ov7720_ov7221_full.pdf> [Acceso 2 de febrero 2015]

Ozsoy, U., Demirel, B., Yildirim, F., Tosun, O. and Sarikcioglu, L., *Method Selection in craniofacial measurements: Advantages and disadvantages of 3D digitization method*, Journal of Cranio-Maxillofacial Surgery, Vol. 37, 2009, pp. 285-290

Pilar, L. (2015) Morfometría facial en poblaciones sanas mediante un sistema de estereovisión. Tesis de Maestría, Universidad Nacional Autónoma de México.

Prince, S.J.D., *Computer Vision: Models Learning and Inference*, Cambridge University Press, United Kingdom, 2012

Stalling, D., Westerhoff, M., Hege, H.C., *Amira: A Highly Interactive System for Visual Data Analysis*. The Visualization Handbook (Elsevier), pp. 749–767. 2005

Sforza, C. and Ferrario, V., *Soft-tissue facial anthropometry in three dimensions: from anatomical landmarks to digital morphology in research, clinics and forensic anthropology*, Journal of Anthropological Sciences, Vol. 84, 2006, pp. 97-124

Sifakis, E., Selle, A., Rbinson-Mosher, A. y Fedkiw, R., *Simulating speech with a physics-based facial muscle model*, In M.Cani, editor, In ACM SIGGRAPH/Eurographics Symp. on Computer Animation, Eurographics Association, 2006, pp 261–270.

Wong, J., Oh, A., Ohta, E., Hunt, A., Rogers, G., Mulliken, J. and Deutsch, C., *Validity and Reliability of Craniofacial Anthropometric Measurement of 3D Digital Photogrammetric Images*, Cleft Palate-Craniofacial Journal, Vol. 45, No. 3, 2008, pp. 232-239

Woodward, A., Delmas, P., Chan, Y., Gastelum, A., Gimel'farb, G. and Marquez, J., *An interactive 3D video system for human facial reconstruction and expression modeling*, Journal of Visual Communication and Image Representation, Vol. 23, 2012, pp 1113-1127

Xu, G., Zhang, Z., *Epipolar Geometry in Stereo, Motion and Object Recognition. An Undefined Approach*, Springer Science+Bussines Media, The Netherlands, 1996, pp. 7-69

Zhang, Z., *Flexible Camera Calibration by Viewing a Plane From Unknown Orientations*, The Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Sept. 20-27, 1999. IEEE, Vol. 1, pp. 666-673