Escola Politècnica Superior
d'Edificació de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

INGENIERÍA TÉCNICA EN TOPOGRAFÍA
PROYECTO FINAL DE CARRERA

MODELO DE ORIENTACIÓN RELATIVA BASADA EN LAS ECUACIONES DE COLINEALIDAD

Proyectista: Javier González

Directores: Albert Prades y Felipe Buill

Convocatoria: Junio 2013

RESUMEN

En este trabajo se pretende implementar un programa en C++ que permita la adquisición de las coordenadas imagen de puntos homólogos de un par estereoscópico. Estas coordenadas se corregirán de posibles errores sistemáticos (distorsión de la lente) y posteriormente serán utilizadas en el cálculo de la orientación relativa del par que basaremos el modelo geométrico en la ecuación de colinealidad. Finalmente, se conseguirá ver el modelo estereoscópico tridimensional mediante unas gafas de anáglifos.

ÍNDICE

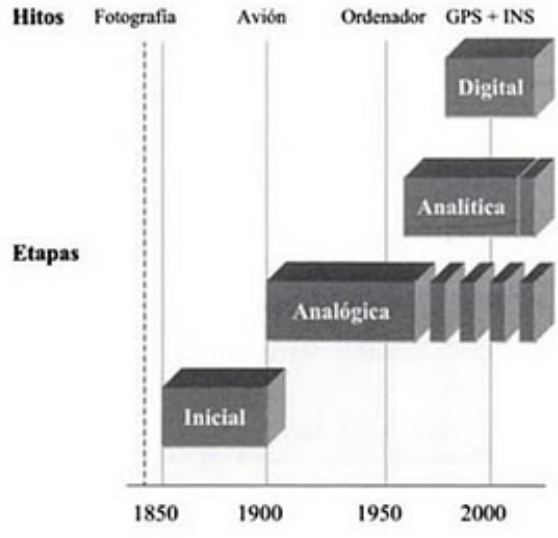
1. INTRODUCCIÓN

La fotogrametría *groso modo* se puede definir como la disciplina encargada de calcular las dimensiones y posiciones de los objetos en el espacio a partir de medidas realizadas sobre fotografías. Al trabajar con la fotogrametría se puede obtener información bidimensional o tridimensional. Se suele trabajar con dos o más fotografías las cuales tienen una zona en común.

La fotogrametría ha ido evolucionado a lo largo de su historia. Se podría establecer diferentes etapas en función de su resolución. Así se puede distinguir entre Fotogrametría Analógica, donde la resolución se realiza mediante analogías mecánicas y ópticas y por el otro lado la Fotogrametría Electrónica donde los sistemas óptico-mecánicos son sustituidos casi en su totalidad por equipos informáticos
Dentro de la fotogrametría electrónica se pueden distinguir entre:

- Fotogrametría Analítica
- Fotogrametría digital

La ***fotogrametría analítica*** aborda la resolución informatizada de los cálculos fotogramétricos, utilizando como información de entrada las medidas realizadas sobre la fotogrametría. Aún mantiene equipos mecánicos de servomotores que manejan los portaplacas y sistemas ópticos que permiten realizar las observaciones fotogramétricas.



Por otra parte tenemos la ***fotogrametría digital*** que se caracteriza por la utilización de información (imágenes) en formato digital. Se produce un cambio en el soporte de la información, ahora está totalmente preparado para el tratamiento informático desde el principio del proceso fotogramétrico. Esto traerá un gran número de ventajas, que van relacionadas principalmente con la automatización de tareas fotogramétricas, disminución del coste de los equipos: no tienen problemas de estabilidad dimensional, no se deterioran… Pero también traen una serie de desventajas como la perdida de precisión y peor calidad de definición de los elementos que aparecen en los fotogramas. Desventaja que van disminuyendo progresivamente conforme aumenta la capacidad de los sistemas informáticos junto a mayor resolución de imágenes.

2. IMAGEN DIGITAL

2.1. DEFINICIÓN

Una imagen digital es una matriz bidimensional de niveles grises con elementos de información mínima, g_{ij} que varían en función de la posición de *i,j* (fila y columna) de la matriz que define la imagen (*Fig. 1*). Cada posición de esta matriz se denomina píxel ('picture element') y tiene un tamaño finito de muestreo $\Delta d_F \cdot \Delta d_C$ dónde normalmente Δd_F = Δd_C. En resumen, en una imagen digital se habla de elementos de imagen o píxeles en vez de puntos de imagen.

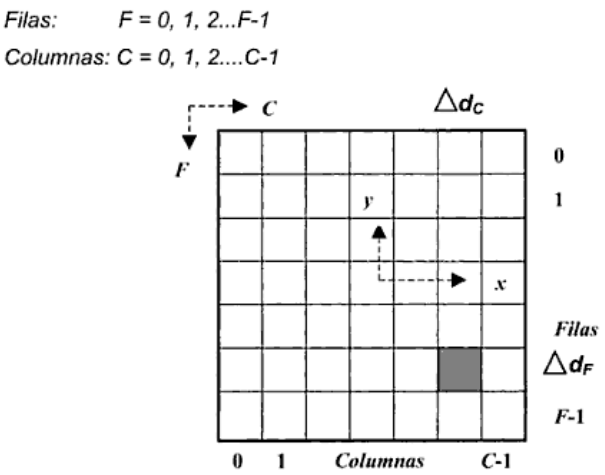


Fig. 1. Imagen digital

Las imágenes digitales se forman a partir de un proceso de muestreo o digitalización. En dicho proceso, una pequeña área sensorial de tamaño mínimo es capaz de registrar i captar de manera directa e indirecta la información electromagnética. (*Fig. 2*)

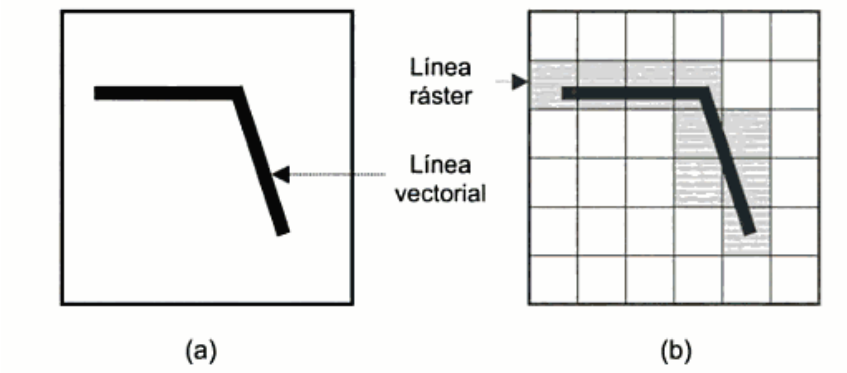


Fig. 2. Concepto de muestreo o digitalización: (a) fotografía analógica (b) imagen digitalizada

En fotogrametría digital se trabaja tradicionalmente con coordenadas imagen. Cuando la captura de la imagen digital se produce en modo indirecto a través de fotogramas, la obtención de las mismas exige unos pasos previos, pero nosotros nos vamos a centrar en fotografías digitalizadas donde las coordenadas fiduciales se sustituyen por las fotocoordenadas de las esquinas.

Las coordenadas digitales no están referidas al centro de la imagen digital. El sentido de los ejes de coordenadas tampoco suele coincidir y se debe corregir (*Fig. 1*). Las coordenadas fiduciales dejan de existir y se reemplazan por las coordenadas de las esquinas de la propia imagen. En los sensores digitales estables, la orientación interna está ligada a la definición de los parámetros de calibración.

2.2. CONCEPTOS RELACIONADOS

En el ámbito de la compresión de imágenes se emplean una serie de conceptos que permiten caracterizar los algoritmos de compresión.

2.2.1. RAZÓN DE COMPRESIÓN

La razón de compresión (RC) es un cociente entre el número de bytes de la imagen original y el número de bytes de la imagen comprimida:

$$RC = \frac{tamaño\ imagen\ original\ (bytes)}{tamaño\ imagen\ comprimida\ (bytes)}$$

Conviene remarcar que este parámetro no indica nada acerca de la fiabilidad de la compresión. Es más, la razón de compresión (RC) depende de una serie de factores como el tamaño del píxel, el nivel radiométrico, el tamaño del grano de la película, la heterogeneidad de valores radiométricos, el ruido, etc.

La razón de compresión está directamente relacionada con diferentes conceptos y criterios como el tiempo de compresión/descompresión y el nivel de pérdida de información.

La razón de compresión máxima en algoritmos de compresión sin pérdida suele oscilar entre 1.5-5, mientras que para algoritmos de compresión con pérdida la razón puede llegar a 10 e incluso a tomar valores superiores a 100.

La razón de compresión depende de la complejidad del algoritmo de compresión/descompresión y de la información contenida en la imagen, que es susceptible de ser modificada, comprobamos que los factores que influyen sobre la razón de compresión son:

- *Tamaño de muestreo*: a menor tamaño de píxel, mayor correlación entre píxeles, y por tanto, la razón de compresión que se alcanza también es mayor. No obstante, la imagen original también aumenta de tamaño, con lo que hay que establecer el tamaño mínimo de muestreo.
- *Niveles radiométricos*: a mayor número de bits por píxel, menos correlación entre píxeles y menos capacidad de compresión. La disminución del número de niveles de grises de la imagen digital depende de las aplicaciones ulteriores.

2.2.2. RESOLUCIÓN

Una imagen digital presenta distintos tipos de resolución según el parámetro de medida: resolución geométrica, resolución radiométrica y resolución espectral.

La **Resolución geométrica** hace referencia al tamaño de la matriz bidimensional (filas x columnas) de la imagen digital. A mayor número de píxeles en la dirección vertical/horizontal, mayor resolución geométrica, mayor definición de imagen y mayor tamaño de almacenamiento.

Ejemplo: *si se digitaliza una imagen a 1200ppp (puntos por pulgada) indicará que 25,4 mm (que corresponden a 1 pulgada) se ha dividido en 1200 partes iguales (tamaño del píxel) por tanto este será de:*

$$\frac{1\ pulgada}{resolucion\ de\ la\ digitalizacion(ppp)} = \frac{25{,}4\ mm}{1200} = 21{,}2\mu$$

Para ver cuánto ocuparía el fichero digitalizado de dicha imagen:

$$\frac{1200\ ppp}{2{,}54\ cm} = 472\ puntos\ por\ cm; 472\ pt\ por\ cm\ x\ 23\ cm = 10856\ pt(píxeles)$$

10856 x 10856 = 117852736 puntos (píxeles)

Tamaño de la imagen en B/N= 118 Mb

Tamaño de la imagen en color = 354 Mb

La **Resolución radiométrica** especifica el numero de niveles de gris que se utilizan por banda y viene definida en función del número de dígitos binarios (bits). La información radiométrica contenida en una imagen de color (tres bandas espectrales: rojo, verde, azul) o multiespectral (tres o más bandas) es múltiplo del número de bandas.

La **resolución espectral** indica el rango de longitudes de onda del espectro electromagnético registrado en la imagen digital. Cuanto mayor sea el número de bandas espectrales, mayor precisión se obtiene en la creación de patrones de respuesta espectral, y más fáciles y seguras serán las tareas de reconocimiento geométrico de las formas.

3. ORIENTACIÓN

3.1. ORIENTACIÓN RELATIVA

La orientación relativa es el proceso que permite determinar en un par de imágenes fotográficas las orientaciones angulares y las posiciones relativas que se dieron en el momento de la toma. La orientación relativa suele ser el primer paso en la orientación de pares de imágenes fotográficas, y finaliza con la formación del modelo (estereoscópico) y con la determinación de coordenadas modelo de puntos homólogos en un par de imágenes.

La orientación relativa se habrá realizado correctamente si se produce la intersección de todos los **rayos homólogos**. El mínimo número de puntos de intersección para lograr la formación del modelo es de cinco, distribuidos según Vön Grüber (Fig. 3)

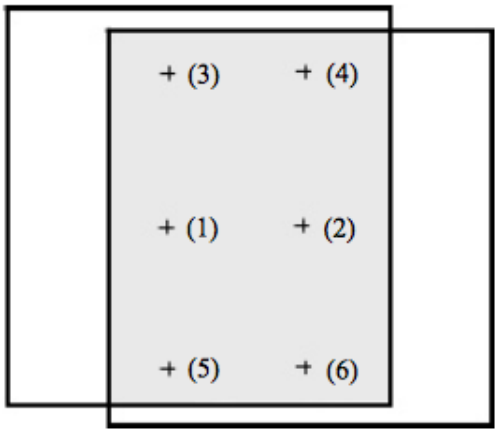
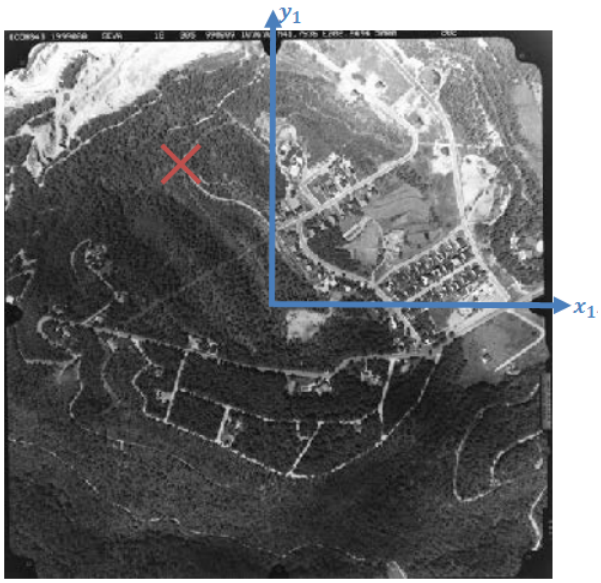


Fig.3 Distribución de Vön Grüber

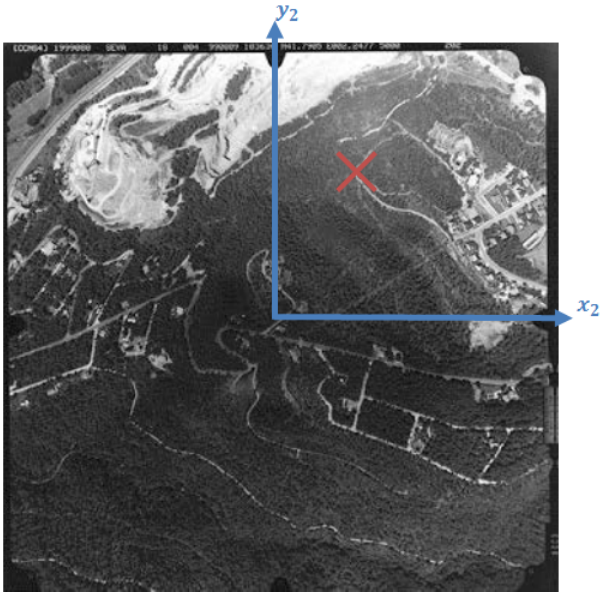
Cuando se tiene dos fotogramas se puede llegar a solucionar de diferentes maneras según los datos de partida que disponemos. Si no tenemos coordenadas terreno de los puntos de apoyo se puede utilizar algún método de orientación relativa:

- Orientación relativa mediante la condición de coplanariedad
- Orientación relativa mediante la **condición de colinealidad**

En este proyecto la aplicación utiliza el proceso de orientación relativa mediante la **condición de colinealidad**.



Fotograma 1



Fotograma2

Con la finalidad de poder comprobar la fiabilidad de los cálculos a realizar, vamos a establecer un sistema de ecuaciones redundante para poder resolverlo con MMCC en un proceso iterativo.

La orientación relativa es un proceso que nos permite determinar en un par de imágenes fotográficas las orientaciones angulares, así como las posiciones relativas que se dieron en el momento de realizar la captura. Por tanto el objetivo es la orientación de dos haces de un modelo relativo, uno respecto del otro, de manera que los rayos homólogos se corten.
Si conseguimos eso, obtendremos un modelo estereoscópico, es decir, podremos conseguir una representación 3D del par de fotografías (de la zona en común) y para obtener eso tenemos que conseguir que un mínimo de 5 puntos homólogos se corten.

3.2. PARALAJE

Para realizar los cálculos tendremos que tener en cuenta diferentes factores como son la paralaje (horizontal y vertical). La paralaje es el desplazamiento aparente en la posición de un objeto al observarlo desde dos puntos de vista distintos. En fotogrametría se distingue entre paralaje horizontal y paralaje vertical, siendo el primero (Px) el que permite encontrar las coordenadas del modelo, y el segundo (Py), el que debe ser eliminado en el proceso de orientación relativa para poder tener visión estereoscópica.

La paralaje estereoscópica o paralaje absoluta, es el cambio en posición de la imagen de un mismo punto en dos fotografías, producido por el cambio de posición de la cámara.

3.3. CONDICIÓN DE COLINEALIDAD

La condición de colinealidad establece que, para cualquier fotografía dada; el centro de proyección, el punto imagen y el punto objeto deben estar en la misma recta. De tal forma que si conseguimos esta condición en las dos perspectivas, queda asegurada la intersección de rayos homólogos.

Asumimos los siguientes puntos para empezar:

- Se asume que la fotografía es una proyección central, es decir, se supone un centro de proyección en el objetivo, a pesar de que la fotografía, en realidad, no es una estricta proyección central, hay haces que atraviesan el objetivo por diferentes partes.
- Se asume que el objetivo está libre de distorsión, aunque sabemos que no es así. Los rayos procedentes del objeto no convergen en el plano del negativo para formar la imagen correcta.
- Se asume que la imagen de un punto objeto es un punto. Al procesar el negativo un punto queda representado por un área.

3.3.1. MÉTODO

Suponiendo que disponemos de las fotocoordenadas de al menos 5 puntos podemos calcular lo siguiente:

- los ángulos de orientación externa independientes $b_y, b_z, \kappa_2, \omega_2, \varphi_2$
- las coordenadas modelo de puntos medidos

$$R = \begin{pmatrix} cos\varphi\ cos\kappa & cos\ \varphi\ sin\kappa & -sin\varphi \\ -cos\omega\ sin\kappa + sin\omega\ sin\varphi\ cos\kappa & cos\omega\ cos\kappa + sin\omega\ sin\varphi\ sin\kappa & sin\omega\ cos\varphi \\ sin\omega\ sin\kappa + cos\omega\ sin\varphi\ cos\kappa & -sin\omega\ cos\kappa + cos\omega\ sin\varphi\ sin\kappa & cos\omega\ cos\varphi \end{pmatrix}$$

Código c++: matriz de giro

```
//rellenamos matriz M(M->R->M giro)

M[0][0]= cos(o)*cos(k);
M[0][1]= cos(o)*sin(k);
M[0][2]=-sin(o);
M[1][0]=-(cos(w)*sin(k))+(sin(w)*sin(o)*cos(k));
M[1][1]= (cos(w)*cos(k))+(sin(w)*sin(o)*sin(k));
M[1][2]= sin(w)*cos(o);
M[2][0]= (sin(w)*sin(k))+(cos(w)*sin(o)*cos(k));
M[2][1]=-(sin(w)*cos(k))+(cos(w)*sin(o)*sin(k));
M[2][2]= cos(w)*cos(o);
```

3.3.2. ECUACIONES DE COLINEALIDAD

$$x = x_0 + f\frac{m_{11}(x_m - x_0) + m_{12}(y_m - y_0) + m_{13}(z_m - z_0)}{m_{31}(x_m - x_0) + m_{32}(y_m - y_0) + m_{33}(z_m - z_0)}$$

$$y = y_0 + f\frac{m_{21}(x_m - x_0) + m_{22}(y_m - y_0) + m_{33}(z_m - z_0)}{m_{31}(x_m - x_0) + m_{32}(y_m - y_0) + m_{33}(z_m - z_0)}$$

x, y son las fotocoordenadas del punto de estudio

$x_0, y_0\ y\ f$ son los parámetros de orientación interna de la cámara

m_{ij} son los coeficientes de la matriz de rotación espacial (R)

x_m, y_m, z_m son las coordenadas modelo del punto de estudio

x_0, y_0, z_0 son las coordenadas modelo del centro de proyección

Para conseguir una expresión más compacta definiremos las siguiente variables (*q*,*s*,r) como:

$$q = m_{31}(x_m - x_0) + m_{32}(y_m - y_0) + m_{33}(z_m - z_0)$$
$$s = m_{21}(x_m - x_0) + m_{22}(y_m - y_0) + m_{23}(z_m - z_0)$$
$$r = m_{11}(x_m - x_0) + m_{12}(y_m - y_0) + m_{13}(z_m - z_0)$$

$$x = x_0 + f\,\frac{r}{q}$$
$$y = y_0 + f\,\frac{s}{q}$$

Suposiciones para nuestro modelo: tenemos una cámara de la cual tenemos sus parámetros de calibración. Por tanto, x_0, y_0 y f son conocidas. Tenemos fotocoordenadas en las dos imágenes de sus puntos característicos

DATOS:

$x_i^1, y_i^1, y_i^2, y_i^2$	$i = 1, \dots, n$	fotocoordenadas de los *n* puntos
x_0, y_0, f		parámetros de calibración de la cámara
$\Delta x_i = \Delta y_i = 0$		coordenadas imagen con errores menospreciables

INCÓGNITAS:

$b_y, b_z, \omega_2, \varphi_2, \kappa_2$	ángulos de orientación y dos bases
$x_m^i, y_m^i, z_m^i \quad = 1, \dots, n$	coordenadas modelo de los *n* puntos

Como disponemos de más ecuaciones que incógnitas es un sistema incompatible. Dado el modelo estocástico de que disponemos, una solución la podemos encontrar mediante un ajuste mínimo cuadrático:

$$\widehat{\boldsymbol{X}} = (\boldsymbol{A}^{\boldsymbol{T}}\,\boldsymbol{P}\boldsymbol{A})^{-1}\,\boldsymbol{A}^{\boldsymbol{T}}\,\boldsymbol{P}\boldsymbol{L}$$

Es un sistema de ecuaciones no lineales por lo que tendremos que pasarlo a un sistema lineal a través de un desarrollo en serie. Posteriormente realizamos las derivadas, y lo expresamos de manera matricial para así obtener la matriz de diseño A.

A es la matriz de diseño (*Fig. 4*).

$$\mathbf{A} = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & \frac{\partial F_x^1}{\partial x_m^1} & \frac{\partial F_x^1}{\partial y_m^1} & \frac{\partial F_x^1}{\partial z_m^1} & \cdots & \frac{\partial F_x^1}{\partial x_m^P} & \frac{\partial F_x^1}{\partial y_m^P} & \frac{\partial F_x^1}{\partial z_m^P} \\
0 & 0 & 0 & 0 & 0 & \frac{\partial F_y^1}{\partial x_m^1} & \frac{\partial F_y^1}{\partial y_m^1} & \frac{\partial F_y^1}{\partial z_m^1} & & \frac{\partial F_y^1}{\partial x_m^P} & \frac{\partial F_y^1}{\partial y_m^P} & \frac{\partial F_y^1}{\partial z_m^P} \\
\frac{\partial F_x^2}{\partial \omega_2} & \frac{\partial F_x^2}{\partial \varphi_2} & \frac{\partial F_x^2}{\partial \kappa_2} & \frac{\partial F_x^2}{\partial b_y} & \frac{\partial F_x^2}{\partial b_z} & \frac{\partial F_x^2}{\partial x_m^1} & \frac{\partial F_x^2}{\partial y_m^1} & \frac{\partial F_x^2}{\partial z_m^1} & & \frac{\partial F_x^2}{\partial x_m^P} & \frac{\partial F_x^2}{\partial y_m^P} & \frac{\partial F_x^2}{\partial z_m^P} \\
\frac{\partial F_y^2}{\partial \omega_2} & \frac{\partial F_y^2}{\partial \varphi_2} & \frac{\partial F_y^2}{\partial \kappa_2} & \frac{\partial F_y^2}{\partial b_y} & \frac{\partial F_y^2}{\partial b_z} & \frac{\partial F_y^2}{\partial x_m^1} & \frac{\partial F_y^2}{\partial y_m^1} & \frac{\partial F_y^2}{\partial z_m^1} & \cdots & \frac{\partial F_y^2}{\partial x_m^P} & \frac{\partial F_y^2}{\partial y_m^P} & \frac{\partial F_y^2}{\partial z_m^P} \\
\vdots & & & \vdots & \vdots & & & \vdots & & \vdots & & \vdots \\
0 & 0 & 0 & 0 & 0 & \frac{\partial F_x^{2P-1}}{\partial x_m^1} & \frac{\partial F_x^{2P-1}}{\partial y_m^1} & \frac{\partial F_x^{2P-1}}{\partial z_m^1} & & \frac{\partial F_x^{2P-1}}{\partial x_m^P} & \frac{\partial F_x^{2P-1}}{\partial y_m^P} & \frac{\partial F_x^{2P-1}}{\partial z_m^P} \\
0 & 0 & 0 & 0 & 0 & \frac{\partial F_y^{2P-1}}{\partial x_m^1} & \frac{\partial F_y^{2P-1}}{\partial y_m^1} & \frac{\partial F_y^{2P-1}}{\partial z_m^1} & & \frac{\partial F_y^{2P-1}}{\partial x_m^P} & \frac{\partial F_y^{2P-1}}{\partial y_m^P} & \frac{\partial F_y^{2P-1}}{\partial z_m^P} \\
\frac{\partial F_x^{2P}}{\partial \omega_2} & \frac{\partial F_x^{2P}}{\partial \varphi_2} & \frac{\partial F_x^{2P}}{\partial \kappa_2} & \frac{\partial F_x^{2P}}{\partial b_y} & \frac{\partial F_x^{2P}}{\partial b_z} & \frac{\partial F_x^{2P}}{\partial x_m^1} & \frac{\partial F_x^{2P}}{\partial y_m^1} & \frac{\partial F_x^{2P}}{\partial z_m^1} & & \frac{\partial F_x^{2P}}{\partial x_m^P} & \frac{\partial F_x^{2P}}{\partial y_m^P} & \frac{\partial F_x^{2P}}{\partial z_m^P} \\
\frac{\partial F_y^{2P}}{\partial \omega_2} & \frac{\partial F_y^{2P}}{\partial \varphi_2} & \frac{\partial F_y^{2P}}{\partial \kappa_2} & \frac{\partial F_y^{2P}}{\partial b_y} & \frac{\partial F_y^{2P}}{\partial b_z} & \frac{\partial F_y^{2P}}{\partial x_m^1} & \frac{\partial F_y^{2P}}{\partial y_m^1} & \frac{\partial F_y^{2P}}{\partial z_m^1} & \cdots & \frac{\partial F_y^{2P}}{\partial x_m^P} & \frac{\partial F_y^{2P}}{\partial y_m^P} & \frac{\partial F_y^{2P}}{\partial z_m^P}
\end{pmatrix}$$

Fig. 4. Aspecto de la matriz A

Código c++ : formación matriz de diseño A

```
//Matriz A

//w
A[n+numpt*2][0]=(-s2*xc2)/q;
A[n+numpt*3][0]= f-(yc2/q2)*s2;
//f
A[n+numpt*2][1]=-(f/q2)*(sin(o)*cos(k)*(X.Valor(3*n+0+5,0)-…
A[n+numpt*3][1]= (f/q2)*(M.Valor(1,2)*(cos(k)*(X.Valor(3*n+0+ …
//k
A[n+numpt*2][2]=(f/q2)*(-M.Valor(0,1)*(X.Valor(3*n+0+5,0)-…
A[n+numpt*3][2]=(f/q2)*(M.Valor(1,0)*(X.Valor(3*n+1+5,0)-…

//by
A[n+numpt*2][3]=-(f*M.Valor(0,1)+M.Valor(2,1)*xc2)/q2;
A[n+numpt*3][3]=-(f*M.Valor(1,1)+M.Valor(2,1)*yc2)/q2;

//bz
A[n+numpt*2][4]=(-f*M.Valor(0,2)-M.Valor(2,2)*xc2)/q2;
A[n+numpt*3][4]=(-f*M.Valor(1,2)-M.Valor(2,2)*yc2)/q2;

//xm y zm para fx1
A[n][5+3*n]=(f*R.Valor(0,0)+R.Valor(0,2)*xc)/q;
A[n][7+3*n]=(f*R.Valor(0,2)+R.Valor(2,2)*xc)/q;

[…]
```

X el vector de parámetros y **L** el vector de términos independientes:

$$\mathbf{X} = \begin{pmatrix} d\omega_2 \\ d\varphi_2 \\ d\kappa_2 \\ db_y \\ db_z \\ dx_m^1 \\ dy_m^1 \\ dz_m^1 \\ \cdots \\ dx_m^p \\ dy_m^p \\ dz_m^p \end{pmatrix} \qquad \mathbf{L} = \begin{pmatrix} F_x^1 \\ F_y^1 \\ F_x^2 \\ F_y^2 \\ \vdots \\ F_x^{2P-1} \\ F_y^{2P-1} \\ F_x^{2P} \\ F_y^{2P} \end{pmatrix}$$

Donde las dimensiones de las matrices Ax=L

A(4x núm. de puntos, 5+3*numero de puntos) **x**(5+ nombre de puntos,1) = **L** (4x núm. de puntos,1)

Código c++ : resolución MMQQ

```
//Matriz L
	U[n][0]= xc - P.Valor(n,0);
	U[n+numpt][0]= yc - P.Valor(n,1);
	U[n+numpt*2][0]=xc2-P.Valor(n,2);
	U[n+numpt*3][0]=yc2-P.Valor(n,3) ;
[…]
//RESOLUCIÓN MMQQ (ITERACIÓN)
	AT=A.Transposada();
	N=AT*A;
	N1=N.InvertirMatriu();
	X1=N1*AT;
	dX=X1*U;

//cambio condiciones iniciales
w=w+dX.Valor(0,0);
o=o+dX.Valor(1,0);
k=k+dX.Valor(2,0);
Y02=Y02+dX.Valor(3,0);
Z02=Z02+dX.Valor(4,0);

	X = X + dX;
[…]
```

3.4. EPIPOLARIZAR

Una vez tenemos la matriz de diseño rellena ya podemos empezar a realizar los cálculos para obtener los valores de (ω,ϕ,κ,by,bz). Estos datos son necesarios para la creación de imágenes epipolares (o imágenes normalizadas).

El modo más sencillo de implementar es la epipolarización de un par de imágenes en el espacio modelo después de una orientación relativa mediante rotaciones, dicha orientación relativa, determina el valor de las rotaciones de las imágenes en el espacio para formar el modelo.

Relación matemática utilizada para epipolarizar:

$$x = f\,\frac{r}{q} = x_0 - f\,\frac{m_{11}(x_m - x_0) + m_{12}(y_m - y_0) + m_{13}(z_m - z_0)}{m_{31}(x_m - x_0) + m_{32}(y_m - y_0) + m_{33}(z_m - z_0)}$$

$$y = f\,\frac{s}{q} = y_0 - f\,\frac{m_{21}(x_m - x_0) + m_{22}(y_m - y_0) + m_{33}(z_m - z_0)}{m_{31}(x_m - x_0) + m_{32}(y_m - y_0) + m_{33}(z_m - z_0)}$$

3.4.1. ROTACIÓN IMÁGENES

Antes de empezar a implementar la relación matemática que definirá nuestras imágenes epipolarizadas, tenemos que calcular las nuevas dimensiones que tendrán las nuevas imágenes, ya que al haber un giro las posiciones de cada píxel variarán, empezando por las esquinas que definen la imagen, por ejemplo, en una imagen de 3008*2000 pixeles podemos sacar las coordenadas de sus esquinas:

Esquina superior izquierda: x=0
y=0

Esquina superior derecha: x=3008
y=0

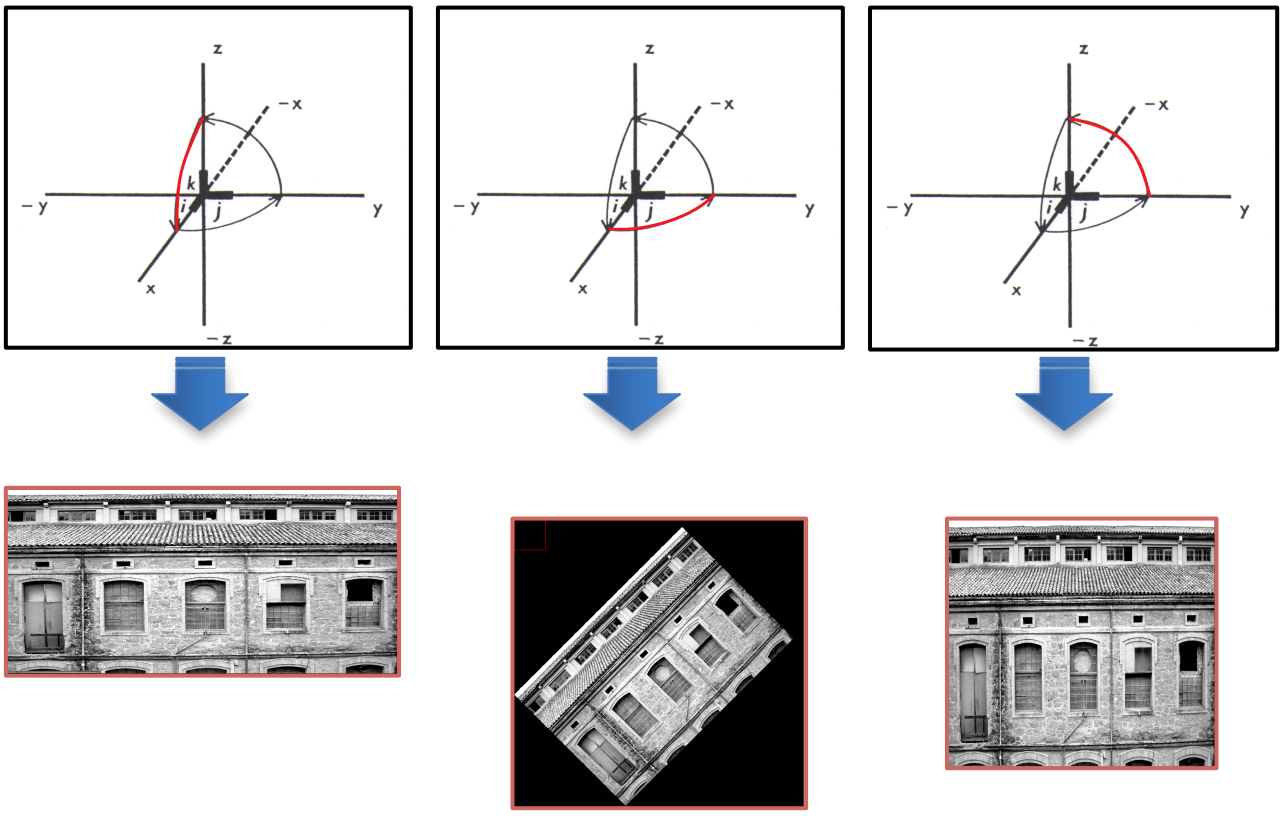
Esquina inferior derecha: x=3008
y=2000

Esquina inferior izquierda: x=0
y=2000

Para hacernos una idea de cómo afectan los giros (3D) en la visión 2D de la imagen se muestra una serie de ejemplos exagerados (45º de giro) para cada eje (x,y,z):

Imagen original Imagen girada

Omega (ω) ***Fi (φ)*** ***Kapa (κ)***



Código c++ : proceso de epipolarización

```
//Dimensiones nuevas
for( int j=0 ; j<2000 ; j++ )
	for( i=0 ; i<3008 ; i++ )
	{	ii = i-3008/2; //centramos giro
		jj = 2000/2 - j;

	x= f*(ii*M1B.Valor(0,0) + jj*M1B.Valor(0,1) -
f*M1B.Valor(0,2))/(ii*M1B.Valor(2,0)+jj*M1B.Valor(2,1) - f*M1B.Valor(2,2)); //x=f*(r/q)
	y= f*(ii*M1B.Valor(1,0) + jj*M1B.Valor(1,1)-
f*M1B.Valor(1,2))/(ii*M1B.Valor(2,0)+jj*M1B.Valor(2,1) - f*M1B.Valor(2,2)); //y=f*(s/q)
		if(xmin>x) xmin= x;
		if(xmax<x) xmax= x;
		if(ymin>y) ymin= y;
		if(ymax<y) ymax= y;
[…]
//ECUACIONES DE GIRO
for( jj=0 ; jj<altura ; jj++ )
	for( ii=0 ; ii<amplada ; ii++ )
	{	x= f*((M1B.Valor(0,0)*(ii+xmin)+M1B.Valor(0,1)*(jj+ymin)+M1B.Valor(0,2)*(-
f))/(M1B.Valor(2,0)*(ii+xmin)+M1B.Valor(2,1)*(jj+ymin)+M1B.Valor(2,2)*(-f)));
		y= f*((M1B.Valor(1,0)*(ii+xmin)+M1B.Valor(1,1)*(jj+ymin)+M1B.Valor(1,2)*(-
f))/(M1B.Valor(2,0)*(ii+xmin)+M1B.Valor(2,1)*(jj+ymin)+M1B.Valor(2,2)*(-f)));
[…]
```

4. SISTEMAS DE VISIÓN

Para la realización de la visión estereoscópica se necesita la inclusión de sistemas que permitan formar el modelo para su tratamiento. Las diferentes maneras de resolverlo son:

- Realizando una división espacial de la pantalla en dos partes, un lado para cada imagen.
- Mediante la separación temporal de las imágenes, mostrando primero la imagen izquierda y luego la derecha.
- Polarizando las dos imágenes y utilizando el principio de anáglifos.

Técnicas de visualización estereoscópica pasiva

Existen varias técnicas de visualización este, sin embargo las más representativas son [2]: la técnica de anáglifos y la polarización.

En este proyecto final de carrera utilizaré el sistema de gafas anáglifos pasivos.

4.1 SISTEMA DE ANÁGLIFOS

¿Qué es un anáglifo? ¿Cómo funciona?

Las imágenes de anáglifos o anáglifos simplemente, son imágenes capaces de provocar un efecto tridimensional impreso sobre dos dimensiones.

Es un sistema sencillo, las gafas están pensadas para que cada ojo reciba las imágenes (de las dos superpuestas) correspondientes, de manera que cada ojo recibe una imagen diferente. El cerebro unirá las dos imágenes formando una sola imagen tridimensional. (Fig. 5)

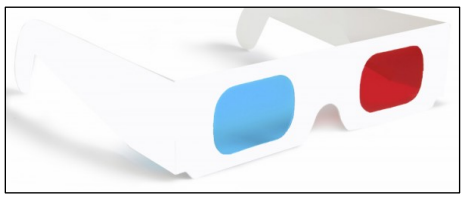


Fig.. 5 Gafas de anáglifos

En monitores convencionales una de las técnicas más utilizadas son los anáglifos. Las gafas que se necesitan son muy baratas y cualquier persona puede disponer de ellas. Dichas gafas utilizan filtros de colores para separar las dos imágenes, así, cuando se observa a través de un filtro rojo, los colores verde o azul se ven como negro. Si se utiliza un filtro verde, azul o cian, el rojo parece negro. A partir de este principio, se pueden mezclar dos imágenes y utilizar lentes con filtros de color para separarlas y observar el efecto estereoscópico. Para imágenes proyectadas y para video se usa un

filtro verde, porque es más brillante; con estos filtros, la imagen aparece en blanco y negro, mientras que otra variante utiliza un filtro rojo y otro cian. Las imágenes o videos producidos en anáglifo se pueden proyectar sin necesidad de equipo especial.

En resumen, un sistema de ***Gafas de anáglifos*** está compuesto por imágenes epipolares con un filtro rojo (para la derecha) y azul (para la izquierda), con la visión con gafas de anáglifos que separan ambas imágenes.(Fig. 6)

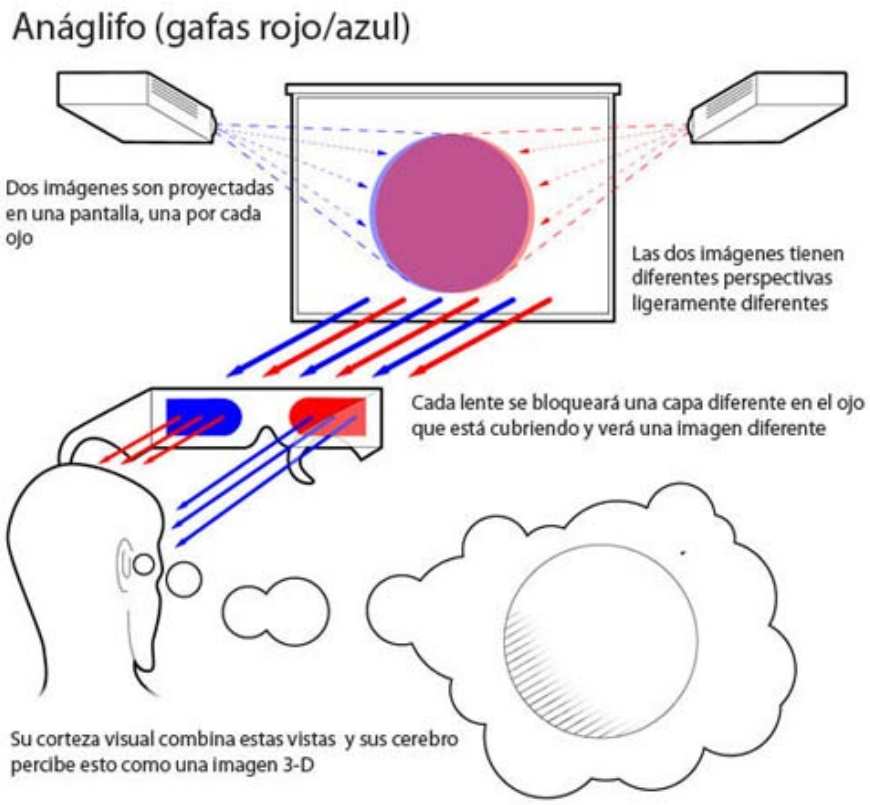


Fig. 6. Grafico de cómo funcionan gafas anaglifas.

Como se vería una de las dos imágenes por separado y como se verá la pantalla o proyección de las imágenes sin las gafas. (Fig. 7 y 8)

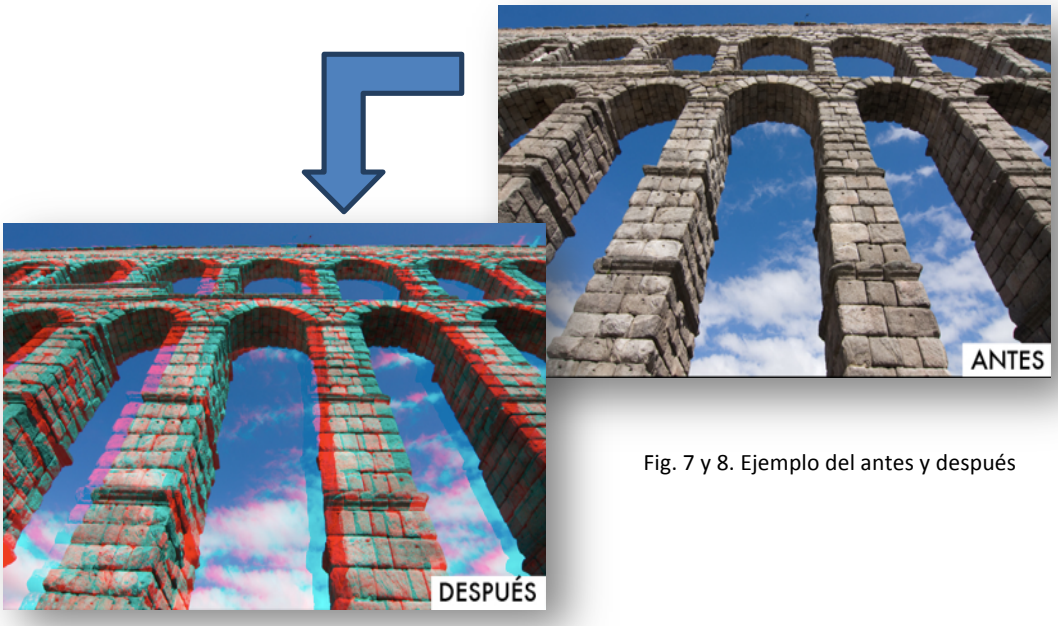


Fig. 7 y 8. Ejemplo del antes y después

5. APLICACIÓN

5.1. OBJETIVO DE LA APLICACIÓN

El objetivo de esta aplicación es la de implementar un programa que soluciones errores sistemáticos y posteriormente realice el cálculo de orientación relativa del par de fotogramas en que se basará el modelo geométrico de las ecuaciones de colinealidad con el objetivo final de mostrar un modelo estereoscópico.

Esta aplicación fue realizada en lenguaje C++ en el entorno de programación Microsoft Visual C++ 2010.

5.2. CÓMO FUNCIONA

5.2.1. INTERFAZ DE LA APLICACIÓN

La interfaz de la aplicación una vez la ejecutamos es una ventana donde veremos un pequeño menú en la parte superior izquierda.

Si entramos en este menú podemos ver diferentes opciones básicas para un programa que trata imágenes como son las opciones de abrir o cerrar imagen, así como un listado de los últimos archivos abiertos (Fig. 9). Una vez carguemos alguna imagen, este menú se nos ampliara con más opciones. (Fig. 10)

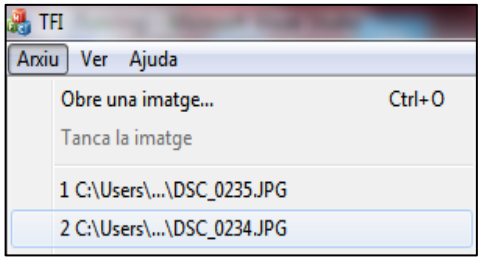


Fig. 9. Menú de inicio.

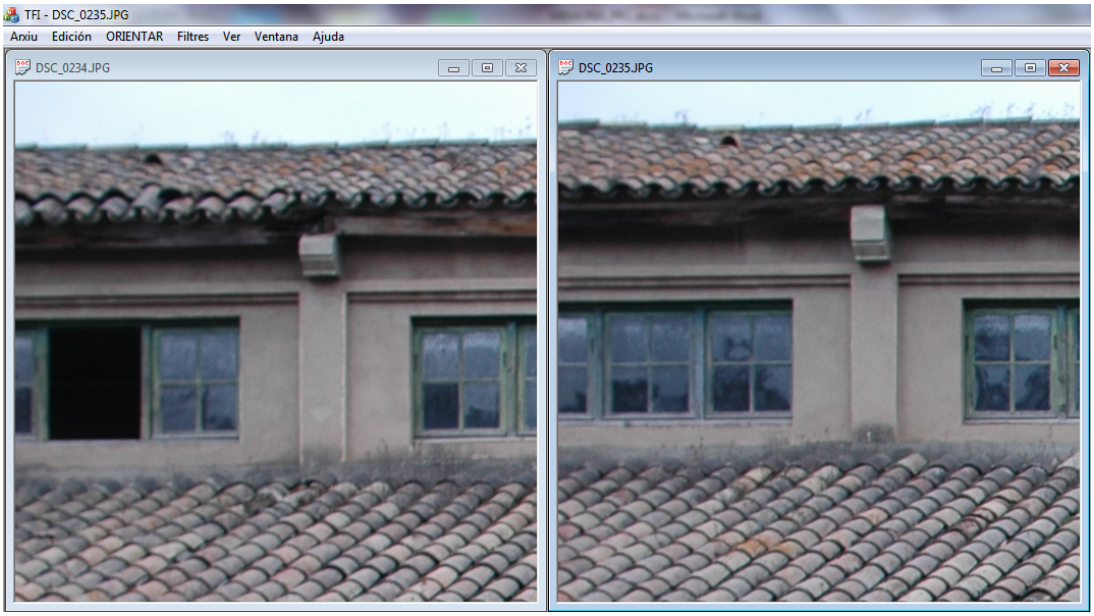


Fig. 10. Ejemplo de dos imágenes cargadas con las nuevas opciones en la barra de herramientas.

5.2.2. ENTRADA DA DATOS

Una vez he cargado las dos imágenes correlativas (a las que me voy a referir a partir de ahora como imagen izquierda e imagen derecha) ya podemos empezar a introducir datos (seleccionar los puntos homólogos) para poder empezar la orientación.

Para ello voy al nuevo menú contextual "**ORIENTAR**" (Fig. 11) en el cual puedo hacer varias cosas. Para facilitar y simplificar los máximo posible el proceso, tanto para el usuario como para mi, a la hora de crear el programa, he creído oportuno dar las siguientes opciones:

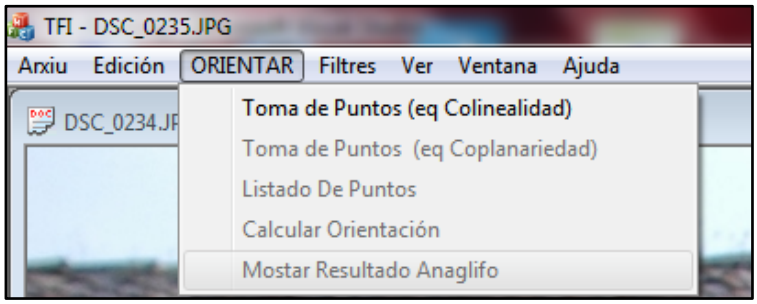


Fig. 11. Opciones dentro del submenú ORIENTAR

- ***Toma de puntos (ec. de Colinealidad)***: activando esta pestaña se me abrirá una nueva ventada, de un tamaño pequeño, para molestar lo mínimo a la toma de los puntos homólogos necesarios y para poder realizar después la orientación a partir de las ecuaciones de colinealidad.

  Esa nueva ventana, que he diseñado para poder tomar los puntos, está formada por unas casillas que muestran la posición del punto que selecciono con el ratón en la imagen, diferenciando los puntos que clico sobre la imagen de la izquierda y sobre la imagen de la derecha (*zona de color verde*) (Fig. 12).

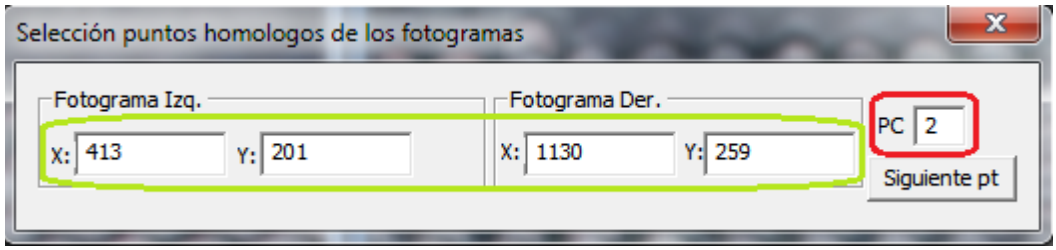


Fig. 12. Ventana dónde se marcan los puntos seleccionados

Además le he añadido un contador de par de puntos (casilla PC, *color rojo*) que aumenta cada vez que clique sobre el botón de "**SIGUIENTE PT**" para saber cuántos puntos llevo y decidir si son suficientes, cosa que dependerá de las imágenes.

Una vez he acabado puedo cerrar esta ventana, y de manera automática, se generara un archivo en formato ".txt" donde estarán los datos de los pares de puntos. Este archivo se creara cada vez que cerremos la ventana y cada archivo txt seguirá la misma estructura, cuatro columnas que contiene las coordenadas pixel de los puntos.

EJEMPLO:

Nombre del archivo: *"puntos homólogos.txt"*

```
Coord.X1.0  Coord.Y1.0  Coord.X2.0  Coord.Y2.0
Coord.X1.1  Coord.Y1.1  Coord.X2.1  Coord.Y2.1
Coord.X1.2  Coord.Y1.2  Coord.X2.2  Coord.Y2.2
```

Las coordenadas de estos puntos se guardaran transformadas para poder realizar los cálculos a posteriori.

Código c++ : formación archivo con puntos homólogos

```
Pt=fopen("puntos homologos.txt","w");
    Fprintf(pt,"%d\n",m_nContadorDePuntos);
    For (int i=0;i<m_nContadorDePuntos;i++)
    {
    m_PnCoordX1[1]=((double)m_PnCoordX1[i]-1504.0)*7.8/1000.0;   FOTO IZQ
    m_PnCoordY1[1]=(1000.0-(double)m_PnCoordY1[1])*7.8/1000.0;

     m_PnCoordX2[i]=((double)m_PnCoordX2[i]-1504.0)*7.8/1000.0;  FOTO DER
    m_PnCoordY2[i]=(1000.0-(double)m_PnCoordY2[i])*7.8/1000.0;

    fprintf(pt," %9.6lf  %9.6lf  %9.6lf  %9.6lf",((double)m_PnCoordX1[i]),
    ((double)m_PnCoordY1[i]), ((double)m_PnCoordX2[i]),
    ((double)m_PnCoordY2[i]));
    }
```

- ***Calcular orientación***: una vez tomo los puntos, y cierro la ventana, ya se genera automáticamente el único archivo que me hará falta para este paso. Con estos valores ya puedo realizar los cálculos para obtener nuevamente un ".txt" con los resultados necesarios como los ángulos de orientación ω, φ, κ.

  Una vez tengo este ".txt" y después de que el programa realice los cálculos de manera interna me aparecerá un mensaje confirmado la finalización de los cálculos.

- ***Mostrar Resultado Anáglifo***: con esta opción, una vez he realizado la toma de puntos homólogos, y posteriormente los cálculos de la orientación, ya puedo alcanzar el objetivo de ver el par de imágenes en 3D con las gafas.

Código c++ : creación anáglifo

```
///// instrucciones de reserva de memoria //////////////
       L= (unsigned char*)new unsigned char[3*ampladaL*alturaL];
////////////////////////////////////////////////////
///// instrucciones de reserva de memoria //////////////
       R= (unsigned char*)new unsigned char[3*ampladaR*alturaL];
////////////////////////////////////////////////////
///// instrucciones de reserva de memoria //////////////
       RES_ANAGLIFO= (unsigned char*)new unsigned char[3*amplada*altura];
////////////////////////////////////////////////////

for( j=0 ; j<3*ampladaL*alturaL ; j++ ) RES_ANAGLIFO[j]=0;
       if((alias1=fopen("Foto_girada.raw", "rb"))==NULL)//imagen izq
       {      puts("Archivo no encontrado");
              system("pause");
              exit(0);
       }
       fread( L , sizeof(unsigned char) , 3*ampladaL*alturaL, alias1 );
       fclose(alias1);

if((alias3=fopen("Foto_girada2.raw", "rb"))==NULL) //imagen der
       {      puts("Archivo no encontrado");
              system("pause");
              exit(0);
       }
       fread( R , sizeof(unsigned char) , 3*ampladaR*alturaR , alias3 );
       fclose(alias3);

for( j=0 ; j<ampladaL ; j++ )
       for( i=0 ; i<alturaL ; i++ )
              RES_ANAGLIFO[amplada*(j+25)+i + 0*amplada*altura]= L[ampladaL*j+i];

for( j=0 ; j<alturaR ; j++ )
       for( i=0 ; i<ampladaR ; i++ )
              RES_ANAGLIFO[amplada*(j+10)+(i+900) + 1*amplada*altura ]=
RES_ANAGLIFO[amplada*(j+10)+ (i+900) + 2*amplada*altura ]= R[ampladaR*j+i];

if((alias4=fopen("RES_ANAGLIFO.raw", "wb"))==NULL)
       {      puts("Archivo no encontrado");
              exit(0);
       }
       fwrite( RES_ANAGLIFO , sizeof(unsigned char),3*amplada*altura ,alias4 );
       fclose(alias4);

////////////////////////GENERAR .HDR//////////////////////
FILE *pt;
       pt=fopen("RES_ANAGLIFO.hdr","w");
       {
       fprintf(pt,"ENVI \n");
       fprintf(pt,"description = { \n");
[…]
       }
       delete [] L;
       delete [] R;
       delete [] RES_ANAGLIFO ;
[…]
```

Ahora voy a mostrar el resultado final del proceso en la siguiente imagen abierta por medio del visor de ENVI (Fig. 13).

Fig. 12. Ventana del visor de ENVI

6. COMPROBACIONES DURANTE EL PROCESO

A lo largo del proceso de programación he ido realizando el programa por partes, para saber los resultados obtenidos paso a paso. De esta manera he podido mirar si los datos tenían sentido, eran correctos y podía ir al siguiente paso con seguridad.

Para poder comprobar los resultados creé un Excel revisado con los cálculos y que he utilizado como soporte para comprobar los resultados.

Aprovechando que un compañero está realizando un PFC similar en que realiza una orientación relativa aplicando las ecuaciones de coplanariedad, nos hemos tenido en cuenta a lo largo del proceso ya que a través de las orientaciones llegaríamos a un mismo punto final, en este caso la creación de imágenes de anáglifos de pares fotográficos.

Por su parte, él también realizó en una hoja excel con el que guiarse, y que junto con el mío nos sirvió de ayuda para ver donde se producían errores en nuestros programas, ya que le resultado tenía que ser el mismo pero el camino a seguir no.

Lógicamente para que esto tenga sentido el par de imágenes utilizadas durante todo el proceso han sido las mismas en ambos proyectos, sino no podríamos ni corroborar nuestros resultados ni demostrar que ya se por un sistema u otro el resultado final es el mismo.

6.1 ITERACIONES

Resultados obtenidos en la primera y la última iteración de lo que buscamos $(b_y, b_z, \kappa_2, \omega_2, \varphi_2)$ y de los puntos:

RESULTADO MMQQ ITER. 1

	dX	X
dw	-0,07140	-0,07140
df	0,06823	0,06823
dk	0,00435	0,00435
by	2,17723	2,17723
bz	-0,35702	-0,35702
dxm1	-0,0421	0,839
dym1	-0,2890	5,811
dzm1	1,4318	-28,568
dxm2	-0,2069	4,270
dym2	-0,2817	5,826
dzm2	1,3864	-28,614
dxm3	-0,1821	0,379
dym3	-0,5815	1,181
dzm3	9,7302	-20,270
dxm4	-0,4833	0,968
dym4	1,4237	-2,812
dzm4	9,9934	-20,007
dxm5	-0,6398	1,248
dym5	2,3645	-4,632
dzm5	10,1679	-19,832
dxm6	-1,9193	4,212
dym6	1,2697	-2,786
dzm6	9,3917	-20,608
dxm7	-2,0281	4,188
dym7	-0,0126	0,026
dzm7	9,7873	-20,213
dxm8	-0,2600	5,520
dym8	-0,2756	5,816
dzm8	1,3496	-28,650

```
Resultado matrices:
w  =     -0.071400505
o  =      0.068231458
k  =      0.004356118
by =      2.177235653
bz =     -0.357028959
Matriz dX y X:

    -0.071400505  =  -0.07140
     0.068231458  =   0.06823
     0.004356118  =   0.00436
     2.177235653  =   2.17724
    -0.357028959  =  -0.35703
    -0.042067244  =   0.83933
    -0.288984363  =   5.81062
     1.431832670  =  -28.56817
    -0.206906039  =   4.27029
    -0.281738290  =   5.82566
     1.386398013  =  -28.61360
    -0.182149078  =   0.37945
    -0.581455816  =   1.18134
     9.730185786  =  -20.26981
    -0.483281142  =   0.96752
     1.423660185  =  -2.81174
     9.993406586  =  -20.00659
    -0.639766494  =   1.24783
     2.364454295  =  -4.63215
    10.167935378  =  -19.83206
    -1.919277822  =   4.21152
     1.269728736  =  -2.78627
     9.391651114  =  -20.60835
    -2.028125547  =   4.18847
    -0.012611418  =   0.02639
     9.787305991  =  -20.21269
    -0.260018192  =   5.51978
    -0.275594143  =   5.81621
     1.349622091  =  -28.65038
    -0.042067244  =   0.83933
    -0.288984363  =   5.81062
     1.431832670  =  -28.56817
    -0.206906039  =   4.27029
    -0.281738290  =   5.82566
     1.386398013  =  -28.61360
```

RESULTADO ÚLTIMA ITER MMQQ REALIZADA Y CONTADA

	dX	X
dw	2,22586E-08	-0,016805573
df	3,3123E-08	0,07905315
dk	-2,1006E-09	0,006878534
by	-5,94717E-07	0,421677651
bz	-5,13229E-07	-0,219600
dxm1	-0,00000008	0,801
dym1	-0,00000049	5,544
dzm1	0,00000251	-27,264
dxm2	-0,00000039	4,101
dym2	-0,00000052	5,595
dzm2	0,00000261	-27,481
dxm3	0,00000000	0,408
dym3	-0,00000013	1,278
dzm3	0,00000168	-21,791
dxm4	-0,00000009	1,057
dym4	0,00000019	-3,083
dzm4	0,00000137	-21,853
dxm5	-0,00000009	1,374
dym5	0,00000038	-5,094
dzm5	0,00000154	-21,839
dxm6	-0,00000034	4,548
dym6	0,00000024	-3,010
dzm6	0,00000175	-22,254
dxm7	-0,00000033	4,556
dym7	-0,00000003	0,031
dzm7	0,00000148	-21,987
dxm8	-0,00000052	5,312
dym8	-0,00000056	5,597
dzm8	0,00000277	-27,571
Núm. Iteraciones		**6**

```
Matriz X:

    -0.016805573
     0.079053150
     0.006878534
     0.421677651
    -0.219599671
     0.800994774
     5.544248266
   -27.264480490
     4.101320765
     5.594855171
   -27.481430713
     0.408038726
     1.278084276
   -21.791096201
     1.056709794
    -3.083056309
   -21.853222480
     1.374174371
    -5.094212672
   -21.839298640
     4.548000032
    -3.010378611
   -22.254447734
     4.556133045
     0.030980204
   -21.987491895
     5.311824378
     5.596719467
   -27.570583743
     0.800994774
     5.544248266
   -27.264480490
     4.101320765
     5.594855171
   -27.481430713
Numero de iteraciones realizadas 6
 FINAL
```

6.2 EPIPOLARIZACIÓN IMÁGENES

Datos de entrada junto a los puntos

BXF	bx	8,8244000	WF	Ω =	0,0000000	rad
BYF	by	0,4216777	OF	Φ =	0,0248520	rad
BZF	bz	-0,2195997	KF	κ =	0,0477491	rad
R	r =	8,8344693				

```
BXF      8.824000000
BYF      0.421677651
BZF     -0.219599671
R        8.834069733
WF       0.000000000
OF       0.024853149
KF       0.047751258
```

Matriz de giro base (MBB) y matrices de giro que aplico a cada imagen (M1B y M2B)

MBB		
0,99855	0,04772	-0,02485
-0,04773	0,99886	0,00000
0,02482	0,00119	0,99969

M1B		
0,99855	0,04772	-0,02485
-0,04773	0,99886	0,00000
0,02482	0,00119	0,99969

M2B		
0,99770	0,03993	0,05479
-0,04073	0,99908	0,01356
-0,05420	-0,01576	0,99841

```
Matriz MB(GIR):

  0.9985517   0.0477184  -0.0248506
 -0.0477331   0.9988601   0.0000000
  0.0248223   0.0011862   0.9996912

Matriz M1B(GIR IMG 1):

  0.9985517   0.0477184  -0.0248506
 -0.0477331   0.9988601   0.0000000
  0.0248223   0.0011862   0.9996912

Matriz M2B(GIR IMG 2):

  0.9977582   0.0394665   0.0540455
 -0.0404471   0.9990341   0.0171717
 -0.0533156  -0.0193192   0.9983908
```

7. CONCLUSIONES

1) Las primeras semanas de realización del proyecto, se invirtieron en el aprendizaje del lenguaje C++ básico, lo que me proporcionó los conocimientos necesarios para poder empezar a programar mínimamente.

2) El trabajo desarrollado permitirá:
   a) Implementar un programa que soluciones errores sistemáticos y posteriormente realice el cálculo de orientación relativa del par de fotogramas en que se basara el modelo geométrico de las ecuaciones de colinealidad con el objetivo final de mostrar un modelo estereoscópico.

   b) Aplicar un giro en el espacio a una de las imágenes que forman el par estereoscópico (proceso automatizado).

   c) Se epipolarizan las imágenes obtenidas, es decir, que una línea recta pase todo a lo largo de las x en cada imagen, y a pesar de ser relativamente distintas, pase exactamente por los mismos puntos de referencia de la zona de solape de ambas.

   d) Generar una imagen tridimensional final en un solo fotograma con las imágenes modificadas.

3) El método para crear la imagen en 3D, la separación espectral anaglífica, es la más simple pero muy funcional de cara a los resultados visuales y coste económico.

4) Al realizar el proyecto a la par con un compañero que realiza un PFC muy parecido hemos podido demostrar que se obtiene le mismo resultado ya sea utilizando las ecuaciones de colinealidad o las ecuaciones de coplanariedad.

8. BIBLIOGRAFÍA

Ceballos Sierra, Fco Javier (1999), Visual C++.Aplicaciones para Win32 (2ª ed)
Editor: RA-MA 1999

Ceballos, Javier (2007), Programación orientada a objetos.
Editor: RA-MA

Lerma, José Luis (2002), Fotogrametría moderna: analítica y digital
Editor: UPV

9. AGRADECIMIENTOS

El proyecto no hubiera podido llevarse a cabo sin la ayuda de mi tutor, Albert Prades, que tuvo la paciencia de atender a todas mis dudas, que fueron muchas al partir sin ningún tipo de conocimiento sobre el lenguaje de programación, durante todo el proceso de programación.

Agradezco también, a mi familia y amigos, que en todo momento me apoyaron y de esa manera me ayudaron a avanzar cada día, poco a poco, hasta acabar.

Por último, pero no menos importante, los compañeros de laboratorio, con los que he podido compartir buenos y malos momentos además de apoyarnos y ayudarnos siempre que hemos podido.