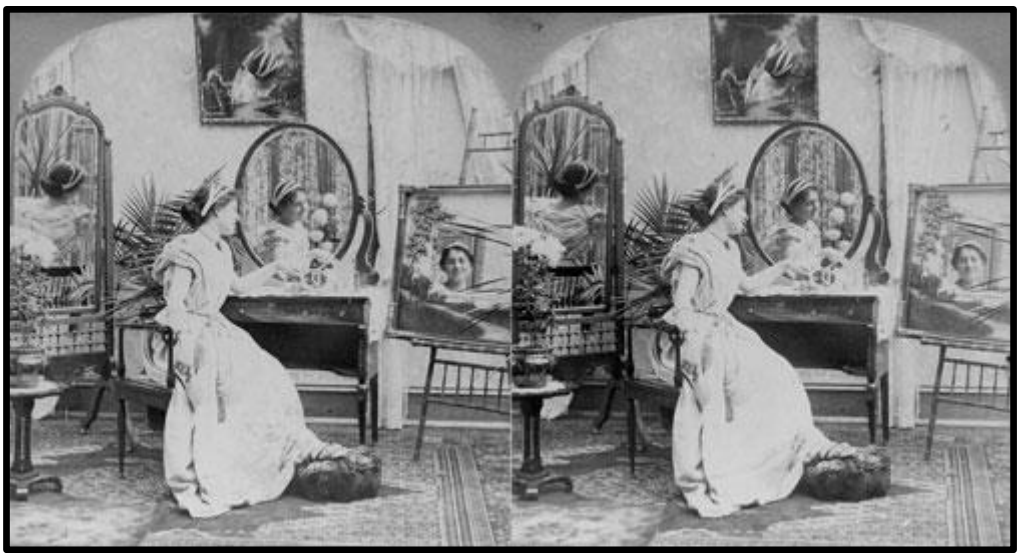
Calibración de un sistema estereoscópico para medida 3D de espacios objeto



Por Pablo Rodríguez Sánchez

bajo la dirección de Juan Antonio Quiroga Mellado

[UCM 2011/2012]

Índice

1 Objetivo

El objetivo del presente trabajo académicamente dirigido es, como indica su título, la calibración de un sistema estereoscópico y su posterior utilización para la reconstrucción tridimensional de una escena.

La persecución de dicho objetivo se ha llevado a cabo en varias etapas, a lo largo de las cuales el autor del presente trabajo ha logrado:

· Comprender los fundamentos de la visión binocular.
· Comprender los rudimentos de la geometría proyectiva y la geometría epipolar.
· Comprender el modelo pinhole para una y dos cámaras.
· Aprender a usar el Toolbox de adquisición de imágenes de Matlab.
· Aprender a usar el Camera Calibration Toolbox de Jean-Yves Bouguet.
· Aprender a usar el software LIBELAS de Andreas Geiger.
· Aprender a integrar para una sola aplicación los tres softwares anteriores.
· Mejorar sus habilidades de programación.
· Introducirse en aspectos técnicos novedosos para él, más cercanos a la ingeniería que la ciencia.

Como suele suceder, el destino ha resultado ser menos provechoso que el camino para llegar hasta él.

2 Modelo geométrico de formación de imágenes

2.1 Modelo básico de cámara oscura

A pesar de que utilizaremos cámaras CCD, nos bastará con un modelo de cámara pinhole o cámara oscura para nuestros propósitos. El siguiente esquema basta para explicar el funcionamiento de una cámara oscura clásica:

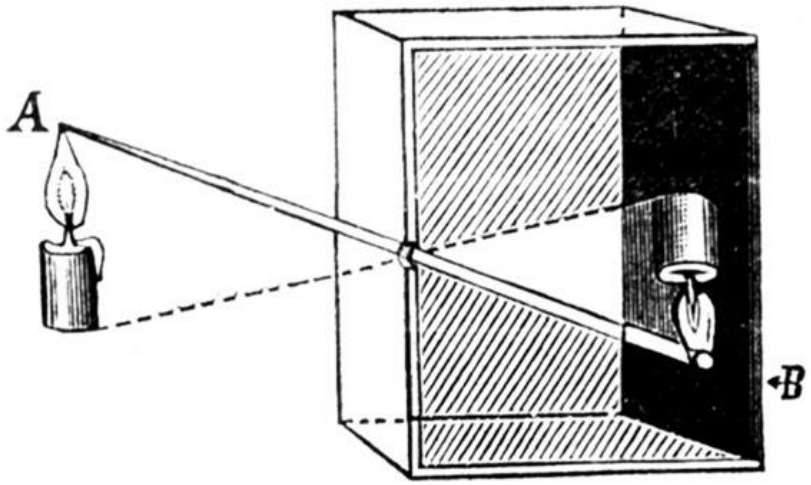


figura 1. Esquema de una cámara oscura

Las cámaras CCD proyectan, al igual que la cámara oscura, una imagen invertida de arriba a abajo sobre su sensor, que posteriormente es corregida por el sistema informático y presentada “del derecho”. Por éste motivo, usaremos un modelo ligerísimamente modificado, que incluye tanto la proyección como la generación de una imagen “derecha”. Dicho modelo se presenta en la figura 2.

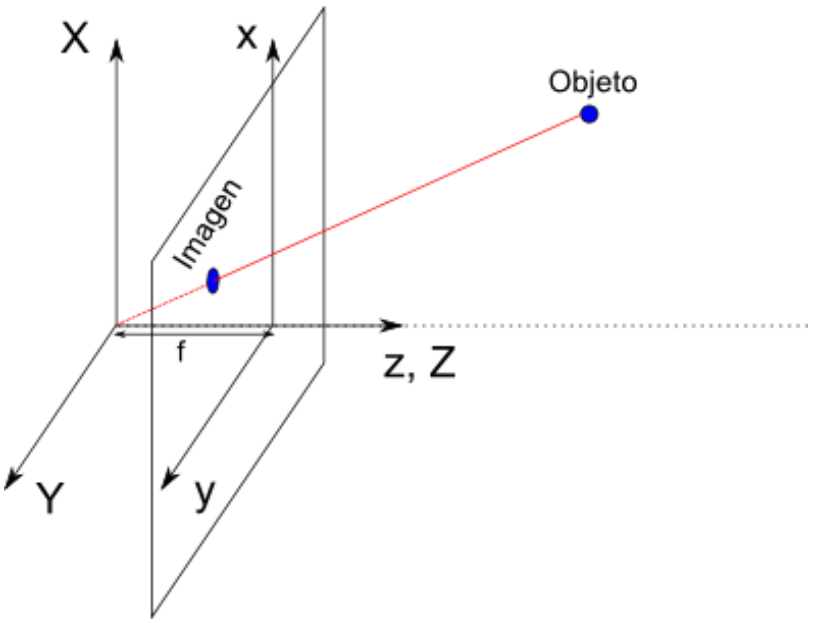


figura 2. Esquema del modelo

Mediante el uso de triángulos semejantes es inmediato demostrar que el punto (X,Y,Z) del espacio se proyecta sobre el punto (x,y,0) del plano de proyección con:

$$\begin{bmatrix} x(X,Y,Z) \\ y(X,Y,Z) \end{bmatrix} = \frac{f}{Z}\begin{bmatrix} X \\ Y \end{bmatrix} \qquad (1)$$

Podemos comprobar fácilmente que ésta relación no es lineal ni invertible. Basta observar que dos o más objetos situados sobre el mismo rayo (por ejemplo, uno en (X_0, Y_0, Z_0) y otro en $(\mu X_0, \mu Y_0, \mu Z_0)$) dan lugar al mismo punto imagen.

Es precisamente el hecho de que la transformación (1) no sea invertible el que exige introducir algún dato extra para poder realizar una reconstrucción tridimensional del espacio imagen. Éste "dato extra" será una segunda imagen de la misma escena tomada por otra cámara.

Resulta útil escribir ésta ecuación utilizando coordenadas homogéneas, muy habituales en geometría proyectiva, pues adoptan la forma de una aplicación lineal:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad (2)$$

Naturalmente, el fenómeno de la proyección sigue siendo no lineal, pero hemos logrado "maquillar" la no linealidad, que reaparecerá de forma explícita al momento de regresar a las coordenadas cartesianas.

2.2 Modelo realista de cámara oscura

Como suele suceder, las cámaras auténticas, especialmente si no son cámaras oscuras, se alejan en mayor o menor medida del modelo sencillo que hemos propuesto en el apartado anterior.

La cámara puede modelarse como un proceso que nos lleva de las coordenadas de un punto del espacio medido desde cierto sistema de referencia al que llamaremos sistema del mundo[1] a las coordenadas de su imagen sobre la pantalla CCD, que llamaremos coordenadas píxel, y denotaremos con el subíndice p.

El proceso completo se compone, en el modelo que vamos a utilizar, de cuatro etapas. Damos a continuación una breve descripción de las mismas:

1. Paso del sistema del mundo (w) al centrado en la cámara (c) mediante una rototraslación.
2. Proyección ideal que proyecta coordenadas c en coordenadas n, sobre el plano imagen.
3. Transformación de los puntos de la proyección ideal (n) en los puntos reales (d), teniendo en cuenta las distorsiones radial y tangencial.
4. Conversión de los puntos (d) a coordenadas píxel (p), que siempre llevará asociada una cierta distorsión.

En la página siguiente mostramos el proceso completo, junto con las ecuaciones necesarias para efectuar cada paso.

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = D_{adquisición} D_{óptica} P_{ideal} R \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \qquad (3)$$

Dónde R representa la rototraslación, P_{ideal} la proyección ideal, $D_{óptica}$ la distorsión debida a la óptica del sistema y $D_{adquisición}$ la conversión a píxeles con su distorsión asociada.

En el caso de una sola cámara, es costumbre ubicar los ejes de coordenadas del mundo solidarios con dicha cámara, de modo que la primera rototraslación se convierte en una matriz unidad, quedando la expresión, ligeramente más simple[2]:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = D_{adquisición} D_{óptica} P_{ideal} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \qquad (4)$$

La expresión explícita de éstas matrices depende de gran cantidad de parámetros y resulta algo complicada. La acción de cada una se muestra en la página siguiente. La buena noticia es que el software que utilizaremos gestiona todos los parámetros de manera casi automática, haciendo que, a pesar de las apariencias, el proceso resulte bastante cómodo.

[1] Lo denotaremos con el subíndice w.
[2] Evidentemente esto no será posible en el caso de dos cámaras, puesto que el origen de coordenadas nunca podrá estar situado simultáneamente sobre ambas.

Rototraslación

$$\vec{X}_w \longrightarrow \vec{X}_c$$

$$R\vec{X}_w + \vec{T} = \vec{X}_c$$

con:

R: matriz de rotación

T: vector de traslación

Proyección ideal

$$\vec{X}_c \longrightarrow \vec{X}_n$$

$$X_n = \frac{fX_c}{Z_c}$$

$$Y_n = \frac{fY_c}{Z_c}$$

Distorsión óptica

$$\vec{X}_n \longrightarrow \vec{X}_d$$

$$D\vec{X}_n + d\vec{X} = \vec{X}_c$$

con:

$$D = 1 + k_1 r^2 + k_2 r^4 + k_5 r^6$$

$$r^2 = x_n^2 + y_n^2$$

$$d\vec{X} = \begin{bmatrix} 2k_3 x_n y_n + k_4(r^2 + 2x_n^2) \\ 2k_4 x_n y_n + k_3(r^2 + 2y_n^2) \end{bmatrix}$$

Distorsión en la adquisición

$$\vec{X}_d \longrightarrow \vec{X}_p$$

$$KK \begin{bmatrix} X_d \\ Y_d \\ 1 \end{bmatrix} = \begin{bmatrix} X_p \\ Y_p \\ 1 \end{bmatrix}$$

con:

$$KK = \begin{bmatrix} f_1 & \alpha_c f_1 & c_1 \\ 0 & f_2 & c_2 \\ 0 & 0 & 1 \end{bmatrix}$$

3 Calibración de cámaras CCD

A la vista de todo lo anterior resulta evidente que es necesario conocer todos los parámetros involucrados en las transformaciones para poderlas llevar a cabo.

El proceso mediante el cuál se obtienen los valores de dichos parámetros se conoce como calibración. Para tal efecto, utilizaremos una toolbox de Matlab escrita por Jean-Yves Bouguet, y disponible gratuitamente en la web: http://www.vision.caltech.edu/bouguetj/calib_doc/

Su funcionamiento consiste, a grandes rasgos, en lo siguiente: se toman varias fotografías de un patrón como el de la figura siguiente colocado en diversas posiciones.

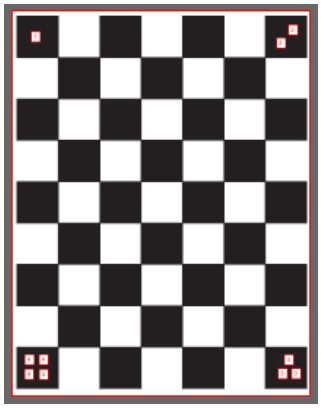


figura 1. Patrón de calibrado

La parte de interés es el rectángulo, formado por 7 filas y 5 columnas que queda tras eliminar las filas y columnas de los bordes.

El usuario debe marcar manualmente la posición de las esquinas de éste rectángulo, en el orden dado por los números indicados en los cuadros rojos. A partir de éste dato, el programa calcula la posición de las esquinas de todos los cuadrados interiores.

Comparando las esquinas así obtenidas con las localizadas a través de un análisis en el cambio de color, y sabiendo además las dimensiones de los cuadrados, el programa es capaz de estimar los valores de los parámetros antes citados, calibrando así la cámara.

Incluye además una herramienta de optimización, que permite mejorar la precisión de los resultados obtenidos. En el siguiente apartado se muestra un breve tutorial del uso de éste software, aplicado directamente a un sistema con dos cámaras.

4 Calibración de cámaras en estéreo

El dispositivo que vamos a utilizar está formado por dos webcams colocadas en un trípode, como se muestra en la figura a continuación:

figura 2. Sistema estereoscópico

En éste caso, dado que las dos cámaras no pueden escogerse simultáneamente como origen de coordenadas[3], será necesario calibrar también los parámetros extrínsecos.

Para tal efecto, tomaremos una serie de unos 15 pares de fotografías del patrón de calibrado simultáneamente con ambas cámaras, como las de la figura siguiente:

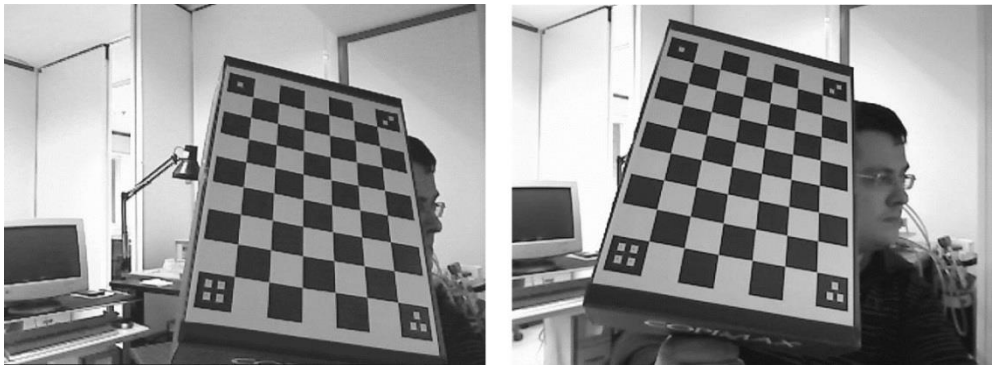


figura 3. Par de imágenes simultáneas

Las imágenes tomadas con la cámara de la izquierda se almacenarán en la carpeta activa de Matlab como imageLn.jpg, y las de la derecha como imageRn.jpg, dónde n es un índice que variará de 1 a N, siendo N el número de fotos tomadas (en el ejemplo aquí descrito se ha hecho N = 15).

A título de ejemplo, describiremos a continuación el proceso de calibrado de parámetros intrínsecos para la cámara L. En primer lugar, debemos abrir el camera calibration toolbox ejecutando en Matlab el comando

[3] Tomaremos como origen de coordenadas el centro de la cámara izquierda.

calib_gui. En la ventana emergente, seleccionamos la opción Standard[4], accediendo así a la ventana principal del programa de calibrado:

figure 4. Interfaz de calib_gui

Pulsamos sobre Image Names para cargar las imágenes tomadas con la cámara izquierda. Nos pedirá que indiquemos el nombre base de las imágenes, que en éste caso es imageL, así como el formato (.jpg).

Una vez cargadas, mostrará un mosaico con las quince imágenes:

figura 5. Resultado tras la carga de imágenes

A continuación, pulsamos sobre Extract Grid Corners, herramienta que usaremos con los parámetros por defecto (procesar todas las imágenes, ventana del buscador de esquinas de 11x11[5], método automático de conteo de cuadros), y a continuación nos pedirá que identifiquemos, en orden, las cuatro esquinas del rectángulo de la primera fotografía.

Una vez que hayamos pinchado con el ratón sobre las cuatro esquinas, se nos pedirá que introduzcamos las dimensiones horizontal y vertical de los cuadrados, que en nuestro caso resultan ser ambas 29 mm. Aparecerá entonces una pantalla en la que se nos mostrará dónde ubica el programa las esquinas de los restantes cuadrados, representadas como cruces rojas (ver figura 6).

[4] La opción Memory Efficient es necesaria en caso de utilizar imágenes muy grandes, y/o un elevado número de ellas.
[5] Esto significa que la identificación manual de las esquinas debe tener como mucho un error de 5 píxeles, de modo que la esquina esté dentro de un cuadrado de tamaño 11x11 píxeles centrado en el puntero del ratón.

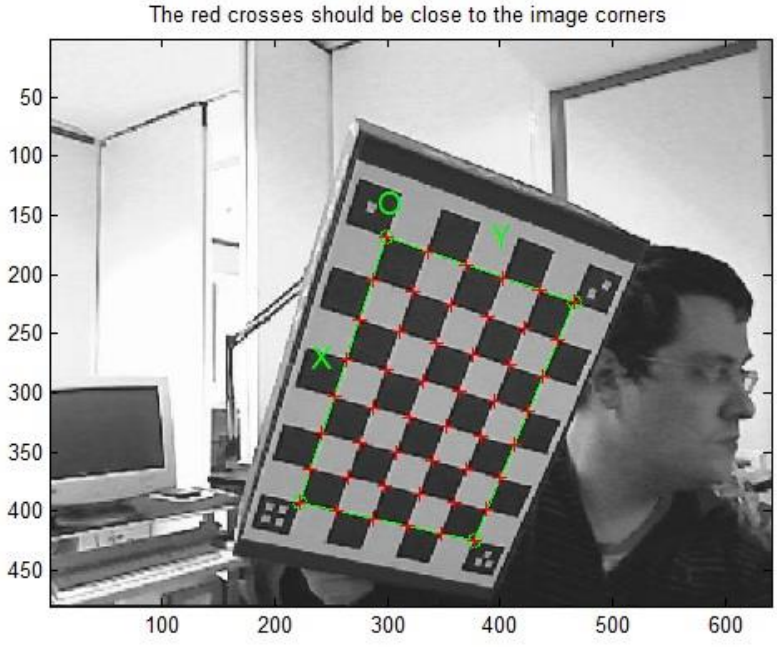


figura 6. Identificación de esquinas

Si éstas coinciden aproximadamente con la posición de las esquinas en la fotografía, no será necesario que introduzcamos un valor inicial para la distorsión.

Debemos repetir éste proceso para todas las fotografías, y una vez terminado, pulsando sobre Calibration obtenemos una primera calibración de los parámetros intrínsecos de la cámara.

La herramienta Analyse Error nos permite visualizar, en píxeles, el error cometido en la reproyección con los parámetros de calibración obtenidos en forma de gráfico.

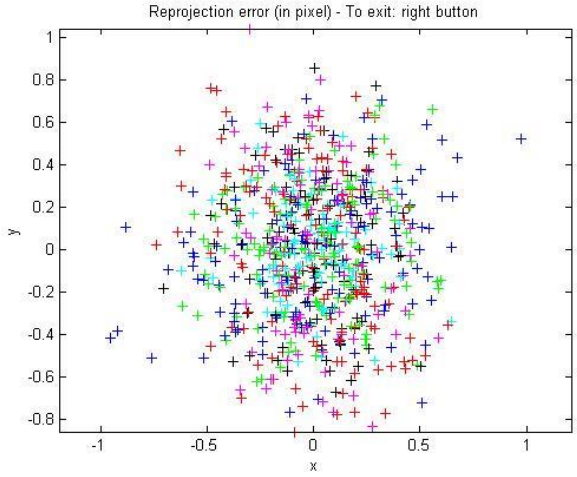


figura 7. Errores de reproyección

Mediante Recomp. Corners podemos llevar a cabo un proceso de optimización que mejora ligeramente la precisión de la calibración.

Una vez satisfechos con los resultados y su precisión, pulsando Save quedan almacenados en un archivo llamado Calib_Results.mat, que renombraremos como Calib_Results_left.mat. El proceso para la cámara derecha es completamente análogo, y en éste caso usaremos el nombre Calib_Results_right.mat.

A continuación, ejecutaremos stereo_gui para proceder a la calibración de los parámetros extrínsecos. El interfaz presenta el aspecto de la figura:

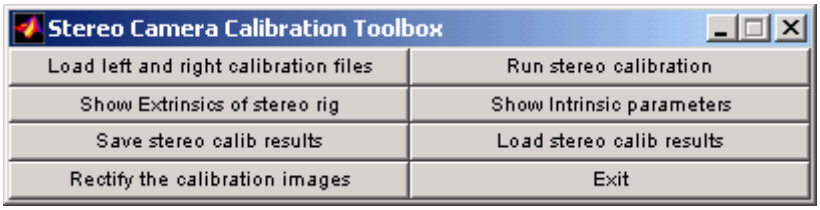


figura 8. Interfaz de stereo_gui

Pulsando sobre Load left and right calibration files, se nos pedirán los nombres de los dos archivos .mat anteriormente citados, que deben encontrarse en la carpeta activa de Matlab. Inmediatamente obtendremos los resultados de calibración para todos los parámetros, intrínsecos y extrínsecos.

Mediante Run Stereo Calibration se pone en marcha un proceso iterativo de optimización que nos permite obtener unos resultados más precisos.

Los resultados obtenidos tras el proceso completo en éste ejemplo son:

Intrinsic parameters of left camera:

Focal Length: fc_left = [560.09952 561.12420] ± [2.64342 2.78444]
Principal point: cc_left = [343.33608 250.35276] ± [4.34693 3.90266]
Skew: alpha_c_left = [0.00000] ± [0.00000] => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion: kc_left = [-0.02435 0.11236 -0.00050 0.00565 0.00000] ± [0.02159 0.09463 0.00240 0.00277 0.00000]

Intrinsic parameters of right camera:

Focal Length: fc_right = [562.50875 561.98770] ± [2.62270 2.75029]
Principal point: cc_right = [324.54642 235.12279] ± [4.37335 3.57900]
Skew: alpha_c_right = [0.00000] ± [0.00000] => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion: kc_right = [-0.04109 0.14165 0.00076 0.00583 0.00000] ± [0.01713 0.05564 0.00208 0.00270 0.00000]

Extrinsic parameters (position of right camera wrt left camera):

Rotation vector: om = [0.05088 0.03428 0.02725]
± [0.00653 0.00927 0.00072]
Translation vector: T = [-75.01179 -1.45643 0.30012]
± [0.22148 0.17398 1.01915]

Por último, utilizando la herramienta Rectify the calibration images ejecutamos un algoritmo de rectificación planar, que genera las imágenes rectificadas y las almacena en formato .bmp para posterior utilización.

La rectificación no sería necesaria si las cámaras estuviesen perfectamente alineadas y no introdujesen distorsiones, es decir, si se comportasen como un sistema ideal. El papel de la rectificación es precisamente el de simular un comportamiento ideal, utilizando los parámetros de calibración para distorsionar las imágenes de modo que cumplan con todas las propiedades de las imágenes capturadas por un sistema ideal.

El uso de imágenes rectificadas tiene dos ventajas importantes; la primera de ellas es que permite trabajar con un modelo de sistema estéreo más sencillo posible (que veremos en el apartado 5), y la segunda es que simplifica enormemente el proceso de búsqueda de puntos correspondientes al hacerlos yacer sobre la misma fila.

Para comprobar visualmente la correcta rectificación, resulta buena idea colocar una pareja alineada y a la misma escala como en la figura adjunta:

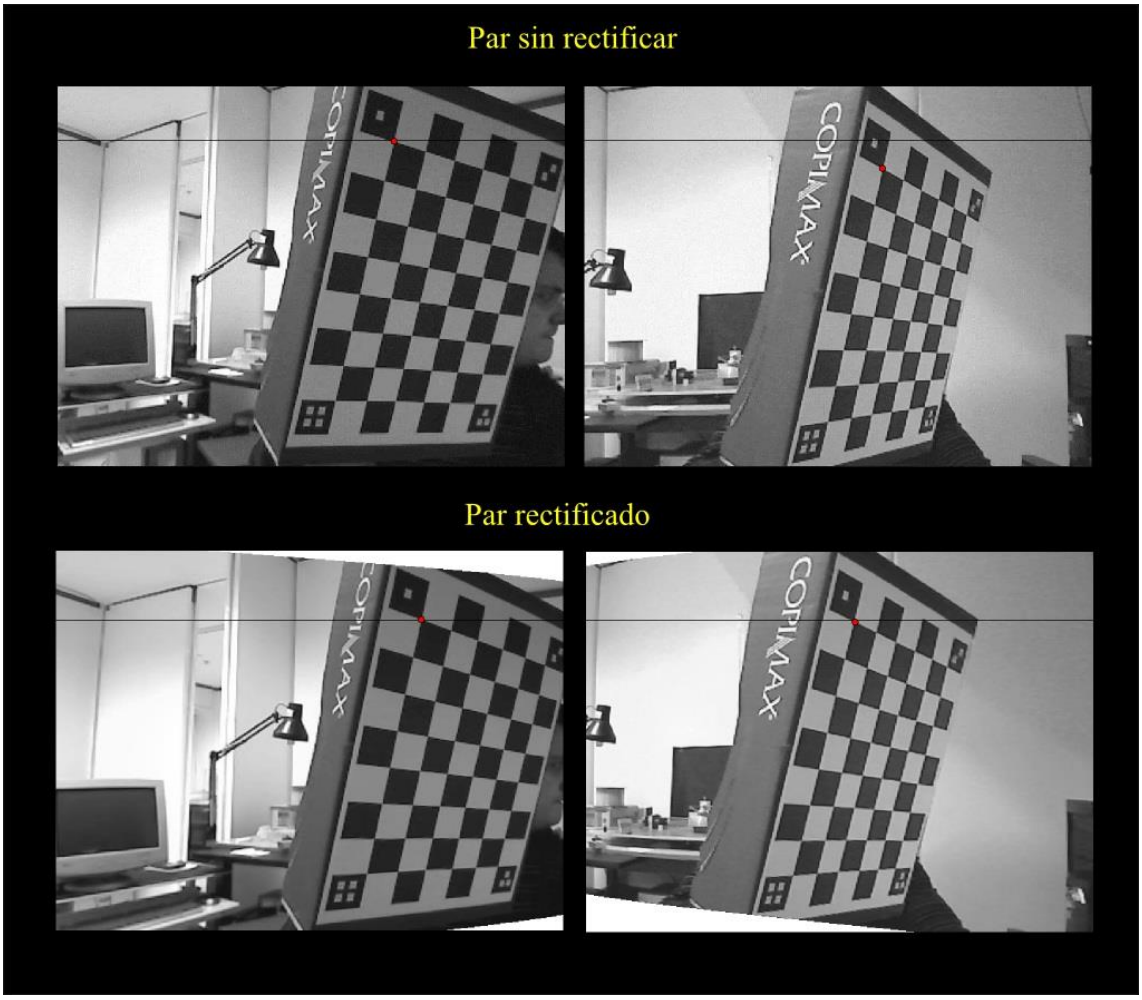


figura 9. En un par rectificado, los puntos equivalentes yacen sobre la misma horizontal.

Del mismo modo podemos rectificar pares de imágenes diferentes de las de calibración, tomadas con la misma configuración del sistema en estéreo.

Utilizando la función stereo_triangulation.m obtenemos las coordenadas de los puntos de calibración en el espacio. Como comprobación, podemos tomar las coordenadas de dos puntos consecutivos y comprobar que la distancia entre ellos es de 29 mm, al menos aproximadamente. También puede ser útil representar éstos puntos en un gráfico 3D.

5 Determinación de la profundidad a partir de la disparidad

El uso de imágenes rectificadas nos permite utilizar el modelo más sencillo de sistema en estéreo, que consiste en dos cámaras pinhole alineadas como las de la figura:

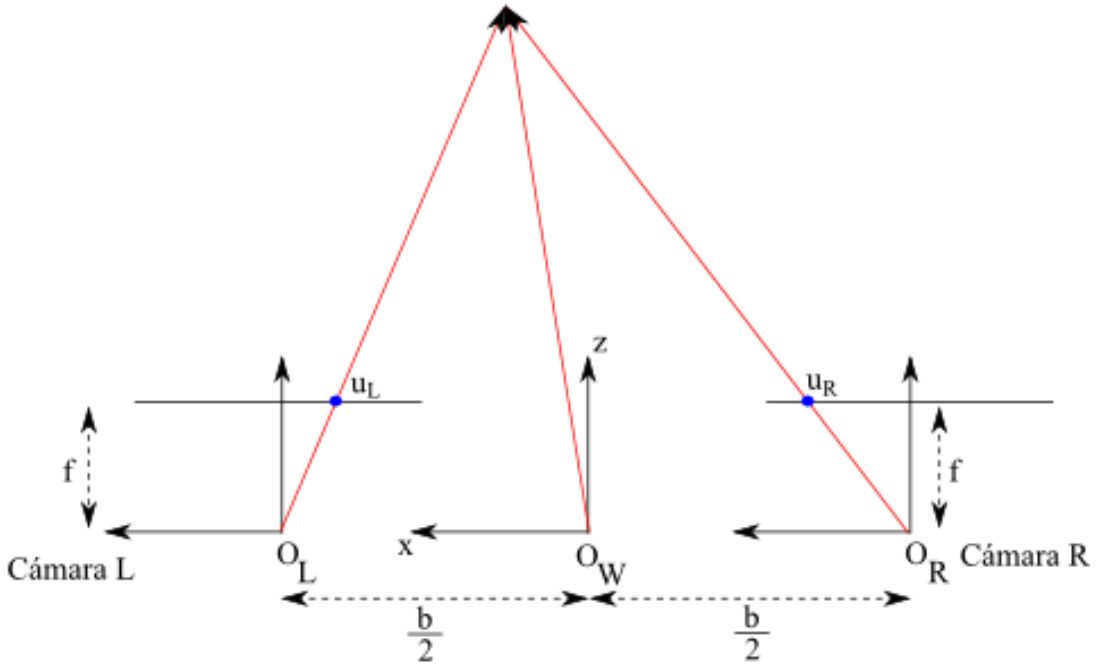


figure 10. Modelo para la configuración en estéreo.

Mediante semejanza de triángulos, es inmediato observar que existe la siguiente relación:

$$\begin{cases} \frac{z_L}{x_L} = \frac{f}{u_L + c_L} \\ \frac{z_R}{x_R} = \frac{f}{u_R + c_R} \end{cases} \quad (5)$$

Despejando las coordenadas u, y teniendo en cuenta que las coordenadas x centradas en las cámaras y las centradas en el sistema W se relacionan mediante una transformación afín sin rotación se obtiene fácilmente:

$$\left.\begin{aligned} u_L + c_L &= \frac{f}{z_W}\left(x_W + \frac{b}{2}\right) \\ u_R + c_R &= \frac{f}{z_W}\left(x_W - \frac{b}{2}\right) \end{aligned}\right\} \rightarrow u_L - u_R = \frac{fb}{z_W} + c_R - c_L \equiv d \quad (6)$$

La separación entre cámaras (b) y la focal (f) son datos que conocemos de la calibración en estéreo. Para calcular la disparidad (d), utilizaremos una librería C++ llamada LIBELAS[6], que devuelve la disparidad en píxeles de un par estéreo de imágenes rectificadas.

[6] LIBrary for Efficient LArge scale Stereo Matching. http://www.rainsoft.de/software/libelas.html

Veamos un ejemplo de mapa de disparidad, dónde se representa una mayor disparidad con colores cálidos y viceversa:

figura 11. Mapa de disparidad de una escena.

Los puntos azul oscuro son aquellos para los que el software no ha logrado encontrar una pareja de puntos equivalentes. Esto puede deberse a que dicho punto no era visible desde ambas cámaras simultáneamente, o de un modo más prosaico aún, a un simple error en el algoritmo de identificación de puntos equivalentes.

Una vez obtenido el mapa de disparidad, basta aplicar la ecuación (4) para obtener un mapa de profundidad (z_W). Logramos de ésta manera asignar la profundidad correspondiente a cada píxel de la imagen.

Una vez asignada la coordenada z_W a cada píxel, las ecuaciones (3) nos permiten calcular también las dos coordendas restantes.

$$\begin{cases} x_W = \dfrac{z_W}{f} \cdot (u - c_1) \\ y_W = \dfrac{z_W}{f} \cdot (v - c_2) \end{cases} \qquad (7)$$

6 Imágenes de ejemplo

Recapitulando, lo que hemos conseguido es asociar unas coordenadas espaciales a cada uno de los píxels de una imagen rectificada. Nuestros resultados son, por tanto, tres matrices (X, Y, y Z) con tantos elementos como la fotografía original[7]. Mostramos a continuación una representación gráfica de dichos resultados, obtenidos a partir del siguiente par de imágenes:

figura 12. Imágenes rectificadas

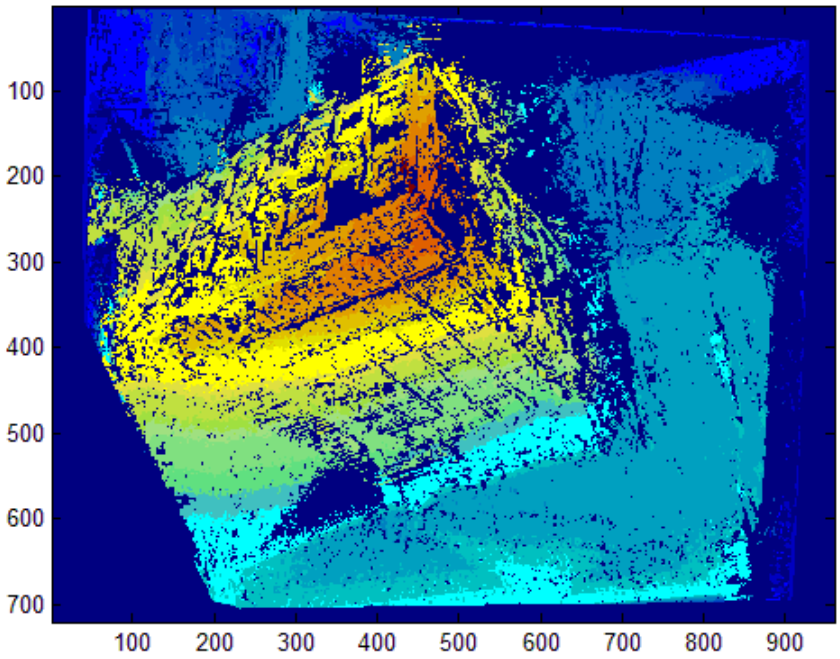


figure 13. Mapa de disparidad

[7] 720 x 960 en el ejemplo que mostraremos a continuación.

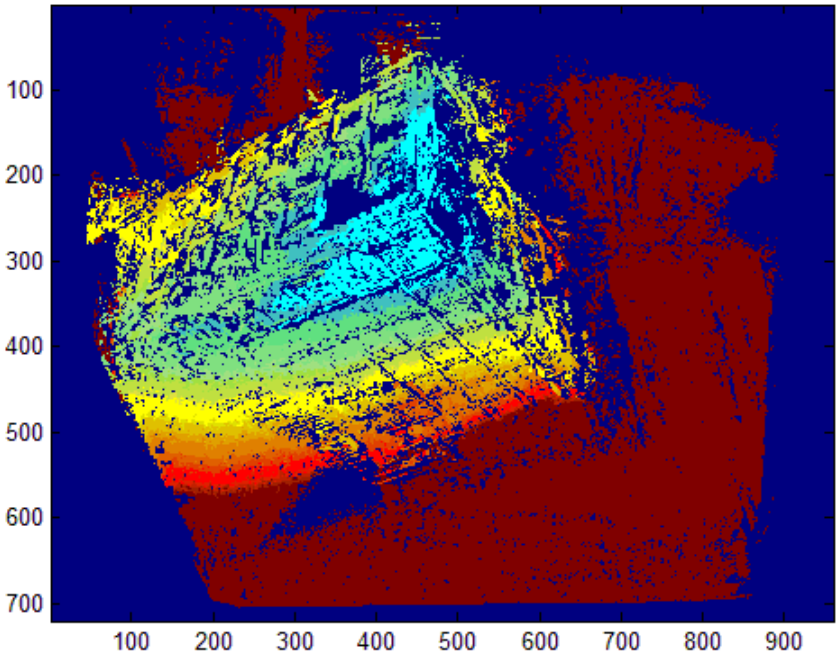


figure 14. Coordenada Z

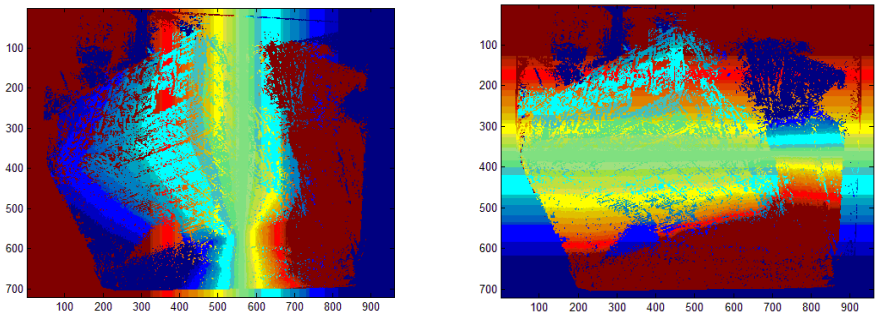


figura 15. Coordenadas X e Y

Como comprobación de la precisión del método, es recomendable tomar las tres coordenadas de al menos un par de puntos que se hayan resuelto correctamente y calcular el módulo del vector que los une. Dicho módulo debe coincidir con la distancia real entre los puntos del objeto. En el caso que nos ocupa, el dispositivo mide una altura para la caja de 217 mm, siendo su altura real de 215 mm, de modo que su precisión es bastante aceptable.

Es importante notar que no todos los puntos se resuelven con la misma exactitud. Para resolver correctamente puntos lejanos es necesario, entre otras cosas, aumentar la distancia entre cámaras. Para posibles aplicaciones de telemetría es fundamental tener en cuenta éste tipo de consideraciones.

7 Integración del software utilizado

Gran parte del presente trabajo académicamente dirigido ha consistido en lograr integrar los paquetes de software Camera Calibration Toolbox y LIBELAS. Para tal efecto ha sido necesario hacer una pequeña modificación en LIBELAS, así como escribir un programa que, con los datos proporcionados por los dos anteriores, calcule las coordenadas X, Y y Z de cada píxel de la imagen. A continuación se detallan éstos pequeños cambios.

7.1 Parche en LIBELAS

LIBELAS está escrito en C++, de modo que para ejecutarlo en Matlab es necesario utilizar wrappers. Los archivos necesarios, así como las instrucciones para llevar a cabo éste proceso se pueden encontrar en la página web del software[8] de modo que no insistiremos más en ese asunto.

Para llevar a cabo éste proyecto, hemos añadido una leve modificación en el archivo process.m , que es el "corazón" de LIBELAS. Lo detallamos a continuación:

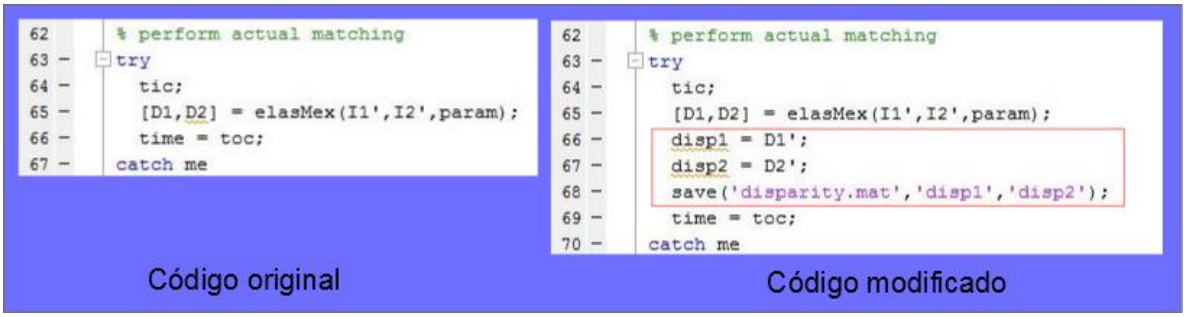


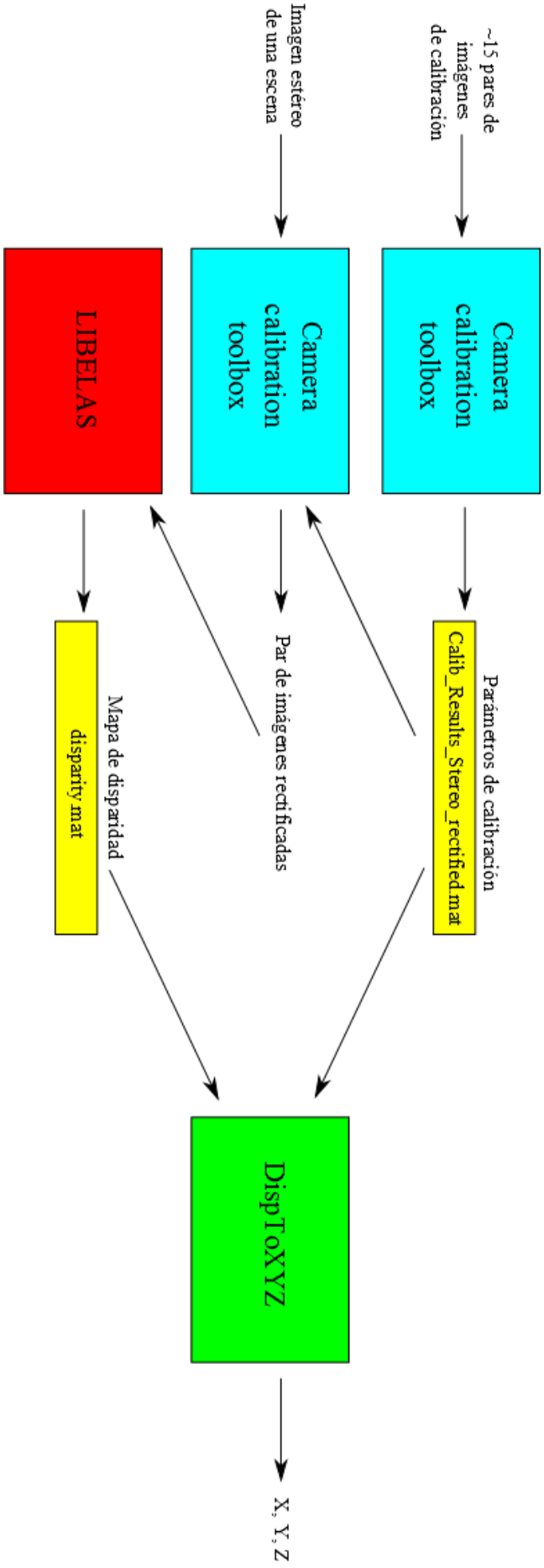
figura 16. Modificación en process.m

El objeto de esto es que los mapas de disparidad, además de mostrarse en pantalla, se guarden para posterior uso.

7.2 Integración del proceso

Ya hemos visto cómo se calculan las coordenadas X, Y y Z a partir del mapa de disparidad desde un punto de vista teórico; la realización práctica de éste proceso se resume en el diagrama de flujo de la página siguiente.

[8] http://www.rainsoft.de/software/libelas.html


~15 pares de imágenes de calibración
Imagen estéreo de una escena
Camera calibration toolbox
Camera calibration toolbox
LIBELAS
Parámetros de calibración
Calib_Results_Stereo_rectified.mat
Par de imágenes rectificadas
Mapa de disparidad
disparity.mat
DispToXYZ
X, Y, Z

Dónde el programa llamado DispToXYZ es un programa de Matlab escrito exprofeso para hacer la función indicada. Presentamos su código a continuación.

7.3 Código de DispToXYZ

```
function [X, Y, Z] = DispToXYZ(disp, KK_left_new, KK_right_new, T_new, fc_left_new,
cc_left_new)

% Calcula las coordenadas X, Y, Z de cada píxel de una imagen a partir de su mapa de
disparidad.
%
% Éste programa está diseñado para funcionar con los datos de calibración generados por
Camera Calibration Tool, de Jean-Yves Bouguet (
http://www.vision.caltech.edu/bouguetj/calib_doc/ ) y con los mapas de disparidad
generados con LIBELAS, de Andreas Geiger
% ( http://www.rainsoft.de/software/libelas.html ).
%
% Síntaxis:
% [X, Y, Z] = DispToXYZ(disp, KK_left_new, KK_right_new, T_new, fc_left_new,
cc_left_new);
% [X, Y, Z] = DispToXYZ(disp)   carga automáticamente los datos de
% calibración del archivo 'Calib_Results_stereo_rectified.mat'.
% [X, Y, Z] = DispToXYZ()   carga además el mapa de disparidad del archivo
'disparity.mat'.
%
% Por Pablo Rodríguez Sánchez. UCM. Mayo de 2012.

% Si el usuario no introduce parámetros se cargan automáticamente del
% espacio de trabajo:

% ---------------
if nargin == 0
    load('disparity.mat'); load('Calib_Results_stereo_rectified.mat'); disp = disp1;
elseif nargin == 1
    load('Calib_Results_stereo_rectified.mat');
else
end
% ---------------
% En primer lugar obtenemos la coordenada Z:
% ---------------
f1L = KK_left_new(1,1); f1R = KK_right_new(1,1);
f_mean = (f1L + f1R)/2;
c1L = KK_left_new(1,3); c1R = KK_right_new(1,3);
b = norm(T_new);
Z = f_mean * b ./ (disp - c1R + c1L);
% ---------------
% Una vez obtenida Z, calculamos las dos restantes:
% ---------------
[Sx, Sy] = size(Z);

for i = 1:Sx
    for j = 1:Sy
        X(i,j) = (Z(i,j)/fc_left_new(1)) * (j - cc_left_new(1));
        Y(i,j) = (Z(i,j)/fc_left_new(2)) * (i - cc_left_new(2));
    end
end
% ---------------
```

8 Conclusiones y comentarios

El autor considera que la aproximación al mundo de la ingeniería que ha supuesto el presente trabajo académicamente dirigido ha sido una enseñanza enormemente útil. Hasta la fecha, en su experiencia como estudiante se había visto obligado a lidiar rara vez con problemas prácticos. Resulta curioso (al menos para el neófito) comprobar hasta qué punto, en éste tipo de problemas, los detalles suelen ser mucho más trabajosos que los procesos principales.

En un orden de cosas más filosófico, el presente trabajo muestra con bastante elocuencia cómo una idea simple puede dar lugar a aplicaciones muy interesantes. Es de resaltar el hecho de que bastan unos conocimientos básicos de geometría para entender los fundamentos del problema de la percepción de la profundidad.

Hay, sin embargo, un asunto fascinante, y cuyas matemáticas distan mucho de ser triviales, sobre el que hemos pasado de puntillas; me refiero a la identificación por parte del software de los puntos equivalentes. Hemos podido llegar a nuestra meta a pesar de ésta carencia gracias a que hemos utilizado un software (LIBELAS) ya escrito y plenamente funcional.

A pesar de habernos servido de paquetes de software creados por profesionales, ha costado cerca de tres meses de trabajo lograr convertir lo que en un principio eran un par de cámaras web en un trípode en un sistema capaz de determinar las coordenadas espaciales de cada punto.

Tras comprender y experimentar “en propias carnes” la dificultad de éste proceso, el autor no puede menos que sobrecogerse ante la naturalidad y sencillez aparentes sus propios ojos, sistema estereoscópico del que él mismo disfruta, a pesar de ser miope.

9 Bibliografía

9.1 Libros consultados

HARTLEY, Richard y ZISSERMAN, Andrew. *Computer vision*. Cambridge University Press, 2000.

PAJARES, Gonzalo y de la CRUZ, Jesús M. *Visión por computador*. Madrid. Ra-Ma, 2001.

9.2 Sitios web consultados

Camera Calibration Toolbox:

http://www.vision.caltech.edu/bouguetj/calib_doc/

LIBELAS:

http://www.rainsoft.de/software/libelas.html

Matlab documentation center:

http://www.mathworks.es/help/

Wikipedia:

http://en.wikipedia.org/wiki/Epipolar_geometry
http://en.wikipedia.org/wiki/Projective_geometry
http://en.wikipedia.org/wiki/Homogeneous_coordinates
http://en.wikipedia.org/wiki/Pinhole_camera_model
http://en.wikipedia.org/wiki/Image_rectification
http://en.wikipedia.org/wiki/Computer_stereo_vision
http://en.wikipedia.org/wiki/Computer_vision