Calibración de un mapa digital

1. INTRODUCCIÓN: En esta práctica vamos a ver algunos problemas prácticos donde usar los conocimientos del tema de ajuste de datos.

En la primera parte (directorio calib) partiremos de un mapa topográfico en papel que se ha escaneado y guardado como una imagen ('mapa.jpg'). En MATLAB, para leer y visualizar un fichero JPEG usaremos los comandos imread y image:

```
>> im = imread('mapa.jpg'); image(im);
```

La imagen es un escaneado de un mapa topográfico de la Sierra de Gredos. Nos gustaría usar este mapa digital para conocer las coordenadas geográficas de los distintos puntos al movernos por el mapa (p.e. para planificar un recorrido y subirlo a un GPS). Desde MATLAB podemos saber en qué píxel de la imagen se encuentra nuestro ratón, pero nos hace falta la relación entre los píxeles (X, Y) y las correspondientes coordenadas (E, N) geográficas.

Conociendo ciertos datos, como las coordenadas de algún punto de la imagen y la resolución de la imagen en metros/píxel podríamos intentar establecer dicha relación entre píxeles y coordenadas. Con cuidado, podríamos intentar estimar estos parámetros. La resolución del mapa podríamos deducirla de la escala del mapa original y la densidad de escaneo (DPI, o puntos por pulgada). También podríamos localizar algún punto de coordenadas conocidas en el mapa.

Sin embargo, pronto se detectaríamos algunos problemas adicionales: el mapa podría estar girado al ponerlo en el scanner, por lo que las coordenadas E Y N no irían exactamente en la dirección de los píxeles escaneados X e Y. También es posible que la resolución del scanner no sea exactamente la misma en ambos ejes, etc. Veremos como las técnicas de ajuste de datos pueden ayudarnos a solucionar este problema sin tener cuantificar todos esos aspectos que pueden alterar la relación píxeles/coordenadas que deseamos hallar.

El problema de la **calibración de un mapa digital** consiste en determinar una relación entre los valores de los píxeles y sus correspondientes coordenadas geográficas. Una sencilla relación que cubre los problemas antes mencionados son las transformaciones afines. La expresión general de dicha transformación de píxeles (X,Y) a coordenadas (E,N) es la siguiente:

$$\begin{pmatrix} E \\ N \end{pmatrix} = \begin{pmatrix} c_1 & c_2 \\ c_3 & c_4 \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} + \begin{pmatrix} oE \\ oN \end{pmatrix} = M \cdot \begin{pmatrix} X \\ Y \end{pmatrix} + \overline{D} \qquad (1)$$

donde **(E,N)** son las coordenadas geográficas y **(X,Y)** el píxel sobre la imagen.

Hay 6 parámetros que caracterizan la transformación afín, cuatro (**c1,c2,c3,c4) agrupados** en la matriz M (con información sobre escala, giros, cizalladuras, ...) y dos (**oE,oN**) en el vector **D** que pueden interpretarse como las coordenadas E,N correspondientes al origen (X=Y=0) de la imagen.

Lo que queremos es estimar los parámetros del ajuste (obtener la matriz M y el vector **D**) sin tener que medir cuidadosamente sobre el mapa, conocer el giro al escanear ni tener que estimar la resolución/escala de nuestro mapa digital, etc.

El problema puede plantearse como un problema de ajuste si disponemos de un conjunto de puntos de control, esto es, una lista o tabla de píxeles sobre la imagen **{Xk,Yk}** con coordenadas conocidas **{Ek,Nk}**. Dada la lista de control se trata de hallar los 6 parámetros (**c1,c2,c3,c4,oE,oN**) que mejor verifican:

$$E_k \approx c_1 \cdot X_k + c_2 \cdot Y_k + oE$$
$$N_k \approx c_3 \cdot X_k + c_4 \cdot Y_k + oN$$

Estas ecuaciones de ajuste pueden plantearse como dos problemas separados:

1) hallar parámetros c1,c2,oE que mejor transforman los píxeles **{Xk, Yk}** en la correspondiente coodenada **{Ek}**:

$$E_k \approx c_1 \cdot X_k + c_2 \cdot Y_k + oE$$

2) Repetir el ajuste en las coordenadas **{Nk}** para los parámetros c3, c4, oN:

$$N_k \approx c_3 \cdot X_k + c_4 \cdot Y_k + oN$$

Plantear ambos problemas de ajuste (por separado) y escribirlos como nuestros típicos sistemas sobredeterminados:

$$\begin{pmatrix} & \\ H & \\ & \end{pmatrix} \begin{pmatrix} \\ \overline{C} \\ \end{pmatrix} \cong \begin{pmatrix} \\ \overline{V} \\ \end{pmatrix}$$

1.1 Identificar cuál es la matriz H, el vector de coeficientes **C** y el vector **V** de los datos a ajustar en cada uno de los casos (ajuste en coordenadas Este y Norte). Adjuntar scanner/foto de vuestro deducción y resultado.

¿Algunos de los elementos de ambos problemas son comunes?

Tras plantear los sistemas anteriores hay que resolverlos para encontrar c1,c2,oE en un caso y c3,c4,oN en el otro. Esa parte es muy sencilla con lo que sabemos sobre resolver sistemas sobredeterminados con MATLAB. Una vez conocidos los parámetros los organizaremos en una matriz M y un vector **D**, de la forma indicada en (1):

$$M = \begin{pmatrix} c_1 & c_2 \\ c_3 & c_4 \end{pmatrix} \qquad \overline{D} = \begin{pmatrix} oE \\ oN \end{pmatrix}$$

Planteado el sistema ahora hay que obtener los datos para poder construirlo. Lo que necesitamos es una lista de puntos de control, con sus correspondientes posiciones en píxeles (Xk,Yk) y en coordenadas (Ek,Nk). En la siguiente lista damos las coordenadas geográficas de una serie de puntos distintivos del mapa:

	Coordenada E (UTM) mt.	**Coordenada N (UTM) mt.**
Cabeza Nevada	305090	4460830
Pto. Candeleda	309535	4457780
Plataforma (K12)	310290	4460895
Refugio del Rey	308485	4458290
Peña Rayo	307030	4460170
Almanzor	304690	4457805
Casquerazo	305890	4457240
La Galana	304555	4458945

La lista anterior corresponde a los datos {Ek,Nk}. Para poder completar nuestra "tabla" de puntos de control nos faltan las posiciones en píxeles {Xk,Yk} de cada punto. Podríamos usar cualquier programa de manejo de imágenes para localizar los puntos de control sobre el mapa y anotar sus correspondientes coordenadas (en píxeles). Para hacer menos aburrida esta tarea he escrito una pequeña aplicación en MATLAB. Para ver su interfaz simplemente teclear:

>> map Tras cargar mapa.jpg (File/Load Map) veréis algo parecido a esto:

El programa muestra un mapa en la ventana de la izquierda y una serie de botones y paneles (derecha) que nos permitirán ver e introducir los datos.

Abajo a la derecha hay un cuadro de texto con las coordenadas del punto de la imagen donde está situado el ratón. Como todavía no tenemos la conversión entre píxeles y coordenadas, en dicha ventana sólo mostramos por ahora el píxel (x,y) del ratón. Si lleváis el cursor a la esquina superior izquierda veréis que

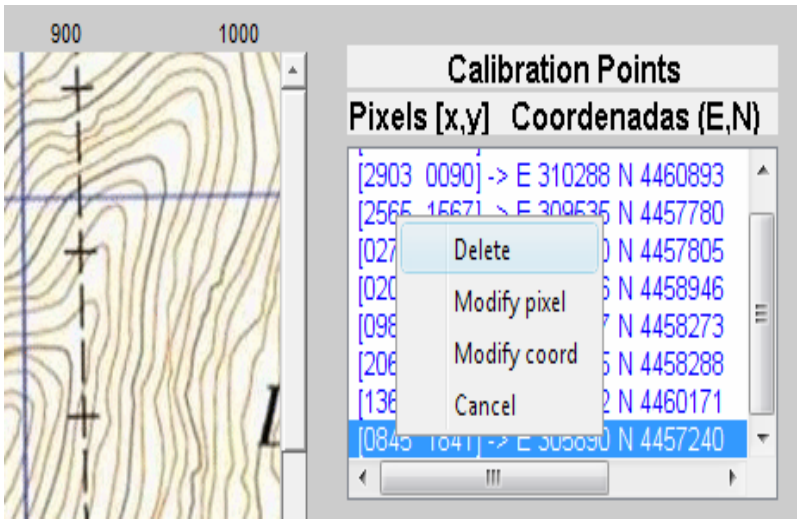
marca pixel = (1,1), que es el criterio habitualmente usado en imágenes, donde la coordenada Y crece hacia abajo y la coordenada X hacia la derecha.

Con los "sliders" que hay junto a la imagen podemos desplazarnos por el mapa, buscando los diferentes puntos de control (marcados con unos círculos rojos para facilitaros la tarea). Una vez localizado cada uno de ellos hay que introducir sus coordenadas. Para ello pincharemos en el botón "MARCAR PTO DE CONTROL" y aparecerá un cursor en forma de mira. Llevaremos el cursor sobre el punto deseado en el mapa y lo marcamos con el ratón (botón izqdo). La posición (en píxeles) del punto se captura y se muestra en los cuadros Xpix e Ypix. Las ventanas East y North se ponen ahora en verde, indicando que se pueden meter las coordenadas del punto en cuestión. Tras introducirlas pulsaremos el botón Añade y ambos pares de coordenadas se añaden a la lista de puntos de control mostrada en la parte superior.

Repitiendo lo anterior con los otros puntos, el programa dispondrá de la lista de posiciones (píxeles y coordenadas) para llevar a cabo el ajuste. Cuando terminéis de introducir los puntos de control, guardar los datos en un fichero. Usar el menú Calibración/Save Calib para elegir un nombre de fichero donde guardar la lista de puntos de control. De esta forma si cerráis el programa y volvéis a entrar podéis cargar (Load Calib) el fichero con los datos de calibración sin tener que volver a marcar los puntos de control. Esto nos permitirá también acceder a la lista de los puntos de control desde fuera de la aplicación map.

Si nos equivocamos al meter algún dato (tanto al pinchar sobre la imagen como al dar las coordenadas) es posible modificar un punto de control.

Para ello, marcar el punto en la lista y pulsar con el botón derecho del ratón para obtener el menú que se muestra en la figura adjunta. Podemos usarlo tanto para borrar como para modificar (píxels/coordenadas) un punto ya introducido.



2. CÓDIGO del AJUSTE

A pesar de haber introducido los puntos de control las coordenadas mostradas siguen siendo píxeles y no coordenadas geográficas. Para hallar los parámetros de la transformación afín está el botón AJUSTE. Al pulsarlo, la aplicación llama a una función ajuste.m que recibe la información de los puntos de control (píxeles XY + coordenadas E,N) y debe devolver la matriz M y el vector D obtenidos para la transformación (de píxeles a coordenadas). Si pulsáis AJUSTE tampoco pasa

nada porque la función está medio vacía y tenéis que completarla vosotros. Si abrís el fichero ajuste.m en el editor de MATLAB veréis esta cabecera:

```
function [M D]=ajuste(Xk,Yk,Ek,Nk)
% Entrada:
%   Xk: vector N x 1 con las X's de los N ptos de control
%   Yk: vector N x 1 con las Y's de los N ptos de control
%   Ek, vector N x 1 con las coord. E's de los N ptos de control
%   Nk, vector N x 1 con las coord. N's de los N ptos de control
% Salida: parámetros de la transformación de píxeles --> coordenadas
%   M : matriz 2 x 2
%   D : vector 2 x 1
N = length(Xk); if (N<3), return; end

D = [0 0]'; M = [1 0; 0 1];  % Ajuste por defecto (no hace nada)

   % Aqui iría vuestro código
   % ...
return
```

Tal como está, la función no hace nada ya que devuelve una matriz M identidad y un vector D nulo (que corresponden a la transformación identidad).

2.1 ¿Cuál es la razón de la comprobación inicial del número de puntos N usados?

Hay que completar el código de la función ajuste.m para que haga lo siguiente:

1) Cree las matrices H y vectores de datos V de los 2 problemas de ajuste a resolver a partir de los datos de entrada (usar vuestra deducción anterior).

2) Resolver ambos problemas de ajuste para obtener los seis parámetros de la transformación afín **(c1,c2,oE) + (c3,c4,oN)**.

3) Construir la **matriz M** a partir de los parámetros **(c1,c2,c3,c4)**, y el **vector D** (vector columna!!) a partir de los coeficientes **(oE,oN).**

   Volcar los valores de la matriz M y el vector D obtenidos tras el ajuste. Para la matriz M basta el volcado que hace MATLAB con 4 decimales. Para D usar fprintf() y volcarlo con 1 decimal (%.1f).

4) Una vez resuelto el ajuste estimar el vector de residuos (discrepancias) tanto en las coordenadas Este como Norte. Volcar (usando fprintf) dichos valores por pantalla (con 1 decimal, %.1f). Poner en una línea los residuos en las coordenadas E y en otra los de las coordenadas N. El resultado debe ser algo similar a esto (aunque los valores no tienen por qué coincidir):

   Residuos Este -1.0 0.2 0.1 -0.3 -0.1 2.5 -2.5 0.5 …
   Residuos Norte 2.3 0.6 3.2 3.1 0.1 -3.4 -2.9 -1.5 …

Para escribir e ir probando la función ajuste.m no necesitamos usar la aplicación map. Lo único necesario es la información que tenemos de los puntos de control para usarlos como argumentos de la función. Para ello:

1) Antes de abandonar la aplicación, guardar los puntos de calibración usando la opción Calibracion/Save Calib del menú. Elegir un nombre de fichero donde guardarlos. Una vez guardados, cerrad la aplicación map.

2) Desde fuera de la aplicación hacer **`>> clear; load fichero`** con el nombre del fichero con los datos de calibración. Haciendo un whos veréis 4 vectores Nx1 llamados Xk, Yk, Ek y Nk. Si volcáis los valores de [Ek Nk] por pantalla debéis ver las coordenadas de la tabla. En el otro par de vectores Xk e Yk están los valores de los píxeles (X,Y) donde caen los puntos de control sobre el mapa (donde pinchamos antes con el ratón).

3) Llamar a la función ajuste, pasándole los parámetros adecuados para ir así ejecutando y probando el código.

Ir añadiendo código poco a poco al fichero ajuste.m suministrado. Al principio ir volcando los resultados intermedios (matriz H, vector V, coeficientes) para ir verificando que vuestro código hace lo que debe. Como siempre, no intentéis escribir todo el código antes de probarlo por primera vez.

Los residuos que os pido calcular al final os servirán también como comprobación de que lo estáis haciendo bien. Debéis obtener residuos del orden de unos pocos metros (los residuos tienen las mismas unidades que las coordenadas E, N que estamos ajustando). Si vuestros residuos llegan a 10 m. es posible que vuestro código sea correcto pero hayáis sido poco cuidadosos al pinchar en los puntos. Valores mucho más altos indican alguna errata en el código de ajuste.m o una equivocación al introducir las coordenadas. En ese caso verificar que la lista de coordenadas en {Ek, Nk} coincide con las coordenadas dadas en la tabla y que los píxeles caen sobre los puntos de control en el mapa.

2.2 Adjuntad código de vuestra función ajuste.m una vez completada

2.3 Adjuntar la matriz M y vector D obtenidos durante el ajuste.

2.4 Volcado de los residuos obtenidos en coordenadas Este y Norte (%.1f). Media de los valores de los residuos en una y otra coordenada. ¿Sería posible encontrar otra solución con la que saliera un valor menor en esas medias?

2.5 Los elementos de la diagonal de la matriz M tienen un valor similar pero con signo cambiado. ¿Cuál es ese valor? ¿A qué corresponde?
¿Cuál es la razón del signo opuesto?

Cuando la rutina esté funcionando, podéis ya usarla dentro de la aplicación map. Una vez que vuestra rutina se ha encargado de hallar la matriz M y vector D mi aplicación usa M y D para pasar de píxeles (X,Y) a coordenadas (E,N) siguiendo la relación vista antes.

Lanzar map y cargar los puntos de control. Al pulsar AJUSTE las coordenadas que da el programa deben ser ahora coordenadas geográficas y no píxeles

Podemos verificar si todo está funcionando sabiendo que las líneas azules del mapa corresponden a una cuadrícula con un salto de 1km. en ambas direcciones. Posicionando el cursor sobre una intersección de la malla tenéis que ver unas coordenadas muy cerca de un múltiplo de 1000 mt. Pasando a la siguiente línea a la derecha debéis ganar 1000 m en la coordenada Este. Moviéndonos hacia abajo en el mapa veréis bajar la coordenada N en 1000 mt.

2.6 Supongamos que queréis llegar al refugio Elola, ¿qué coordenadas destino pondríais en el GPS? (el refugio está por el centro del mapa cerca de una laguna grande). Adjuntar una captura de pantalla de la aplicación con el cursor sobre el refugio, de forma que se vea su posición en la ventana de coordenadas.

3. TRANSFORMACIÓN INVERSA (en esta parte no usamos la aplicación map)

La transformación obtenida (matriz M+vector D) permite a la aplicación convertir píxeles (X,Y) en coordenadas geográficas (E,N):

$$\begin{pmatrix} E \\ N \end{pmatrix} = M \begin{pmatrix} X \\ Y \end{pmatrix} + D \quad (1)$$

Esto nos puede servir para planear una ruta, marcando un recorrido sobre la imagen del mapa y convirtiendo los píxeles (X,Y) de la imagen en coordenadas geográficas (E,N), que podemos volcar a nuestro GPS.

En otras situaciones nos interesa lo contrario, convertir coordenadas geográficas (E,N) en píxeles (X,Y). Una posible aplicación sería solapar sobre la imagen del mapa un recorrido grabado con un GPS. El GPS nos da una lista de coordenadas geográficas (E,N) y si sabemos convertirlas en píxeles podríamos superponer el recorrido sobre la imagen del mapa para ver por donde hemos estado.

En este caso lo que necesitamos es la transformación "inversa" de la anterior que pase del vector de coordenadas (E,N) al correspondiente vector de píxeles (X,Y):

$$\begin{pmatrix} X \\ Y \end{pmatrix} = M' \begin{pmatrix} E \\ N \end{pmatrix} + D' \quad (2)$$

Vamos a calcular M' y D' para nuestro problema. Para ello se trata de despejar (X,Y) de la ecuación original (1) y deducir los valores para la nueva matriz M' y vector D' comparando el resultado con la expresión (2).

3.1 Deducir la expresión de M' y D' a partir de la M y D originales y adjuntar foto de deducción.

Cargar los datos {Xk, Yk, Ek, Nk} y usar ajuste.m para obtener los valores de M y D (serán los mismos que hemos encontrado antes).

3.2 Aplicad a dicha M y D vuestra deducción anterior para obtener M' y D' de la transformación inversa. Volcar la nueva matriz M' y vector D' obtenidos.

Verificad vuestros resultados aplicando la transformación dada por M y D a las coordenadas de un píxel cualquier para encontrar sus coordenadas (E,N). Si ahora aplicáis M' y D' a esas coordenadas deberíais volver al píxel inicial, ya que la segunda transformación es la inversa de la primera.

Una vez comprobada la conversión de coordenadas a píxeles la aplicaremos a un montón de puntos correspondientes a una ruta grabada por el GPS, para luego representarlos superpuestos sobre la imagen del mapa. Cread un script donde:

a) Cargar los datos haciendo **load ruta**. Veréis una variable **ruta** como una matriz de 2 x 350. Cada una de las 350 columnas corresponde a una pareja de coordenadas (E, N) guardada por un GPS durante un recorrido.

b) Aplicar la transformación (2) definida por M', D' al conjunto de posiciones de la ruta. En MATLAB esto puede hacerse de forma muy sencilla multiplicando la matriz M' por la variable ruta, de forma que se aplica a todas las columnas. Luego basta sumar a cada columna del resultado el vector D', para completar la transformación XY = M' · EN + D'. El resultado final es una matriz del mismo tamaño (2 x N) que la original pero ahora conteniendo las posiciones **(X,Y)** en píxeles, listas para pintarse sobre el mapa.

c) **Aseguraros de cerrar la aplicación map** y cargar "manualmente" la imagen contenida en 'mapa.jpg' (imread) y visualizarla usando:

```
image(im);
```

Hacer un hold on (para no borrar el mapa) y superponer el recorrido haciendo un plot de las posiciones en píxeles de los puntos de la ruta.
Usar una línea discontinua de color rojo ('r:') y un grueso de línea 2:

```
>> plot( . . . , 'r:','LineWidth',2)
```

3.3 Adjuntar vuestro código implementando los pasos a), b) y c) anteriores.
Adjuntar la imagen del mapa con la ruta superpuesta.

En esta práctica es conveniente NO USAR Edit/Copy Figure para adjuntar figuras en el archivo Word. Es mejor usar File/Save As para guardar la imagen como un JPG que luego podemos adjuntar al fichero Word (o hacer una captura de pantalla).

La razón es que Copy Figure copia toda la información de la figura (en este caso la imagen del mapa de varios Mbytes). Save As jpg guarda una captura de la ventana como una imagen JPG, ocupando mucho menos.

De esta forma evitaréis tener problemas al exceder el límite en la entrega Moodle.

4. CREACIÓN de PANORAMAS: (30%)

En esta parte estudiaremos el problema (*image stitching*) de enlazar varias imágenes digitales para crear una imagen compuesta o mosaico. La idea es tomar varias imágenes de un mismo motivo y posteriormente combinarlas en el ordenador. Esto posibilita crear imágenes con mayor resolución que la de nuestra cámara o con una relación de aspecto (cociente ancho/alto) no soportada por nuestra cámara (por ejemplo un panorama).

Aunque parezca un problema distinto al que hemos visto en la primera parte de la práctica el problema matemático subyacente es el mismo.

Consideremos el problema de unir dos imágenes con un cierto solapamiento, como por ejemplo las mostradas a continuación:

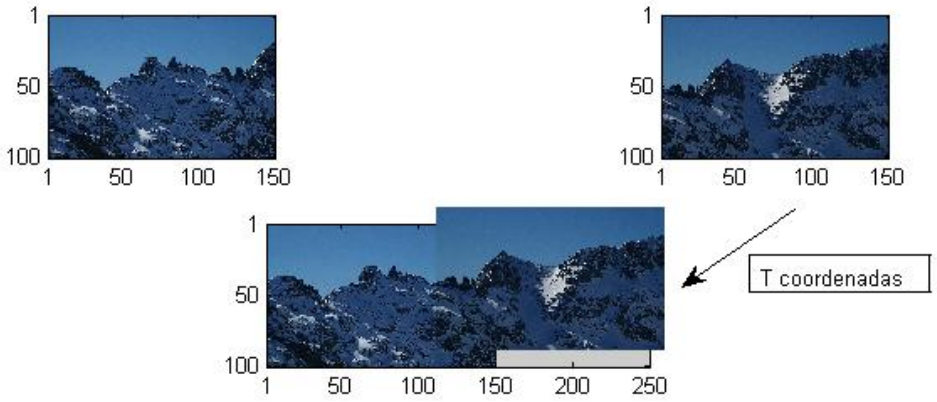

Vemos que la zona derecha de la primera enlaza con el borde izquierdo de la segunda. Podemos entonces buscar una serie de puntos (puntos de control) **comunes** a ambas imágenes y anotar las coordenadas que tienen en una y otra imagen. Tendríamos por ejemplo los siguientes datos:

Ptos de control	Posición en imagen 1		Posición en imagen 2	
	X1	Y1	X2	Y2
1º	1181	358	109	478
2º	1494	184	422	315
3º	...	...	...	...

Al estar la imagen 2 a la derecha de la imagen 1, los puntos de control tienen valores altos (derecha) en las coordenadas X de la primera imagen y valores bajos (izquierda) en sus coordenadas X referidas a la imagen 2.

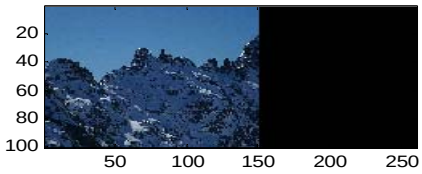
La idea para unir ambas imágenes es sencilla. Se trata de hallar una transformación T que transforme las coordenadas de la segunda imagen en las correspondientes coordenadas referidas a la imagen 1. En la tabla anterior se trataría de hallar la transformación T tal que T(109,478) fuese (1181, 358) o algo muy parecido. De esta forma podríamos barrer la imagen 2, transformar cada píxel (x2,y2) en el correspondiente (x1,y1) referido a la imagen 1 y adjuntar dicho píxel a la imagen 1, obteniendo una imagen ampliada.

En la fila superior de la figura adjunta tenemos las dos imágenes de partida, ambas de 100x150 píxeles y ambas referidas a sus propio orígenes:



La transformación T nos permite "trasladar" la imagen 2 a las coordenadas de la imagen 1. En la fila inferior, la imagen 1 (izquierda) mantiene su origen de coordenadas pero la imagen 2 ha ido a parar a las coordenadas donde le lleve la transformación T (lo que hace que solape con la imagen #1).

En la práctica trabajamos con la transformación inversa T^{-1} que pasa de coordenadas de la imagen 1 a la imagen 2. La razón es que la forma de proceder es la siguiente. Crearemos una imagen panorama extendida (por ejemplo 100x250 píxeles en el ejemplo anterior) y se trata de barrer dicha imagen e ir rellenándola con los datos de una u otra imagen según corresponda.
Si las coordenadas caen dentro del rango de la imagen 1 usaremos los datos de esa imagen, obteniéndose algo como la figura siguiente (con la imagen 1 a la izquierda):



Si nuestro píxel está fuera de la imagen 1, aplicaremos la transformación T^{-1} (desde img1 --> img2) para ver si es posible que el resultado entre en el rango (1:100 x 1:150) adecuado. Si ese es el caso usaremos la correspondiente información de la imagen 2 para rellenar el píxel.

Lo bueno es que ya sabemos resolver el problema de hallar una transformación que convierta (con el menor error cuadrático) una serie de coordenadas en otras. Es lo que resolvimos en la primera parte de la práctica. Allí buscábamos una trasformación entre coordenadas de píxeles {Xk,Yk} y coordenadas geográficas {Ek,Nk}. Ahora el primer grupo de datos son las coordenadas {X1,Y1} sobre la primera imagen y el segundo las coordenadas sobre la otra imagen {X2,Y2}.

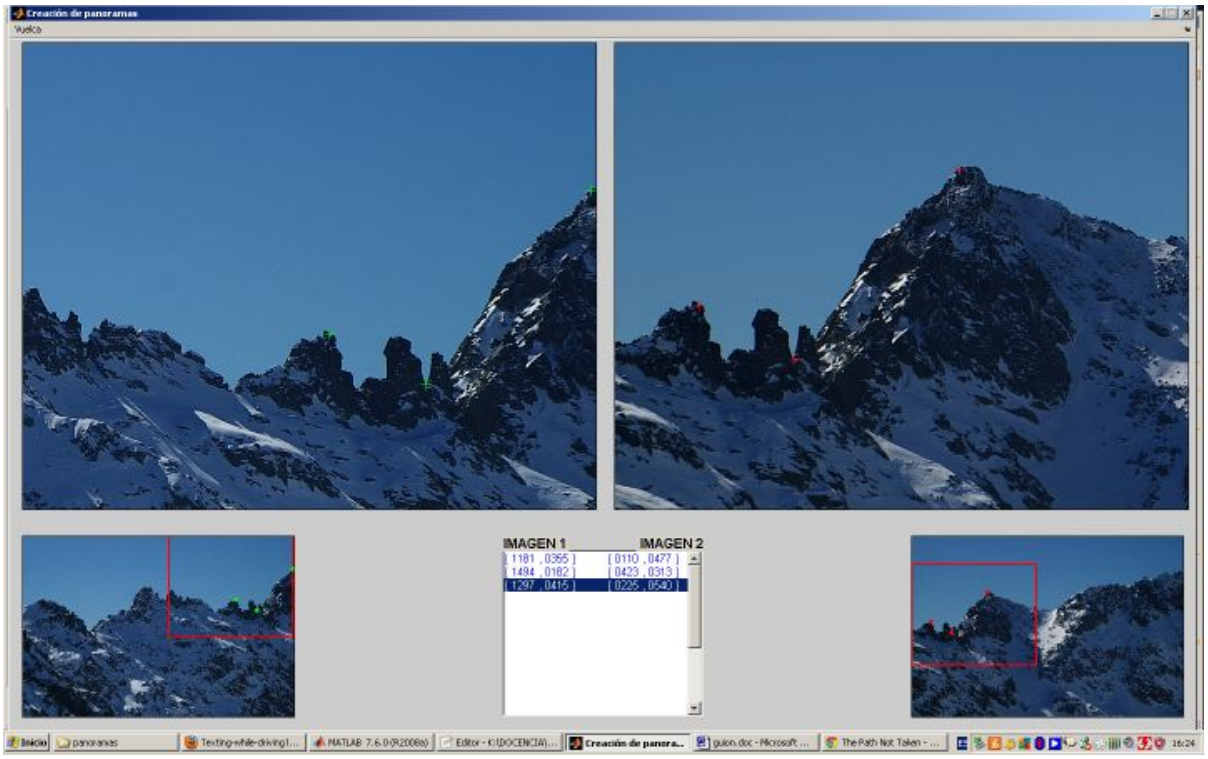
Como antes, se trata de hallar el mismo tipo de transformación afín tal que:

$$\begin{pmatrix} X \\ Y \end{pmatrix}_2 = M \cdot \begin{pmatrix} X \\ Y \end{pmatrix}_1 + D$$

sea un buen ajuste para el conjunto de las coordenadas de los puntos de control.

Los {X1,Y1} son las 2 primeras columnas de la tabla anterior (coordenadas en la imagen 1) y las {X2,Y2} las dos últimas (coordenadas en imagen 2). A partir de estos datos podemos usar la función ajuste.m para determinar la trasformación (matriz M y vector D) de coordenadas en img1 a coordenadas en img2.

Al igual que antes, para que sea rápido obtener las coordenadas de varios puntos de control para una pareja de imágenes dada, podéis usar la aplicación control.m suministrada. Ejecutando control('img1','img2') veréis algo como esto:



Abajo podéis ver sendas versiones reducidas (al 25%) de las dos imágenes (img1 a la izquierda, img2 a la derecha). Sobre ellas se superponen dos rectángulos rojos, que delimitan la zona de las imágenes que se muestra arriba (ampliadas al 200%). Arrastrando los rectángulos nos movemos por distintas zonas de la imagen. Se trata de buscar las zonas comunes y escoger puntos distintivos que puedan servir como puntos de control. Una vez escogidos pinchar alternativamente en una y otra ventana sobre el punto seleccionado. Las coordenadas de los puntos de control elegidos se muestran en la lista central tanto para la primera imagen (izquierda) como para la segunda (derecha).

Una vez seleccionados los puntos, con el Menu (Datos), los datos de la tabla (etiquetados como vectores columna X1,Y1,X2,Y2) pueden guardarse en el fichero elegido. Haciendo >> load fich podemos tener acceso a los datos X1, Y1 (coordenadas en imagen izquierda) y X2, Y2 (coordenadas en imagen derecha) desde fuera de la aplicación.

Luego, usando ajuste(X1,Y1,X2,Y2) podemos obtener la matriz M y vector D de la transformación que convierte coordenadas de la 1ª imagen (izquierda) a la 2ª imagen (derecha).

En esta práctica os propongo unir hasta tres imágenes, img1, img2 e img3, colocadas de izquierda a derecha en el orden dado. La imagen img2 está en el centro y será la que usemos como base o **imagen ancla** del panorama. Para hacerlo más sencillo podéis unir solamente las imágenes img1 y img2, ignorando las partes del proceso que hacen referencia a img3.

Descripción completa del proceso para crear el panorama:

1. Correr la aplicación control('img1','img2') para obtener una **lista de unos 5 / 7 puntos de control** comunes a img1 e img2. Conviene repartirlos por toda la zona, de arriba abajo en la zona común de las imágenes.

A partir de esos datos obtener (con función ajuste) la **matriz M1 y el vector D1** de la transformación que convierte **coordenadas de la imagen 2 en coordenadas de la imagen 1**.

4.1. Volcado puntos de control [X1 Y1 X2 Y2]
Parámetros M1 y D1 de la **transformación img2 -> img1**

2. Repetir haciendo control('img2','img3') para hallar la **matriz M3 y vector D3** para pasar de **coordenadas de imagen 2 a coordenadas de imagen 3**.
Esta parte podéis no hacerla si sólo vais a hacer un panorama con 2 imágenes.

4.2. Volcado puntos de control [X1 Y1 X2 Y2].
Parámetros M3 y D3 de la **transformación img2 -> img3**

3. Componer el panorama a partir de las tres imágenes:

a) Cargar las tres imágenes, de tamaño 900x1400 (alto=900, ancho=1400):

```
im1 = imread('img1.jpg');  im2 = imread('img2.jpg'); im3 = imread('img3.jpg');
```

b) Crear una imagen nueva más grande de tamaño (1000x3600) donde colocar el panorama resultante

```
pano = uint8(zeros(1000,3600,3);
image(pano);
```

c) Definir los parámetros DX = 1150 y DY=50, que indican en que posición del panorama resultante vamos a colocar nuestra imagen "ancla" (im2 en este caso). Los valores DX, DY elegidos colocan a la imagen im2 aproximadamente en el centro del panorama resultante:

d) Lo único que queda es rellenar la imagen pano (que ahora está vacía). Para ello haremos un bucle en k (desde 1 a 1000) y otro en j (desde 1 a 3600). En cada iteración de este bucle tratamos de rellenar el píxel (k,j) de la imagen pano. Para ello escribiremos el siguiente código dentro del bucle:

- Hacer x2 = j - DX; y2 = k - DY, para obtener las coordenadas (x2,y2) referidas a la imagen ancla (en nuestro caso la 2ª imagen, im2).

- Si (y2>=1) && (y2<=alto) && (x2>=1) && (x2<=ancho) es que estamos dentro de la imagen ancla (im2). En ese caso rellenamos el píxel (k,j) de pano con los contenidos del píxel (y2,x2) de im2 y pasamos al siguiente paso del bucle. Para ello podemos usar el comando continue que hace que el código restante en el bucle ya no se ejecute.

- Si no hemos podido usar la imagen 2, aplicamos la T (2->1) obtenida en el apartado 1) al píxel (x2,y2) para pasar de coordenadas de la imagen 2 a coordenadas en la imagen 1:

$$\begin{pmatrix} x1 \\ y1 \end{pmatrix} = M_1 \begin{pmatrix} x2 \\ y2 \end{pmatrix} + D_1,$$

  redondeando (round) los valores (x1,y1) de las nuevas coordenadas.

- De nuevo, si (y1>=1) && (y1<=alto) && (x1>=1) && (x1<=ancho) es que estamos dentro de im1 y podremos rellenar el pixel (k,j) de pano con los contenidos del píxel (y1,x1) de im1. Tras hacerlo, saltar al siguiente paso del bucle (de nuevo, usando continue). Si sólo estáis uniendo 2 imágenes podéis parar aquí.

- Finalmente, si no hemos conseguido llenar pano(k,j) con los datos de im2 ni de im1, probaremos a ver si el píxel (k,j) puede venir de im3:

$$\begin{pmatrix} x3 \\ y3 \end{pmatrix} = M_3 \begin{pmatrix} x2 \\ y2 \end{pmatrix} + D_3$$

  y de nuevo redondeamos las coordenadas (x3,y3) obtenidas. Verificar si (x3,y3) cae dentro del rango correcto y si es así rellenar el píxel (k,j) de pano con el correspondiente píxel de im3.

Recordar que por cada píxel tenemos en realidad 3 valores al ser imágenes en color. Por lo tanto, al copiar un píxel de im1, im2 o im3 al panorama final, hay que copiar las tres componentes (RGB) haciendo algo así como

```
pano(200,300, : ) = im(407,531,:)
```

que copia simultáneamente los 3 valores (RGB) de una imagen a otra.

4.3) Adjuntad todo el código usado para para formar el panorama.
4.4) Adjuntar imagen final (pano) obtenida. Indicad problemas observados.