

CAPÍTULO 5. PROCESAMIENTO

En este punto, y tras haber finalizado el proceso de adquisición de datos, se tiene un **fichero** que contiene los datos con los que se pretende generar un modelo digital del terreno (MDT). Estos datos, como se ha explicado anteriormente, pertenecen tanto al dispositivo LD-MRS como al MTi-G. Se hace referencia al procesamiento de los datos usando los términos *procesamiento* o *corrección* indistintamente, debido a que el procesamiento lo que pretende es corregir las coordenadas adquiridas dado que la posición y orientación inerciales de la aeronave están implícitas en ellas, y hay que realizar transformaciones con el fin de obtener las coordenadas reales de los puntos de la superficie expresados en un sistema de coordenadas pseudo-inercial.

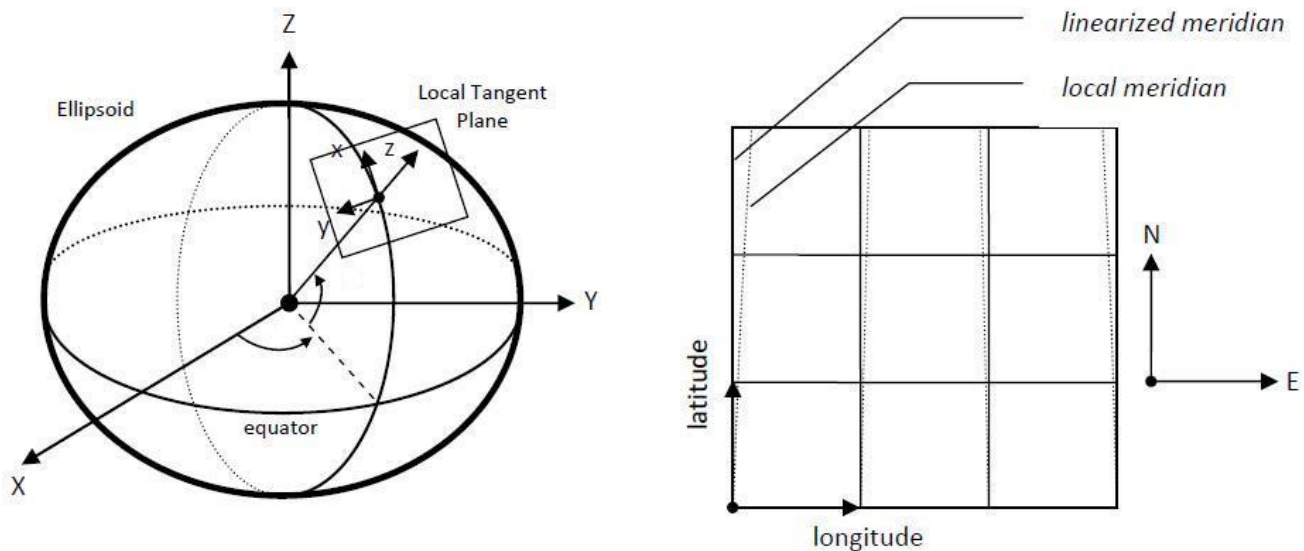


Figura 45. Sistema de coordenadas de navegación y linealización de la latitud y la longitud (13)

En el sistema basado en tecnología LIDAR que se ha diseñado, ambos dispositivos utilizan **sistemas de referencia diferentes**, propios e intrínsecos a su estructura, a los que se les denomina sistemas de coordenadas

body b-frame. Para representar el MDT se necesita un sistema de coordenadas único y adecuado para tal fin. No tiene sentido el transformar todos los datos del LD-MRS para expresarlos en el sistema de coordenadas *body fixed* del MTi-G, cuando se sabe que ese sistema de coordenadas no se puede aproximar como un sistema inercial, y por lo tanto no es adecuado para la representación del MDT.

Para representar el MDT el sistema de coordenadas más adecuado es el **ECEF (*Earth Centered Earth Fixed*) o *e-frame***, el cual no es puramente inercial, pero la aceleración a la que está sometido es despreciable con respecto a las aceleraciones que experimenta la aeronave y el sistema en sí.

En la práctica resulta complicado trasladar todos los datos adquiridos a este sistema de referencia, por lo que se emplea una **simplificación** que viene a proporcionar resultados muy parecidos, la cual se explica a continuación. El sistema de referencia con respecto al que el MTi-G expresa su orientación, se conoce como **sistema de coordenadas de navegación o *n-frame***. En el caso del MTi-G se utiliza la configuración **NWU (*North, West, Up*)**. Este sistema de coordenadas de navegación se define sobre un plano tangente (*Local Tangent Plane*) para cada punto de la superficie terrestre (Figura 45). Si se elige uno de esos planos como referencia, los puntos del mismo cercanos al origen (punto de tangencia) serán posiciones que tienen unas diferencias mínimas con respecto a las posiciones propias de la superficie terrestre a las que corresponderían esos puntos (considerando el sistema ECEF como el que representa las posiciones reales con las que se busca comparar los datos). Se puede decir entonces que esos planos resultan de la **linearización** de la superficie terrestre. Ésta es la simplificación que se pretende realizar mediante el uso del plano tangente local como sistema de referencia.

El realizar desplazamientos de pequeña magnitud con respecto al punto tangente del plano de tangencia asegura que el error que se comete es mínimo. Este mecanismo no es más que una **linearización de las coordenadas elipsoidales Latitud, Longitud y Altitud**. Este procedimiento de linearización de los datos del MTi-G viene recomendado en la documentación del propio dispositivo, dadas las limitaciones propias de los sensores inerciales MEMS del MTi-G. Las siguientes expresiones representan matemáticamente el mecanismo de linearización comentado. Se recuerda que al utilizar el sistema de coordenadas *n-frame* con configuración NWU, los ejes del mismo apuntan al norte (N), al oeste (W) y el último coincide con la normal saliente del plano tangente (U).

$$\text{Par latitud – longitud: } (\theta_{ref}, \phi_{ref}) \quad [12]$$

$$\Delta\theta = \theta - \theta_{ref} \quad [13]$$

$$\Delta\phi = \phi - \phi_{ref} \quad [14]$$

$$W = -R \cdot \Delta\phi \cdot \cos \theta \quad [15]$$

$$N = R \cdot \Delta\theta \quad [16]$$

Dónde:

- θ representa la latitud de cualquier punto de un *scan* del LD-MRS.
- ϕ representa la longitud de cualquier punto de un *scan* del LD-MRS.
- $(\theta_{ref}, \phi_{ref})$ son las coordenadas del punto de tangencia del plano tangente local.
- W representa la coordenada este, expresada en el sistema de coordenadas asociado al plano tangente local, es decir, el sistema de coordenadas de navegación o *n-frame*.
- N representa la coordenada norte, expresada en el sistema de coordenadas asociado al plano tangente local, es decir, el sistema de coordenadas de navegación o *n-frame*.
- R es el radio de la Tierra.

Para la coordenada correspondiente al eje U no se tiene que realizar ninguna simplificación ya que ésta coincide con la coordenada de la altitud.

5.1. Procedimiento de procesamiento

Se describe a continuación el procedimiento que se sigue en el código para realizar el procesamiento de los datos, es decir, las transformaciones que se les aplican a las coordenadas para expresarlas en los distintos sistemas de coordenadas que se han descrito anteriormente. El fin del procedimiento es llegar al sistema de coordenadas de representación, el que se utilizará para representar el MDT.

Las transformaciones que se van a realizar durante todo el proceso se aplican sobre **las coordenadas** de los puntos que componen un *scan* del LD-MRS. Los puntos de un *scan* están expresados en el sistema de coordenadas ***b-frame* del LD-MRS** y la primera transformación consiste en expresar los mismos en el **sistema de coordenadas *b-frame* del MTi-G**. Ambos dispositivos tienen distintos sistemas de coordenadas *b-frame* y el montaje del sistema en la aeronave y la distancia entre los dispositivos también influye en esta transformación, tal y como se puede ver en la Figura 46.

Para expresar las coordenadas de los puntos de un *scan* en el sistema de coordenadas *body b-frame* del MTi-G se necesita realizar una **rotación de 90° alrededor del eje 'y' sobre el sistema de coordenadas *b-frame* del LD-MRS**, de manera que los ejes de ambos sistemas de coordenadas queden apuntando en la misma dirección y sentido. Además, será necesaria una **traslación del origen de coordenadas** ya que ambos sistemas están centrados en posiciones distintas. En la Figura 46 puede verse la separación real de los orígenes de coordenadas que existe en el caso del montaje del sistema en el helicóptero CB5000.

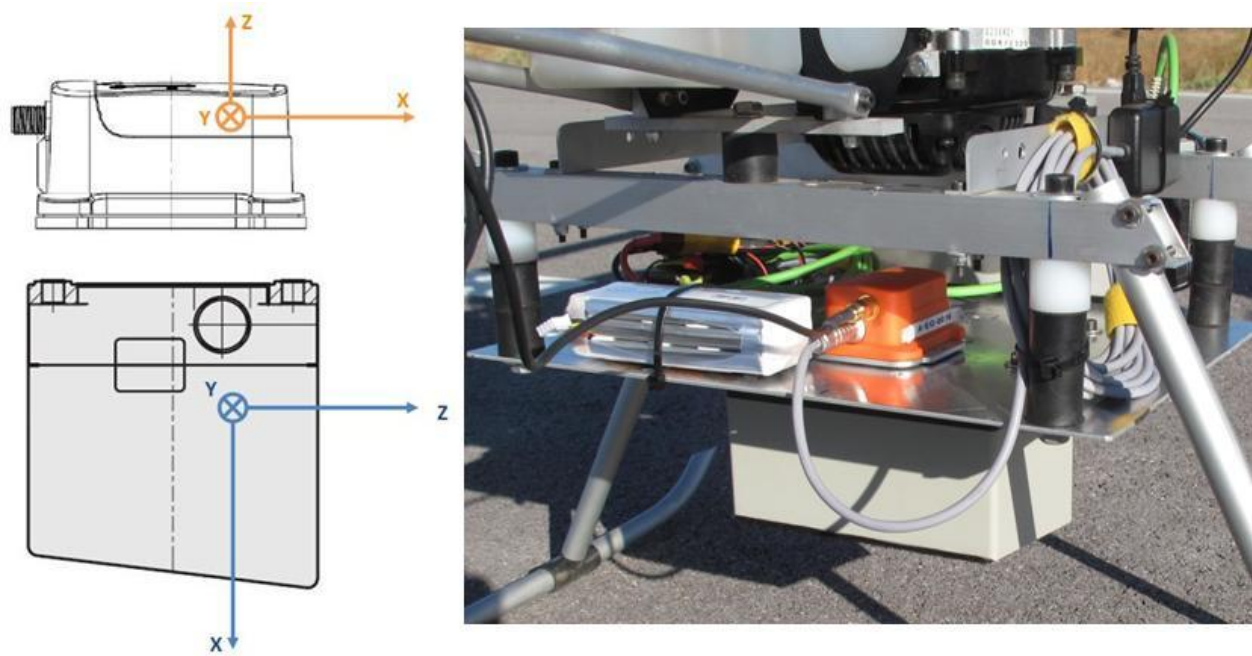


Figura 46. Configuración del montaje del sistema diseñado y sistemas de coordenadas *b-frame*

Tras esta primera transformación se tienen los puntos de un *scan* del LD-MRS expresados con respecto al sistema de coordenadas *b-frame* del MTi-G. Éste último tiene la misma posición y orientación que la aeronave y obviamente éstas varían dependiendo del instante en el que se realizó el barrido con el LD-MRS. Por ejemplo, los *scans* previos y posteriores a uno dado, corresponderán a instantes de tiempo distintos y por lo tanto estarán expresados en sistemas *b-frame* con orientaciones y orígenes de coordenadas distintos. La orientación de los sistemas de coordenadas *b-frame* del MTi-G se expresan con respecto a los sistemas de coordenadas de navegación o *n-frame* mediante los ángulos de Euler.

El fabricante del MTi-G recomienda utilizar una aproximación para trabajar con la posición y la orientación que proporciona el dispositivo. Dadas las características propias de los MEMS del MTi-G, éstos no pueden medir la rotación de la Tierra ni tampoco la tasa de transporte del dispositivo sobre la superficie terrestre cuando éste experimenta otras aceleraciones. Teniendo en cuenta esto, se recomienda trabajar con un sistema de coordenadas local, construido sobre una linearización de la superficie terrestre ya que facilita mucho el trabajo y no introduce errores significativos en las medidas de posición y orientación. Esta linearización corresponde a aproximar la superficie terrestre mediante un plano tangente local (*Local Tangent Plane* o *LTP*), siendo el sistema de coordenadas local sobre el que se trabaja un sistema *n-frame*. El punto de tangencia del LTP con la superficie terrestre (que a efectos prácticos es la posición de la aeronave) se toma como el origen del sistema de coordenadas *n-frame* y sus ejes apuntan según la configuración NWU (*North-West-Up*).

El sistema de navegación o *n-frame*, como ya se ha visto, puede definirse en cada punto cercano de la superficie terrestre. Por lo tanto se dispondrá de muchos sistemas de coordenadas *n-frame* distintos, uno por cada posición distinta de la aeronave durante un vuelo de adquisición de datos. De entre ellos, y como se verá más adelante, se escogerá uno como sistema de referencia para representar el MDT.

En base a lo anterior, el siguiente paso a dar es **la rotación del sistema de coordenadas *b-frame* para expresar las coordenadas en el sistema *n-frame* que tiene asociado**. Se tiene que realizar una rotación para cada *scan*, porque cada *scan* se adquirió en instantes distintos y por lo tanto tienen ángulos de Euler asociados y sistemas de coordenadas *b-frame* y *n-frame* también distintos (Figura 47).

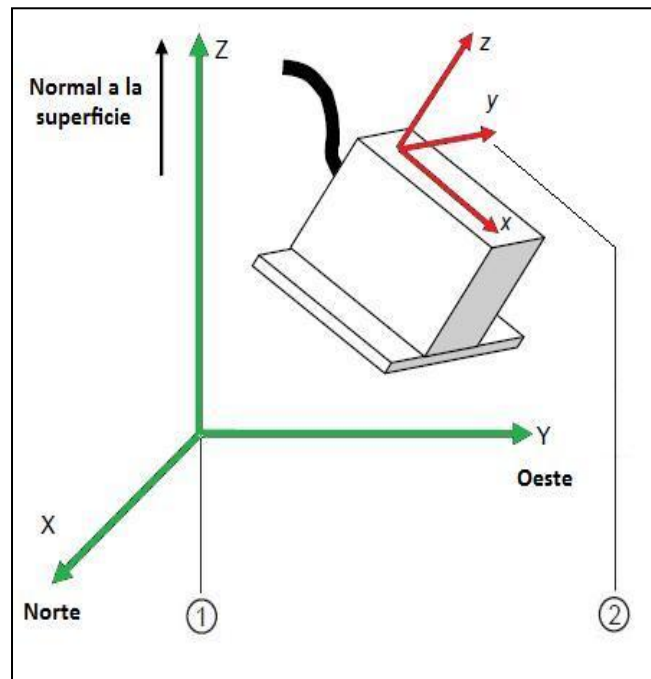


Figura 47. Orientación del Mti-G (13)

1. Sistema de coordenadas de navegación *n-frame*
2. Sistema de coordenadas Body *b-frame* del MTi-G

Estas transformaciones se realizan de manera matricial, por lo que el orden en el que se realizan las rotaciones es importante. Los ángulos de Euler, tal y como están definidos en la documentación del MTi-G, coinciden con los ángulos *Roll*, *Pitch* y *Yaw*, que son muy conocidos en navegación aérea. Primero se aplica la rotación correspondiente al ángulo **Roll (ϕ)**, luego al **Pitch (θ)** y por último al **Yaw (ψ)**. Esto se muestra en la Figura 48.

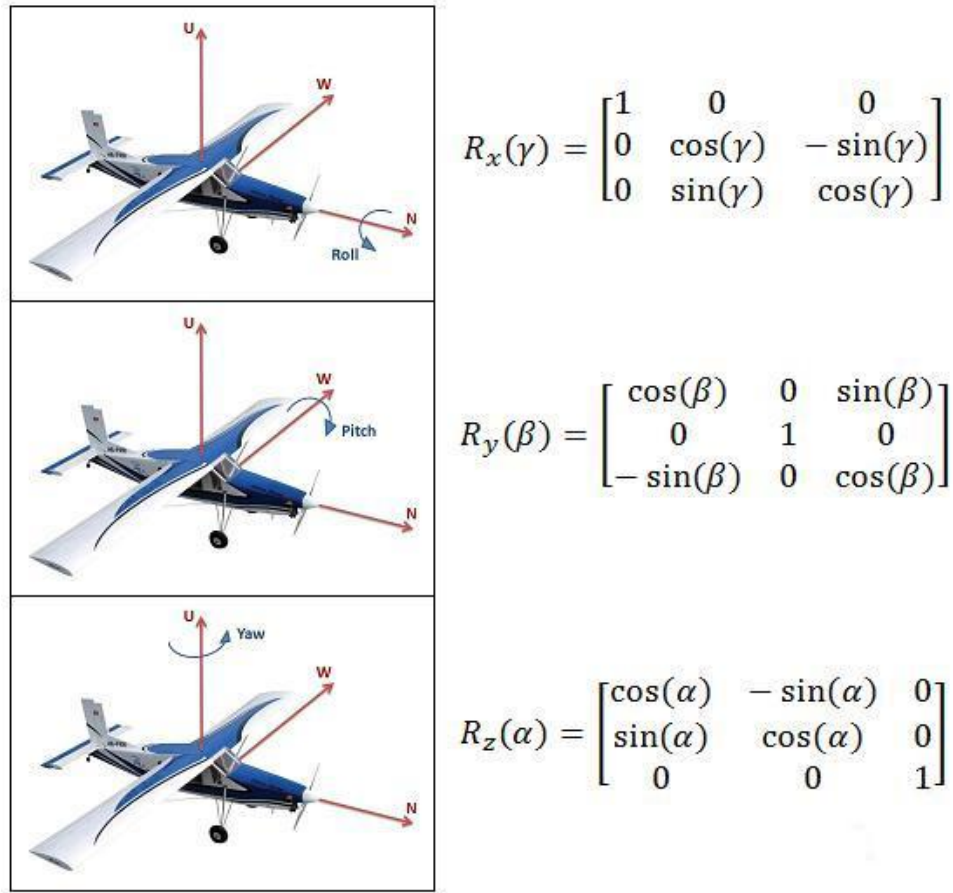


Figura 48. Matrices de rotación de los ángulos de Euler

Un sistema de coordenadas *n-frame* se puede definir en cada punto cercano de la superficie terrestre, por lo tanto, cada *scan* del LD-MRS, que está asociado a la posición que ocupaba la aeronave en el momento en el que fue adquirido, tendrá asociado un sistema de coordenadas *n-frame* distinto. La posición de la aeronave asociada a cada *scan*, en cada instante, será tomada como el origen de coordenadas de los sistemas *n-frame* correspondientes.

Al utilizar la aproximación antes comentada y trabajar con los sistemas de coordenadas *n-frame* para representar el MDT se introduce un error espacial, como el que se muestra en la Figura 49. Para minimizar este error el fabricante recomienda que el sistema de coordenadas *n-frame* de referencia, es decir, el que se va a escoger para representar el MDT, se escoja lo más cercano posible al resto de coordenadas con las que se está trabajando (las posiciones de la aeronave). La razón de esto es que si se trabaja con coordenadas cercanas a las del origen del sistema *n-frame* de referencia, el *Local Tangent Plane* coincide prácticamente con la superficie terrestre y el error cometido es mínimo. Esto significa que los sistemas de coordenadas *n-frame* asociados a las demás coordenadas que ocupa la aeronave durante un vuelo tendrán la misma orientación, salvo mínimas diferencias, que el sistema de coordenadas de referencia.

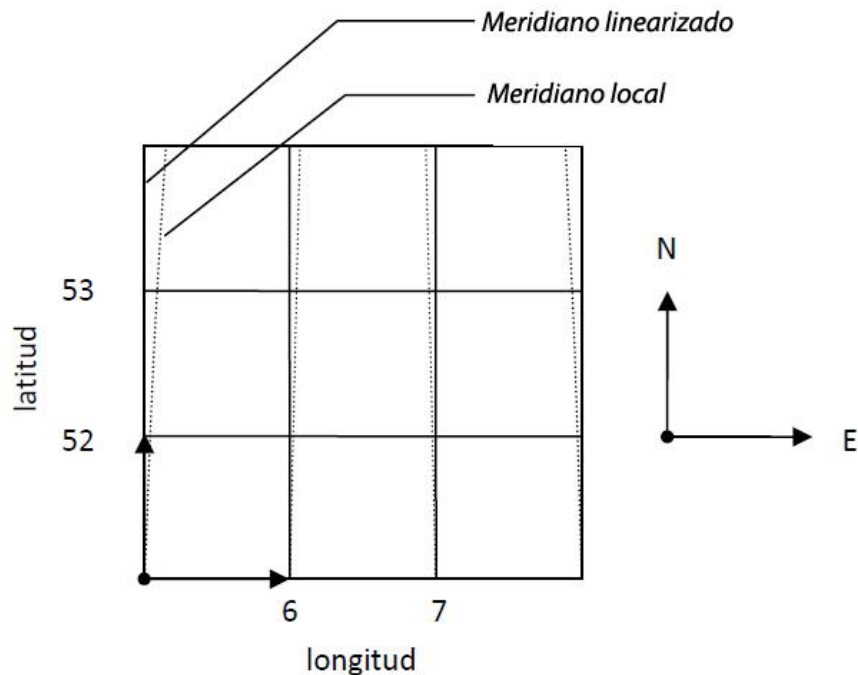


Figura 49. Linearización de la superficie terrestre y ejemplo del error que introduce (13)

Esto puede hacerse así debido a que en los vuelos realizados se han recorrido distancias pequeñas, por lo tanto, al realizar la aproximación, el error introducido es despreciable con respecto a la precisión que el MTi-G proporciona. Para tener una cierta idea del error cometido póngase como ejemplo que se trabaja con un sistema de referencia *n-frame* y los demás sistemas *n-frame* adquiridos en el vuelo distan, como máximo, 1000 m del primero. En esta situación, la diferencia entre las orientaciones del sistema de referencia y de otro colocado a 1000 m, teniendo en cuenta una sola dimensión (la rotación correspondiente a un solo eje) es del orden de magnitud de la centésima parte de un grado. Esto conlleva a un error del orden de magnitud de las unidades de metros entre la posición real a 1000 m del origen del sistema de referencia y la aproximada según la linearización. Teniendo en cuenta que la precisión máxima que proporciona el MTi-G en cuanto a la posición es de 2.5 m y que en el proyecto se ha trabajado con distancias menores que 1000 m, este error introducido por la linearización es despreciable.

Para aplicar la aproximación que se ha comentado, se toma uno de estos **sistemas de coordenadas de navegación *n-frame*** como **sistema de referencia** y sobre él se representa el MDT. Dado que en los vuelos realizados se han sobrevolado áreas pequeñas de la superficie terrestre y las distancias recorridas han sido inferiores a los 1000 m, se puede elegir como sistema *n-frame* de referencia cualquiera de los asociados a las coordenadas de la aeronave durante el vuelo. Como no hay preferencia por ninguno **se elige como referencia el primero de ellos**, el que tiene por origen a las primeras coordenadas elipsoidales válidas (Latitud, Longitud y Altitud) que proporciona el MTi-G. A este sistema de coordenadas se le denomina también en este documento **sistema de coordenadas de representación** ya que es el que se usará para representar el MDT.

Por último se requiere que los puntos de cada *scan* a los que se les han aplicado las anteriores transformaciones, y que en este punto del procedimiento de procesamiento están expresados en sistemas de coordenadas *n-frame* (asociados a otras coordenadas de latitud, longitud y altitud distintas a las del primer *scan*), sean expresados en el sistema de coordenadas de representación. Se necesita por tanto realizar **una nueva traslación para cada *scan***, trasladando los distintos sistemas de coordenadas *n-frame* al sistema de coordenadas de representación, debido a la diferencia de posición entre los orígenes de coordenadas de los sistemas de coordenadas *n-frame* con respecto al de representación.

5.2. Software de procesamiento

Una vez que se ha explicado el procedimiento que se sigue para el procesamiento de los datos, se procede a describir el código que lo implementa. Éste se estructura en **cinco ficheros**. En el principal se encuentra descrita la función ***main***, en otro la definición de las **clases** que se han utilizado para almacenar los datos y en otro la definición de varias **funciones específicas** que se utilizan en el programa. Los otros dos ficheros que faltan para sumar los cinco son ficheros del **tipo .h** necesarios para el compilador.

5.2.1. Main

La función *main* comienza definiendo, en forma de **macros**, las posibles **configuraciones de montaje** que se han utilizado en las pruebas de adquisición de datos y sus distancias correspondientes, es decir, las distancias entre los sistemas de coordenadas del LD-MRS y del MTi-G. En total se han utilizado tres configuraciones, una correspondiente a la aeronave PILATUS Porter, otra para el helicóptero CB5000 y otra para pruebas de laboratorio (sin vehículo aéreo). En la Tabla 15 aparecen las distancias a tener en cuenta en el procesamiento de los datos para cada caso.

	Laboratorio	PILATUS	CB500
MPOS_X	-0.0027	-0.023	0.013
MPOS_Y	-0.0221	0.0514	0.0299
MPOS_Z	-0.0535	-0.1365	-0.0565

Tabla 15. Distintas configuraciones de montaje

Los valores que se muestran aquí corresponden con la **distancia que separa el origen de coordenadas** del sistema de coordenadas *body b-frame* del **LD-MRS** y del **MTi-G**. MPOS_X corresponde a la distancia expresada en el eje X, MPOS_Y la correspondiente al eje Y y MPOS_Z la correspondiente al eje Z del MTi-G. Estas distancias son necesarias para llevar a cabo la traslación del sistema de coordenadas *b-frame* del LD-MRS

al *b-frame* del MTi-G, tras haber rotado primero el del LD-MRS para que la orientación de ambos sea la misma, tal y como se explicaba en el apartado relativo al procedimiento.

En las primeras líneas de código de la función *main*, aparecen ciertas **variables** que son usadas como **contadores** de distintos tipos de parámetros dentro del fichero de adquisición de datos. Éste fichero recoge los datos de entrada para el programa de procesamiento y procede de la anterior etapa de la adquisición de datos. Entre estos contadores se encuentran los que aparecen en la Tabla 16.

<i>Gen_count</i>	Contador general de caracteres. Almacena la posición del carácter que se está leyendo en cada momento en el fichero de entrada.
<i>numScans</i>	Contador del número de <i>scans</i> total del fichero. Se utiliza para saber si se perdió algún <i>scan</i> por problemas de sincronización durante la adquisición de los datos.
<i>invalid_scans</i>	Contador del número de <i>scans</i> que tienen valores de tiempo UTC incorrectos.
<i>empty_scans</i>	Contador del número de <i>scans</i> que no tienen ningún punto válido (todos corresponden a partículas suspendidas, suciedad, etc.).
<i>bad_GPScount</i>	Contador del número de <i>scans</i> que tienen errores en los valores proporcionados por el GPS.

Tabla 16. Contadores para el análisis del fichero de entrada

La primera acción que realiza el código es abrir el fichero de entrada y, a la vez, crear una **carpeta para almacenar los ficheros de salida** que se generarán. En esta carpeta de nombre ***Data_corrected_[fecha_adquisición]*** se almacenarán ficheros con los datos procesados y expresados en el sistema de coordenadas escogido para representar el MDT. Se generará un fichero distinto por cada *scan* procesado y estarán numerados según la numeración que los *scans* recibieron durante la adquisición (Figura 50). Aparte se genera un fichero denominado ***debug_[fecha_adquisición]*** que recoge información de interés para la depuración del código (número de *scans* sin puntos válidos, desaparecidos, etc.); también se crea otro fichero denominado ***lla_eul_vel*** que almacena únicamente los datos inerciales y de posición de cada uno de los *scans*; el último de los ficheros se denomina ***scans*** y guarda la lista de los *scans* que tienen puntos válidos.

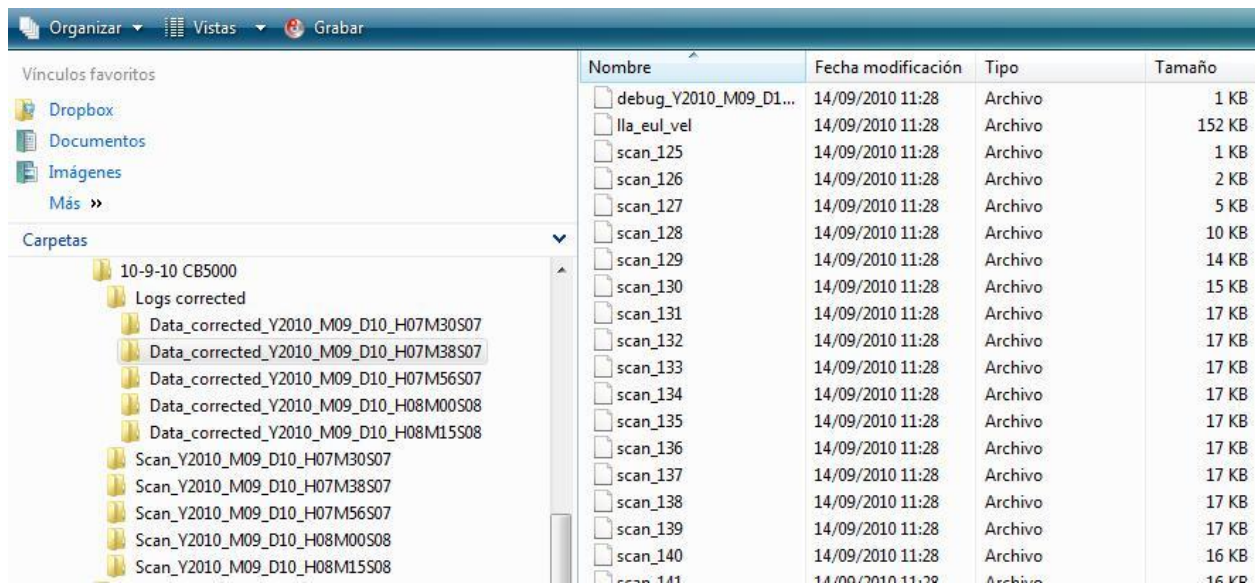


Figura 50. Estructura de carpetas creada por el código de corrección de datos

La Figura 50 representa los ficheros de salida generados tras una ejecución del programa de procesamiento. En la figura se ve que los ficheros que guardan los datos procesados de cada *scan* empiezan por el *scan_125*, lo que significa que los *scans* anteriores no fueron procesados debido a que no tenían puntos válidos o porque los datos del GPS no eran fiables. Más adelante se pueden ver imágenes del contenido de estos ficheros de salida.

Antes de generar todos estos ficheros el código del programa analiza el fichero de entrada para extraer de él los datos necesarios para el procesamiento. Para ello se utilizan funciones escritas específicamente para este propósito, las cuales están recogidas en el fichero de código fuente *functions.cpp*.

Con respecto al análisis del fichero de entrada, en primer lugar se realiza el recuento del **número de scans que almacena el fichero de entrada** (de adquisición). La función que desempeña esta tarea y que es llamada desde la función *main* recibe el nombre de *get_numScans()*. Ésta devuelve el número de *scans* en el contador *numScans*.

En este punto del código se necesita una **estructura de datos** en la que **almacenar toda la información** necesaria que se va a extraer del fichero de entrada. Los datos con los que se trabaja tienen una estructura compleja y por ello se han definido un conjunto de clases para almacenarlos y facilitar las operaciones que se van a realizar sobre ellos. Esta estructura de clases se describe con mayor detalle en el apartado *clases.cpp*, que se encuentra unas páginas más adelante.

Una vez conocido el número total de *scans* que se adquirieron se procede a **reservar memoria** para los objetos de las clases que van a almacenar la información que contienen estos *scans*. Al puntero que apunta a esta estructura de datos se le denomina *scan_vector*. En pocas palabras, *scan_vector* apunta al principio de

una **tabla** en la que en **cada posición** hay un **objeto** del **tipo scan**, y es en sus **atributos** dónde se almacena toda la **información** extraída del fichero de entrada, y sus **métodos** son los que realizan las **operaciones** de rotación y traslaciones necesarias sobre los atributos.

Seguidamente se genera un objeto de la clase **mountinPos**, donde se almacena la información necesaria para realizar la primera de las transformaciones de los sistemas de coordenadas. En este objeto se define la rotación necesaria a realizar entre los sistemas de coordenadas *b-frame* del LD-MRS y del MTi-G y la consiguiente traslación entre sus orígenes de coordenadas.

5.2.1.1. Bucle principal

En este punto del código se llega al **bucle principal** del programa, el cual recorre el fichero de entrada **scan por scan** extrayendo los datos de cada uno, almacenándolos en la estructura de datos y realizando las consecuentes rotaciones y traslaciones. Las coordenadas resultantes que se obtienen tras aplicar una transformación a unas anteriores se guardan en el mismo lugar de la estructura de datos en el que estaban las coordenadas no transformadas, esto es así porque las transformaciones siguientes habrá que realizarlas sobre las que ya se han realizado, y por lo tanto los datos originales no son necesarios para nada más. De esta manera también se consigue no consumir demasiada memoria, ya que una vez que se reserva la memoria necesaria inicial no se vuelve a reservar más.

La primera acción que se realiza al entrar en el bucle principal es **analizar el contenido del primer scan** y **extraer la información** que almacena (coordenadas de la superficie terrestre con respecto al sistema *b-frame* del LD-MRS, coordenadas de la posición de la aeronave, su orientación, velocidad, y otros), esta información se guarda en la estructura de datos de forma ordenada. De este primer *scan* se obtienen las primeras coordenadas **latitud, longitud y altitud** válidas y se almacenan en un objeto definido para tal efecto denominado **lla_ref**. Éstas serán el origen del sistema de coordenadas *n-frame* utilizado para la representación del MDT, y a partir del cual se realiza la linearización de la superficie terrestre descrita anteriormente, utilizando el plano tangente local (LTP) del propio sistema de coordenadas *n-frame*. Éstas primeras coordenadas GPS que se almacenan son a la vez el **punto de tangencia** del plano tangente local con la superficie del modelo terrestre.

Una vez se tienen las coordenadas del sistema de referencia para la representación del MDT se procede a extraer la información que contiene el *scan*. La extracción de datos de los *scans* del fichero de entrada y su consiguiente copia en la estructura de datos del programa se realiza con la función **extract_scan()**, la cual se ha programado específicamente para tal tarea y está definida en el fichero **functions.cpp**. En cada iteración del bucle principal se extrae la información correspondiente a un *scan*.

Una vez extraídos los datos del *scan* que se está procesando, se utilizan los **métodos de los objetos tipo *scan*** en los que se han almacenado. Estos métodos o funciones propias del objeto realizan las transformaciones que se explicaron en el apartado que describe el procedimiento de procesamiento de los datos. En primer lugar **se corrigen**, con una rotación y una traslación, las **diferencias debidas al montaje** de los dispositivos en la aeronave, es decir, de los sistemas de coordenadas *body b-frame* del LD-MRS y del MTi-G, trasladando todos los datos al sistema de coordenadas *body fixed del MTi-G* (Figura 51). Esto se realiza mediante los métodos ***apply_rotation()*** y ***apply_traslation()***.

Seguidamente **se corrige la orientación** de la aeronave con la que se adquirieron los *scans* de la superficie. La orientación de la aeronave viene expresada en todo momento por los valores de los ángulos de Euler que el MTi-G proporciona. Cada scan tiene asociados los valores correspondientes de estos ángulos en el instante en el que fue adquirido y que representan la orientación del sistema de coordenadas *b-frame* del MTi-G y de su correspondiente sistema *n-frame* con la configuración NWU (Figura 52). Esto se realiza mediante el método ***correct_euler()***, propio del objeto tipo *scan*.

Posteriormente se calculan las diferencias entre el origen del sistema de coordenadas que se utilizará para representar el MDT (el sistema de coordenadas correspondiente al primer *scan*, o *n-frame* t_0) y el correspondiente al *scan* que se está procesando en el bucle en cada momento (*n-frame* t_n). Una vez conocidas esas diferencias se realiza la **traslación del sistema de coordenadas de navegación local al sistema de coordenadas de representación del MDT**, expresándose todas las coordenadas con respecto a éste último (Figura 52).

Por último, y una vez que se han realizado todas las traslaciones y rotaciones necesarias sobre las coordenadas, se **escribe en el fichero** de salida correspondiente al *scan* actual con el que se esté trabajando, los **datos procesados**.

Con esto acaba una iteración completa del bucle principal. En la siguiente iteración se pasa a procesar los datos del siguiente *scan* que aparece en el fichero de adquisición de datos, el cual se almacenará en la siguiente posición en la tabla de la estructura de datos. En cada una de estas iteraciones se escriben también en el **fichero *lla_eul_vel*** los datos inerciales y de posición de cada *scan*. De la misma forma, si el *scan* es válido y no se ha descartado se escribe su numeración en el **fichero *scans***.

Una vez fuera del bucle principal, se realiza un **procesado de los contadores** para poder escribir los resultados correspondientes en el fichero ***debug_[fecha_adquisición]***.

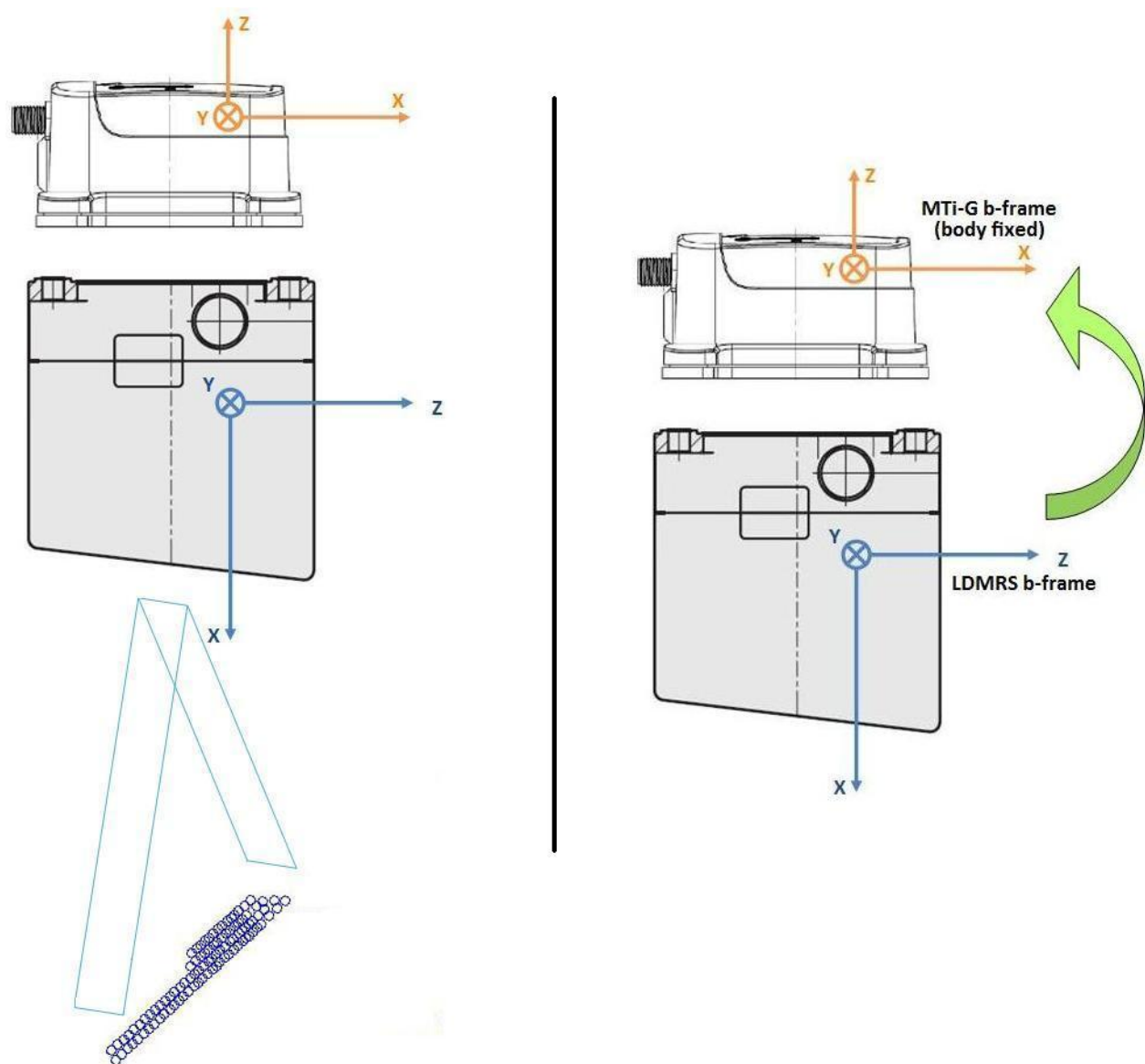


Figura 51. Diferencias entre los sistemas de coordenadas de los dispositivos (izquierda) y primera transformación de las coordenadas al sistema de coordenadas *b-frame* del MTi-G (derecha)

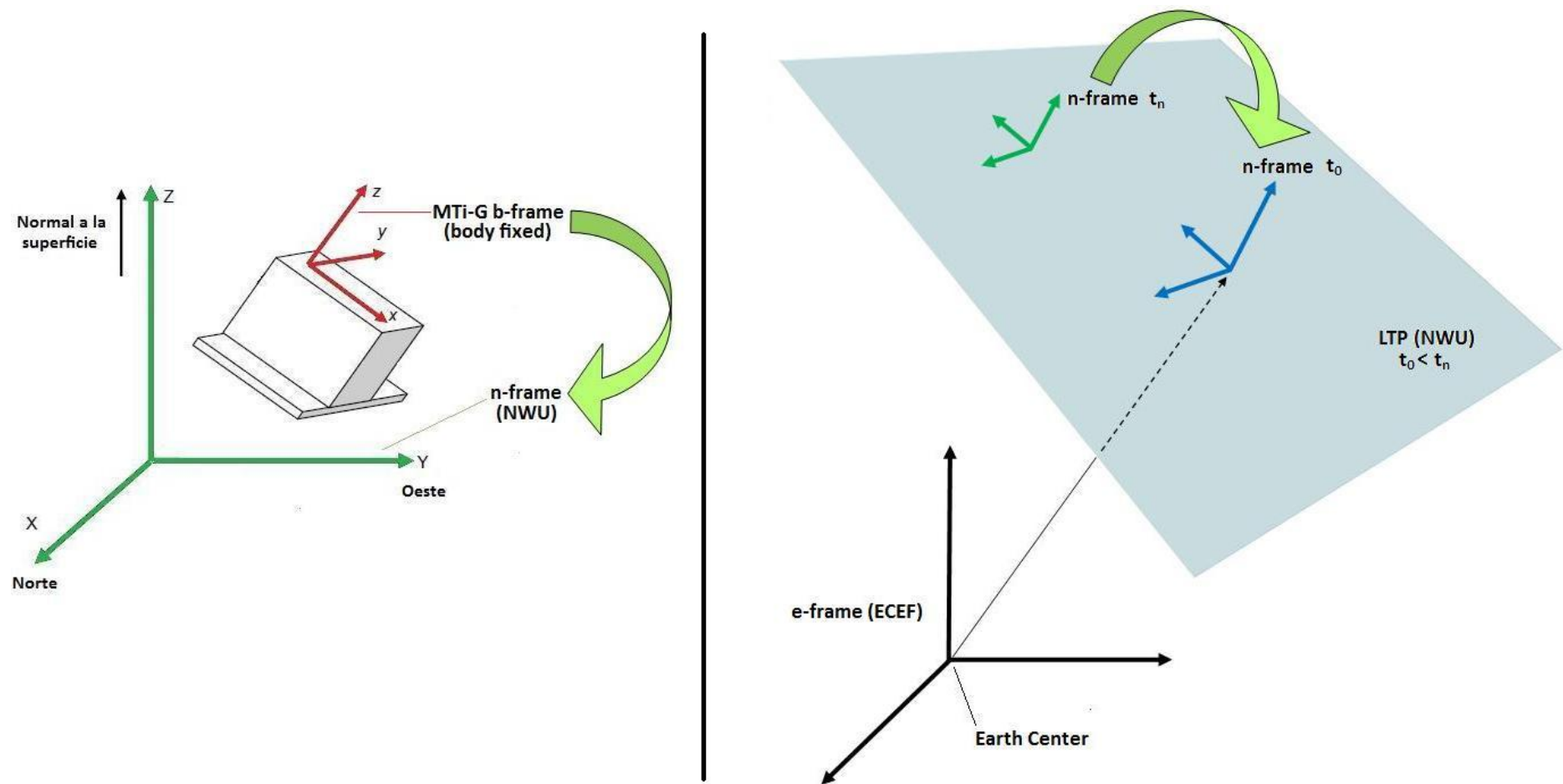


Figura 52. Segunda transformación de las coordenadas o corrección de la orientación con respecto al sistema de coordenadas *n-frame* (izquierda) y tercera transformación o traslación del sistema de coordenadas de navegación *n-frame t_n* (en cada instante) al sistema de coordenadas de representación *n-frame t_0* o inicial (derecha)

5.2.2. Functions.cpp

Este fichero contiene el código de tres funciones las cuales han sido programadas para analizar y extraer la información del fichero de adquisición de datos. Sus características se describen en la Tabla 17.

Función	Descripción
<i>compare_line()</i>	Busca una cadena caracteres determinada en el fichero de entrada, devolviendo, si la encuentra, el puntero al final de la cadena leída, justo delante del próximo valor a leer. Devuelve el valor 0 cuando encuentra la cadena buscada, -1 si encuentra el carácter terminador de fichero sin encontrarla o -2 si se produce un error mientras se leía el fichero.
<i>get_numScans()</i>	Lee el fichero de adquisición de datos y devuelve en una variable tipo referencia el número de <i>scans</i> que contiene. Devuelve -1 cuando encuentra el carácter terminador de fichero o -2 si se produce un error mientras se leía el fichero.
<i>extract_scan()</i>	Extrae toda la información que se necesita de un <i>scan</i> , tanto las coordenadas de la superficie del terreno como la posición de la aeronave, orientación y demás datos. Al extraer la información comprueba que las variables <i>status</i> y <i>valid</i> tengan valores válidos (lo que significa que el filtro de Kalman no divergía durante la adquisición del <i>scan</i> y que el GPS recibía una señal actualizada); en el caso de tener valores inválidos no extrae la información del <i>scan</i> y devuelve 0 para actualizar los contadores correspondientes de la función <i>main</i> . Si se extrajo la información sin problemas devuelve 1. La información que se extrae se almacena en la estructura de datos (o clases) del programa.

Tabla 17. Funciones definidas en el fichero functions.cpp

5.2.3. Clases.cpp

En este fichero se recoge la **definición de las clases** que se han creado **para almacenar la información** que contiene el fichero de **adquisición de datos** y, al mismo tiempo, facilitar las operaciones que se han de realizar sobre estos datos (rotaciones y traslaciones de sistemas de coordenadas).

En la Figura 53 se puede ver la lista de las clases creadas para el programa de procesamiento de los datos. Ésta figura ha sido generada con la aplicación conocida como **doxygen**. Ésta es un **generador de documentación** para numerosos lenguajes de programación, entre ellos **C++**, **C**, **Java**, **Objective-C**, **Python**, **IDL** (versiones *Corba* y *Microsoft*) y en cierta medida para **PHP**, **C#** y **D**. Este programa genera información en HTML, de dónde han sido extraídas estas imágenes, a partir del código fuente del programa.

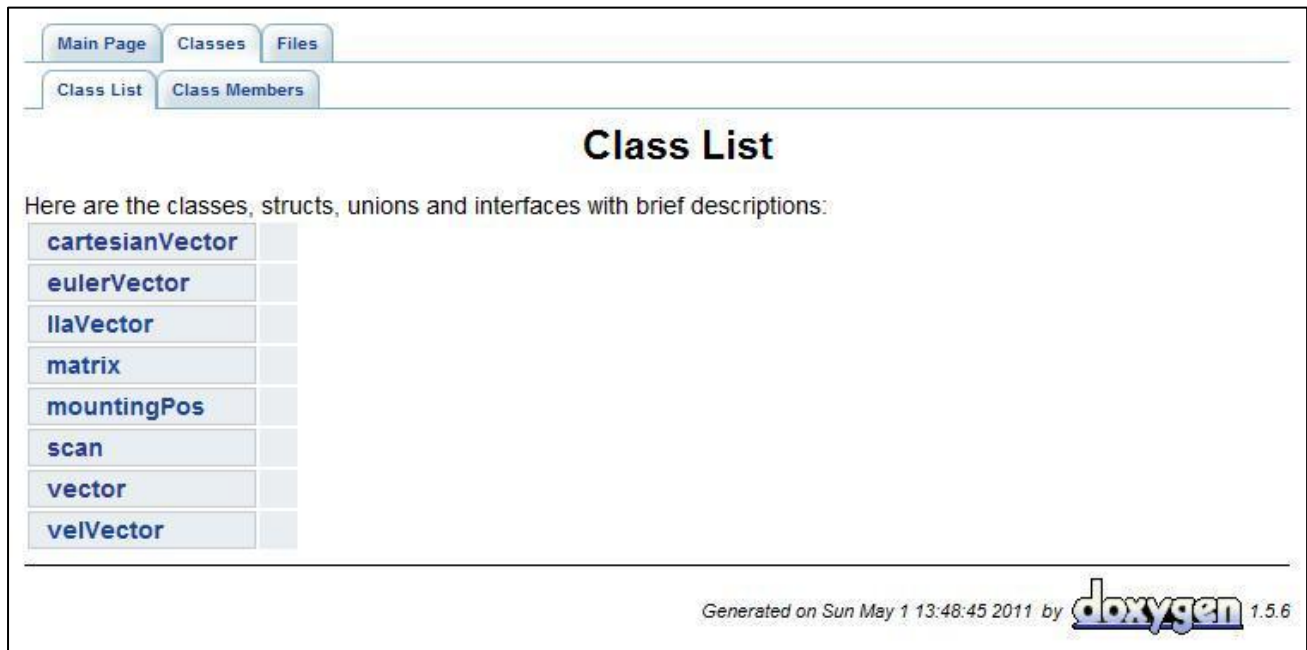


Figura 53. Clases del programa de corrección de datos (fichero clases.cpp)

Se puede decir que la clase de más importancia es la denominada **scan**. En ella se almacena toda la información que se extrae del fichero de adquisición de datos, pero para ello necesita apoyarse en todas las demás clases que aparecen en la Figura 53. Las clases **cartesianVector**, **eulerVector**, **llaVector** y **velVector** almacenan los datos fundamentales en forma de vector, como su propio nombre indica, y se describen en la Tabla 18.

Clase	Datos que almacena
cartesianVector	Coordenadas cartesianas, correspondientes a puntos de la superficie terrestre (expresados en el sistema de coordenadas <i>b-frame</i> del LD-MRS), y las características asociadas a cada punto.
eulerVector	Ángulos de Euler, expresan la orientación de la aeronave con respecto al sistema de coordenadas de navegación <i>n-frame</i> .
llaVector	Coordenadas elipsoidales (Latitud, Longitud y Altitud).
velVector	Velocidad de la aeronave en cada una de las componentes del sistema de coordenadas <i>body b-frame</i> del MTi-G.

Tabla 18. Clases y tipos de datos que almacenan

La clase **mountingPosition** se define principalmente para realizar más fácilmente las **transformaciones** correspondientes a la **posición de montaje** de los dispositivos LD-MRS y MTi-G en la aeronave. Esta clase se apoya en las clases **matrix** para realizar las operaciones de rotación, y **vector** para realizar las operaciones de traslación.

La clase **matrix** también sirve de apoyo a la clase **scan** para realizar el resto de **operaciones de rotación** necesarias, como la **corrección de la orientación** con los ángulos de Euler.

Main Page

Classes

Files

Class List

Class Members

scan Class Reference

```
#include <classes.h>
```

Collaboration diagram for scan: [\[legend\]](#)

[List of all members.](#)

Public Member Functions

	<code>scan ()</code>
long	<code>get_scanNum ()</code>
int	<code>get_numPoints ()</code>
int	<code>get_valid ()</code>
cartesianVector **	<code>get_cartesian ()</code>
eulerVector	<code>get_euler ()</code>
llaVector	<code>get_lls ()</code>
velVector	<code>get_vel ()</code>
scan *	<code>get_scan ()</code>
void	<code>set_scanNum (int &num)</code>
void	<code>set_numPoints (int &num)</code>
void	<code>set_valid (int num)</code>
void	<code>set_euler (double &r, double &p, double &y)</code>
void	<code>set_euler (double *p)</code>
void	<code>set_cartesian_i (int &i, double *p, float ew, int &f, int ch, int subch)</code>
void	<code>set_lls (double &la, double &lo, double &al)</code>
void	<code>set_lls (double *p)</code>
void	<code>set_vel (double &v_x, double &v_y, double &v_z)</code>
void	<code>set_vel (double *v)</code>
void	<code>apply_rotation (mountingPos &m_pos)</code>
void	<code>apply_traslation (mountingPos &m_pos)</code>
void	<code>apply_traslation (double ox, double oy, double oz)</code>
void	<code>correct_euler ()</code>
void	<code>print_scan (std::ofstream &file)</code>
void	<code>delete_scan ()</code>

Private Attributes

long	<code>scan_num</code>
int	<code>num_points</code>
int	<code>valid</code>
cartesianVector **	<code>cartesian_pp</code>
eulerVector	<code>euler</code>
llaVector	<code>lls</code>
velVector	<code>vel</code>

Figura 54. Atributos y métodos de la clase **scan**

Tal y como se ve en la Figura 54, la clase **scan** almacena en sus atributos la información extraída del fichero de adquisición. Como particularidad, comentar que el atributo **cartesian_pp** es un **puntero a puntero**.

Éste apunta a una tabla de punteros, donde cada uno de las posiciones de la tabla apunta a un objeto *cartesianVector* que recoge la información de las coordenadas de la superficie terrestre escaneada por el LD-MRS. Esto puede verse en la Figura 55.

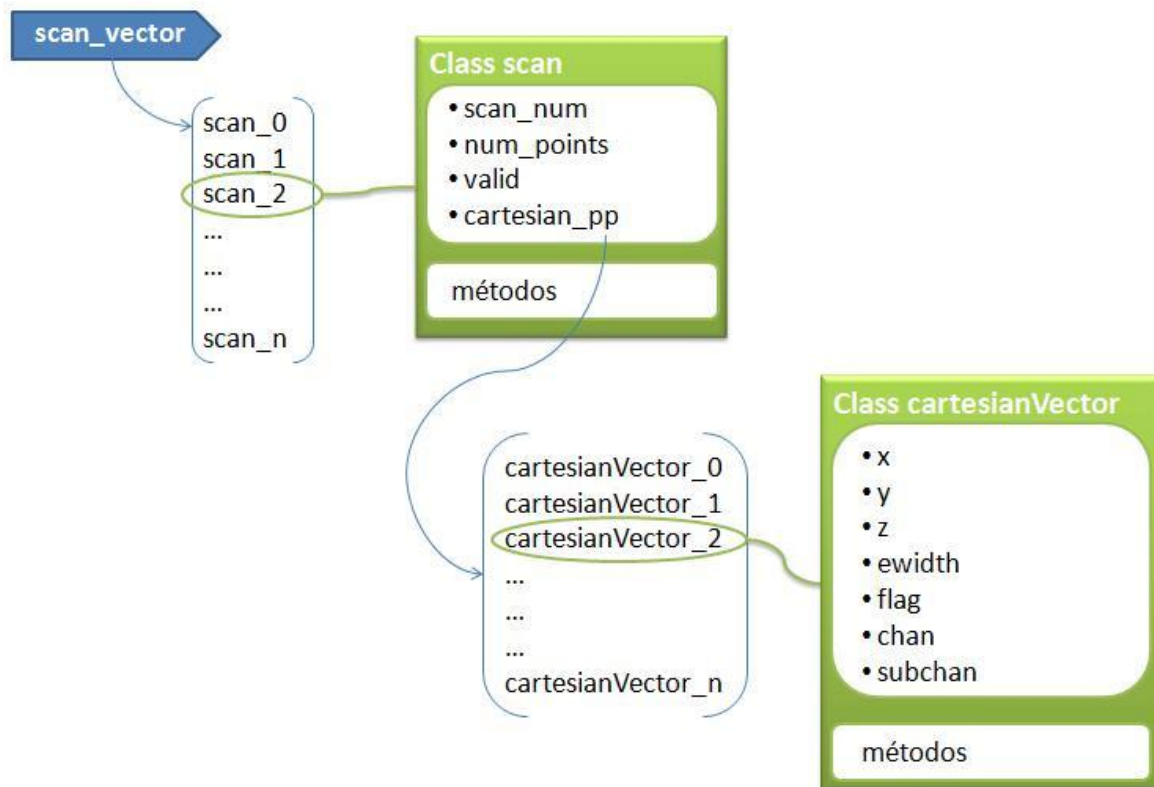


Figura 55. Estructura de las clases que recogen los datos

Por lo demás, dentro de la clase *scan* están los métodos **get** que se utilizan para acceder a la información que alberga el objeto, es decir, para acceder a los valores de los atributos y así poder operar con ellos. También están dentro de esta clase los métodos **set**, necesarios para modificar los valores de los atributos, muy útiles por ejemplo en la función **extract_scan()** en la que se necesita guardar en el objeto la información extraída del fichero de adquisición de datos.

Los demás métodos de la clase *scan* como son **apply_rotation()**, **apply_traslacion()** y **correct_euler()**, realizan las operaciones de traslación y rotación según los procedimientos que han sido descritos a lo largo de este apartado. El método **print_scan()** se encarga de copiar toda la información de los atributos del objeto *scan* en el fichero de salida, una vez que se hayan procesado sus coordenadas.

5.2.4. Formato del fichero de salida

El fichero de salida más importante es el que recoge las coordenadas procesadas de cada *scan*. Éstas coordenadas corresponden a puntos de la superficie terrestre expresadas en el sistema de coordenadas *n-frame* elegido para representar el MDT. Éstas coordenadas vienen expresadas en metros. En la Figura 56 se

puede ver un fichero de salida completo correspondiente a uno de los *scans* tras haber realizado las traslaciones y rotaciones correspondientes sobre las mismas. En el caso particular de este fichero de salida sólo contiene los puntos que aparecen en la imagen, es decir, no tiene más coordenadas procesadas válidas. Esto es debido a que en durante el procesamiento se descartan los puntos que proceden de reflexiones no deseadas, como por ejemplo al impactar sobre las partículas que están en suspensión en el aire, y en el caso particular de este *scan* éstas fueron las únicas coordenadas correspondientes a la superficie terrestre.

689	0.15284227	-0.47029109	-0.16231790	0	2	0
705	0.14425222	-0.46233290	-0.13246350	0	2	0
713	0.14211369	-0.46014328	-0.12155662	0	2	0
714	0.14453989	-0.45673358	-0.12173884	0	3	0
721	0.14525201	-0.46217984	-0.11789374	0	2	0
722	0.14767760	-0.45876975	-0.11807668	0	3	0
729	0.14834013	-0.46417566	-0.11416629	0	2	0
730	0.15076512	-0.46076518	-0.11434997	0	3	0

Figura 56. Fichero de salida correspondiente al *scan 125* procesado

Por lo general, los **primeros scans** que son adquiridos por el LD-MRS suelen contener **más puntos inválidos**, que luego son descartados en la etapa de procesamiento. También es común que en los primeros *scans*, los puntos válidos que los componen, estén muy cerca del origen de coordenadas del sistema de representación. Esto se aprecia en la Figura 56 donde ninguna de las coordenadas llega a la unidad, ya que éste *scan 125* es uno de los primeros *scans* con puntos válidos de uno de los experimentos de adquisición que se realizaron.

En el instante en el que la **aeronave** despegue y se **separa del suelo lo suficiente**, los *scans* empiezan a contener **más puntos válidos**. Esto se aprecia mejor en la Figura 57, que se corresponde con el *scan* número 758 del mismo vuelo de adquisición de datos del que se habla en el párrafo anterior.

Otros ejemplos de ficheros de salida son, como ya se comentó, los ficheros *debug*, *scans* y *lla_eul_vel*, los cuales se pueden ver a continuación en la Figura 58, la Figura 59 y la Figura 60 respectivamente.

```

1 21.54004900 -30.91889233 0.12248491 0 1 0
4 21.00834426 -30.90258193 0.21542601 0 0 0
5 21.06404815 -31.08696058 0.20852777 0 1 0
8 20.64028035 -31.03335549 0.22667784 0 0 0
9 20.69464873 -31.21363390 0.22017462 0 1 0
12 20.33195128 -31.13978101 0.19686579 0 0 0
13 20.35014424 -31.33121174 0.22322173 0 1 0
16 19.99143007 -31.26072042 0.20890236 0 0 0
17 20.04363722 -31.43424342 0.20308298 0 1 0
20 19.64769801 -31.38400357 0.23760097 0 0 0
21 19.73274718 -31.53978270 0.19831704 0 1 0
24 19.34125106 -31.49216582 0.24190941 0 0 0
25 19.39791404 -31.65591735 0.22989818 0 1 0
28 19.07054528 -31.58559788 0.21982126 0 0 0
29 19.08682046 -31.76316745 0.25008314 0 1 0
32 18.82114091 -31.67035057 0.18365712 0 0 0
33 18.83737757 -31.84562754 0.21478279 0 1 0
36 18.52284883 -31.77729727 0.21204603 0 0 0
37 18.56406017 -31.93861072 0.21504502 0 1 0
40 18.24666637 -31.87531374 0.22667242 0 0 0
41 18.29335965 -32.03118302 0.22270347 0 1 0
44 17.97955376 -31.97007610 0.24134988 0 0 0
45 18.02543502 -32.12328541 0.23766677 0 1 0
48 17.76272228 -32.04286653 0.20242380 0 0 0
49 17.80815716 -32.19446848 0.19894836 0 1 0
52 17.50638082 -32.13416154 0.22281806 0 0 0
53 17.54521938 -32.28574586 0.22733865 0 1 0
56 17.29260935 -32.20664289 0.19526663 0 0 0
57 17.33113734 -32.35658097 0.20009927 0 1 0
58 17.32058381 -32.52146831 0.19286404 0 2 0
59 17.33326179 -32.67989991 0.23268731 0 3 0
64 17.05798198 -32.28931303 0.20464811 0 0 0
65 17.09602231 -32.43707640 0.20982888 0 1 0
66 17.07977527 -32.60204028 0.21220604 0 2 0
67 17.09817194 -32.75581846 0.24482406 0 3 0

```

Figura 57. *Scan 758* corregido de un vuelo de adquisición de datos

```

0 scans lost
1374 scans good
0 scans with invalid UTC data
123 scans empty of valid points
0 scans with GPS errors

```

Figura 58. Ejemplo del fichero de salida *debug*


```

125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145

```

Figura 59. Ejemplo del fichero de salida *scans*

2	37.42623520	-6.00334358	60.64235306	0.36226061	-0.40897977	43.50856781	0.15881580	0.10957484	0.23453215
3	37.42623520	-6.00334358	60.64788055	0.35462722	-0.44815108	43.49644470	0.17066751	0.12180597	0.23991022
4	37.42623520	-6.00334311	60.66332626	0.26022339	-0.46780387	43.54830933	0.16931209	0.10292459	0.25569645
5	37.42623520	-6.00334311	60.66841125	0.30844447	-0.47458932	43.60456085	0.19316234	0.10720102	0.24763760
6	37.42623520	-6.00334358	60.68268204	0.32059243	-0.52682245	43.50205994	0.15726267	0.13554095	0.24536662
7	37.42623520	-6.00334358	60.71196747	0.31745943	-0.61427712	43.59379578	0.15571898	0.12824067	0.24348059
8	37.42623520	-6.00334358	60.72713470	0.26409519	-0.59641105	43.50350189	0.17405087	0.09855521	0.24962299
9	37.42623520	-6.00334358	60.71738815	0.27619109	-0.53811777	43.53278351	0.15955131	0.11569468	0.23707895
10	37.42623520	-6.00334358	60.73255157	0.20638724	-0.59331006	43.74286270	0.18179765	0.06082655	0.23353493
11	37.42623520	-6.00334358	60.74355698	0.25549015	-0.61740673	43.80355453	0.17429142	0.07283134	0.22841987
12	37.42623520	-6.00334358	60.75689316	0.25846484	-0.61597908	43.91370010	0.19030070	0.08426794	0.21468383
13	37.42623520	-6.00334358	60.77131653	0.28946796	-0.53485316	43.72544479	0.20087785	0.07874766	0.21158236
14	37.42623520	-6.00334358	60.78516388	0.24633424	-0.48686159	43.81587601	0.17548878	0.09010690	0.23678230
15	37.42623520	-6.00334358	60.78582001	0.28960302	-0.41212812	43.80891800	0.17101151	0.05698334	0.21314901
16	37.42623520	-6.00334358	60.79941177	0.34355298	-0.57341391	43.96395111	0.15413308	0.07982838	0.24022992
17	37.42623520	-6.00334358	60.78889847	0.37474614	-0.77053916	43.87061310	0.13062735	0.08335045	0.25525650
18	37.42623520	-6.00334358	60.80395126	0.28577134	-0.64542788	43.92066956	0.18538956	0.10104062	0.22595176
19	37.42623520	-6.00334358	60.82551193	0.24413222	0.33524442	43.78601837	0.20140743	0.08873662	0.26774463

Figura 60. Ejemplo del fichero de salida *lla_eul_vel*

5.2.5. Pseudocódigo

En el anexo de este mismo documento se adjuntan diagramas de flujo que representan el pseudocódigo de esta parte del software, correspondiente al procesamiento de los datos.