UNIVERSIDAD DE SALAMANCA

ESCUELA POLITÉCNICA SUPERIOR DE ÁVILA

Ingeniería Técnica en Topografía
Departamento de Ingeniería Cartográfica y del Terreno

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Proyecto Final de Carrera

Desarrollo de Software fotogramétrico para la elaboración de modelos digitales de elevaciones de alta resolución a partir de múltiples tomas convergentes

Autor:

Jonathan Herrero Redondo

Tutores:

Carlos Pérez Gutiérrez
José Antonio Martín Jiménez

Ávila, Septiembre 2012

Índice

Prólogo

Presentación

CAPÍTULO 1. INTRODUCCIÓN 1

1.1. Antecedentes 1

CAPÍTULO 2. FUNDAMENTOS TEÓRICOS 5

2.1. Fotogrametría Digital 5

2.1.1. Cámaras "amateur" no profesionales 6

2.1.2. Toma convergente 7

2.1.3. Toma múltiple 7

2.2. Colinealidad 8

2.3. Correspondencia basada en áreas 10

2.4. Metodología de la orientación interna 14

2.5. Metodología de la orientación externa 17

2.6. Restitución de puntos: Cálculo de coordenadas objeto 19

2.7. Generación de ortofotos 21

2.8. El Modelo Digital de Elevaciones 23

2.8.1. Estructuras de un MDE 23

2.8.2. Procedimientos de obtención 25

2.8.3. Etapa analógica y analítica para generar cartografía por fotogrametría 28

2.8.4. Etapa digital para generar MDE por fotogrametría 30

2.8.5. Aplicaciones del MDE 32

CAPÍTULO 3. INTRUMENTACIÓN 41

CAPÍTULO 4. EXPERIENCIA DE USUARIO 53
4.1. Apertura del programa 53
4.2. Especificaciones de la cámara 55
4.3. Selección de las imágenes y orientación interna 56
4.4. Introducción de los Puntos de Apoyo 57
4.5. Captura/Medida de dianas 58
4.6. Orientación externa 60
4.7. Creación del MDE 63
4.7.1. Primeros pasos: uso de la línea epipolar y vertical del lugar 63
4.7.2. Procedimiento de cálculo 68
4.8. Ortofoto y textura del MDE 74
4.9. Depuración del MDE 75
4.10. Perfil longitudinal 76
CAPÍTULO 5. ALGORITMIA 79
5.1. Operaciones previas 79
5.1.1. Parámetros de la cámara y carga de imágenes 80
5.1.2. Función captura de coordenadas de las dianas 83
5.1.3. Función para la Orientación Externa 86
5.2. Creación del MDE 91
5.2.1. MDE por correspondencia basada en areas 92
5.2.2. Depuración 96
5.2.3. Ortofoto 97
5.3. Funciones adicionales 101
5.3.1. Perfil longitudinal 101
5.3.2. Función para crear líneas epipolares 103
5.3.3. Vertical del lugar 109
5.4. Funciones previas descartadas 116

CAPÍTULO 6. RESULTADOS 129
6.1. Resultados con las fotos de "CHISME" 129
6.1.1. Zona lisa 129
6.1.2. Zona media 132
6.1.3. Zona rugosa 135
6.2. Comparativa con los resultados ofrecidos por Orthoengine PCI Geomatica 138
6.3. Prueba con "CHISME" de objetos regulares 144
CAPÍTULO 7. CONCLUSIONES 155
ANEXO: Calibración de la cámara con PhotoModeler Scanner 157
ANEXO: Lenguaje de Programación: Matlab 163
ANEXO: Interfaz Gráfica de Usuario (GUI) 187
ANEXO: Planificación de desarrollo 203
ANEXO: Justificación de precios y Presuspuesto 209
ANEXO: DVD del Proyecto en formato digital 221
Bibliografía 223

Índice de figuras

Figura 2-1. Colinealidad....8
Figura 2-2. Correspondencia basada en áreas....10
Figura 2-3. Relación entre sistemas de coordenadas en la orientación interna....14
Figura 2-4. Orientación externa....17
Figura 2-5. Método ascendente....21
Figura 2-6. Corte en el método ascendente....21
Figura 2-7. Método descendente....22
Figura 2-8. Restituidores del laboratorio de la Escuela Politécnica Superior de Ávila....29
Figura 2-9. Ejemplo de MDE con vista en 3D (fuente: internet)....31
Figura 2-10. Modelo digital de pendientes....33
Figura 2-11. Modelo digital de orientaciones....33
Figura 2-12. Modelo digital de sombreado....34
Figura 2-13. Modelo digital de rugosidad....35
Figura 2-14. Índice de elevación-relieve....35
Figura 2-15. Índice de varianza pendiente....36
Figura 2-16. Curvatura....36
Figura 2-17. Flujo acumulado (Matriz simple)....37
Figura 2-18. Flujo acumulado (Matriz múltiple)....37
Figura 2-19. Cuencas hidrográficas....38
Figura 2-20. Índice de humedad....39
Figura 2-21. Índice de potencia de cauce....39
Figura 3-1. Informe de resultados....43
Figura 3-2. Sistema de coordenadas de CHISME....45
Figura 3-3. De arriba hacia abajo: zona lisa, media y rugosa....46
Figura 3-4. Distintas orientaciones de una misma zona rugosa....47
Figura 4-1. Icono del programa....53
Figura 4-2. Ventana de la interfaz principal....54
Figura 4-3. Interfaz de especificaciones de la cámara....55
Figura 4-4. Interfaz cargar fotos....56
Figura 4-5. Interfaz para pedir puntos de apoyo....57
Figura 4-6. Captura de las fotocoordenadas por pantalla....59
Figura 4-7. Botón calculo de la Orientación Externa....60
Figura 4-8. Barra estado progreso de la Orientación Externa....60

Figura 4-9. Menú herramientas para ver/editar OE....62
Figura 4-10. Panel de vista y edición de los parámetros de OE....62
Figura 4-11. Condición de colinealidad....63
Figura 4-12. Proyección de la línea epipolar....64
Figura 4-13. Proyección de la vertical del lugar....65
Figura 4-14. Intersección de los rayos proyectivos de los límites con el terreno....66
Figura 4-15. Posible error por paralelismo entre línea epipolar y vertical del lugar....67
Figura 4-16. La recta en la imagen digital como caso discreto....68
Figura 4-17. Desplazamientos en el espacio objeto....69
Figura 4-18. Fragmento de código para elegir el mejor candidato entre correlaciones máximas....72
Figura 4-19. Ventana interfaz para configurar el MDE....74
Figura 4-20. Botón para crear ortofoto y visualizar MDE....75
Figura 4-21. Menú herramientas para el perfil longitudinal....76
Figura 4-22. Ejemplo de un perfil longitudinal....76
Figura 5-1. Diagrama de flujo de las operaciones previas....79
Figura 5-2. Función cargar foto....81
Figura 5-3. Función focal, alto, ancho....82
Figura 5-4. Función captura....83
Figura 5-5. Función MiZoom....84
Figura 5-6. Función orientación interna....85
Figura 5-7. Función orientación interna....86
Figura 5-8. Función orientación externa....86
Figura 5-9. Función orientación externa....87
Figura 5-10. Función orientación externa....88
Figura 5-11. Función orientación externa....89
Figura 5-12. Función orientación externa....90
Figura 5-13. Diagrama para la elaboración del MDE....91
Figura 5-14. Función crear MDE por Correspondencia....92
Figura 5-15. Función crear MDE por Correspondencia....93
Figura 5-16. Función crear MDE por Correspondencia....94
Figura 5-17. Función crear MDE por Correspondencia....95
Figura 5-18. Función crear MDE por Correspondencia....96
Figura 5-19. Función depurar MDE....96
Figura 5-20. Función crear ortofoto....97
Figura 5-21. Función crear ortofoto....98
Figura 5-22. Función crear ortofoto....99
Figura 5-23. Función crear ortofoto.... 100

Figura 5-24. Función perfil longitudinal 101
Figura 5-25. Función perfil longitudinal 102
Figura 5-26. Función línea epipolar 103
Figura 5-27. Función línea epipolar 104
Figura 5-28. Función línea epipolar 105
Figura 5-29. Función línea epipolar 106
Figura 5-30. Función línea epipolar 107
Figura 5-31. Función línea epipolar 108
Figura 5-32. Función línea epipolar 109
Figura 5-33. Función vertical del lugar 109
Figura 5-34. Función vertical del lugar 110
Figura 5-35. Función vertical del lugar 111
Figura 5-36. Función vertical del lugar 112
Figura 5-37. Función vertical del lugar 113
Figura 5-38. Función vertical del lugar 114
Figura 5-39. Función vertical del lugar 115
Figura 5-40. Función localización por correspondencia 116
Figura 5-41. Función localización por correspondencia 117
Figura 5-42. Función localización por correspondencia 118
Figura 5-43. Función localización por correspondencia 119
Figura 5-44. Función cálculo verticales del lugar 119
Figura 5-45. Función cálculo verticales del lugar 120
Figura 5-47. Función localizar homólogos 4 fotos 121
Figura 5-46. Función cálculo verticales del lugar 121
Figura 5-48. Función loclizar homólogos 4 fotos 122
Figura 5-49. Función localizar homólogos en terreno 123
Figura 5-50. Función localizar homólogos en terreno 124
Figura 5-51. Función localizar homólogos en terreno 125
Figura 5-52. Función localizar homólogos en terreno 126
Figura 5-53. Función coordenadas terreno finales 127
Figura 6-1. Modelo zona lisa 1 129
Figura 6-3. Modelo zona lisa renderizado 1 130
Figura 6-2. Modelo zona lisa 2 130
Figura 6-4. Modelo zona lisa renderizado 2 131
Figura 6-5. Orfotos zona lisa 131
Figura 6-6. Perfil longitudinal zona lisa 132
Figura 6-7. Modelo zona media 1 132

Figura 6-9. Modelo zona media renderizdo 1 133
Figura 6-8. Modelo zona media 2 133
Figura 6-10. Modelo zona media renderizado 2 134
Figura 6-11. Orfotos zona media 134
Figura 6-12. Perfil longitudonal zona media 135
Figura 6-13. Modelo zona rugosa 1 135
Figura 6-14. Modelo zona rugosa 2 136
Figura 6-15. Modelo zona rugosa renderizado 136
Figura 6-16.Perfil longitudinal zona rugosa 137
Figura 6-17. Modelo zona rugosa renderizado menor resolución 137
Figura 6-18. Orfotos zona rugosa 138
Figura 6-19. Ventana inicio Orthoengine 138
Figura 6-20. Ventana cámara Orthoengine 139
Figura 6-21. Ventana captura de puntos 139
Figura 6-22. Ventana creación de imágenes epipolares Orthoengine 140
Figura 6-23. Ventana creación MDE 140
Figura 6-24. MDE resultado con Orthoengine de resolución 5 píxel en la izquierda y de 1 píxel a la derecha 141
Figura 6-25. MDE resultado con Orthoengine sobre Matlab 141
Figura 6-26. Matriz de diferencias 142
Figura 6-27. MDE resultado con Focus Geomatica (izquierda parte superior) y Orthoengine sobre Matlab (derecha parte inferior) 143
Figura 6-28. Imágenes de objetos regulares con "CHISME" 144
Figura 6-30. MDE objetos regulares 2 145
Figura 6-29. MDE objeto regulares 1 145
Figura 6-31. Orfotos de objetos regulares 146
Figura 6-32. Perfil longitudinal del calendario 147
Figura 6-33. Perfil longitudinal del libro 148
Figura 6-34. Perfil longitudinal de la caja 149
Figura 6-35. Perfil longitudinal del dado 150
Figura 6-36. MDE objetos regulares obtenido con Orthoengine 151
Figura 6-37.Matriz de diferencias 152
Figura 8-1. Imagen de la rejilla utilizada por Photomodeler Scanner 157
Figura 8-2. Ventana inicio 158
Figura 8-3. Selección de imágenes 158
Figura 8-4. Tipo de cámara y focal conocida o desconocida 159
Figura 8-5. Ventana de ejecución 159
Figura 8-6. Informe de resultados 160

Figura 9-1. Hipermatriz de tres dimensiones 174
Figura 9-2. Gráfico del vector x=[1 3 2 4 5 3] 182
Figura 9-3. Colores, markers y estilos de línea 182
Figura 9-4. Diferecias entre mesh y surf 184
Figura 10-1. Menú nuevo GUI 189
Figura 10-2. Panel GUIDE rápido 189
Figura 10-3. Aspecto GUI 191
Figura 10-4. Paleta de componentes 194
Figura 10-5. Inspector de propiedades 197
Figura 11-1. Diagrama de GANTT 206
Figura 11-2. Diagrama de PERT parte 1 206
Figura 11-3. Diagrama de PERT parte 2 207

Índice de tablas

Tabla 2-1. Ventajas y limitaciones de las cámaras convencionales 6
Tabla 2-2. Resumen procedimientos de obtención de un MDE. (Felicísimo, 1994) 26
Tabla 3-1. Especificaciones de la cámara 41
Tabla 3-2. Puntos de apoyo 48
Tabla 3-3. Coordenadas píxel 49
Tabla 4-1. Tabla modelo para los puntos de apoyo 58
Tabla 4-2. Informe de la orientación externa de un ejemplo 61
Tabla 4-3. Informe de correspondencia en el MDE 71

Prólogo

El presente proyecto titulado "Desarrollo de software fotogramétrico para la elaboración de modelos digitales de elevaciones de alta resolución a partir de múltiples tomas convergentes", tiene por objeto crear una aplicación informática que a partir de unos datos de entrada que son las fotografías de la zona a representar, con sus parámetros de la cámara calibrados; obtener una salida que tras varias operaciones previas llevan al MDE.

Se describe una metodología basada en fotogrametría convergente de objeto cercano. El objetivo prioritario es reducir costes, tanto en el aspecto económico, al no ser necesario tener que invertir en operarios e instrumentos especializados para tareas topográficas más o menos costosas; como de eficiencia en campo, al minimizar el tiempo necesario para la medida de cada muestra.

Si analizamos el título observamos que se emplean tomas convergentes. Las tomas convergentes quedan fuera del caso normal, en donde el plano de proyección es paralelo al objeto que se pretende representar, que en este caso sería el suelo. Con las tomas convergentes, al tener cierta inclinación del plano de proyección respecto del objeto, conseguimos que la intersección de los haces proyectivos sea mejor con ángulos de incidencia óptimos, mejorando la geometría final.

Dependiendo de la cámara utilizada, un ángulo de 10-15º con respecto a la vertical es lo más adecuado. Ángulos mayores, pueden producir que la disparidad entre las imágenes devenga en problemas de correlación posteriores.

Se pretende obtener un MDE de alta resolución en cuanto a tamaño se refiere, siendo este configurable y en cualquier caso con tamaño inferior a la resolución de la imagen. Sin embargo, si el tamaño de este es excesivo con una huella de pixel pequeña el tiempo de ejecución será elevado.

Para facilitar la elaboración de los modelos digitales de elevaciones con la aplicación se han hecho algunas pruebas con un dispositivo denominado CHISME (Chasis Imagen-Suelo para medidas elementales).

Adicionalmente al proyecto, dentro de la aplicación, se han incluido algunas operaciones como son crear una ortofoto de la zona que abarca el MDE y poder render el modelo utilizando esta como textura, crear un perfil longitudinal entre dos puntos solicitados por el usuario, y otras operaciones con salida visual como son ver la línea epipolar proyectadas en las fotografías a partir de un punto solicitado al usuario y por otra parte ver la perspectiva de la foto con las proyecciones de las verticales del lugar en las fotografías cuyas intersección nos dan la proyección del nadir.

Finalmente, concluimos los objetivos con la finalidad que tendrá este proyecto. El proyecto irá destinado al análisis de la rugosidad del terreno mediante fotogrametría, no obstante, son varios los usos que se pueden dar a la aplicación, como se verán posteriormente.

Presentación

Este proyecto final de carrera consta de siete capítulos y cinco anexos finalizando con la bibliografía o referencias utilizadas.

El primer de los capítulos es una introducción sobre lo que se ha venido haciendo en el curso y un poco la motivación para realizar este proyecto final de carrera.

El capítulo dos recoge los fundamentos teóricos que son empleados durante el desarrollo del software en cuanto a temática relacionada con fotogrametría digital en su mayoría.

En el tercer capítulo describimos la instrumentación empleada recogiendo características de la misma y algunas fotos ejemplos, así como algunos resultados del software.

Un cuarto capítulo denominado "experiencia de usuario" ilustra lo que el usuario se encuentra al ejecutar el programa. Este capítulo no constituye un manual del programa pero si una explicación de como funciona este internamente.

El quinto capítulo recoge el conjunto de funciones implementadas en Matlab que podemos utilizar de forma independiente ayudándonos al mismo tiempo de diagramas de flujo explicativos, por ello este es el capítulo de Algoritmia.

En el sexto capítulo encontramos resultados, como son las ejecuciones del programa con distintas fotografías y la comparativa realizada con software comercial.

El último capítulo recoge las conclusiones personales del proyecto.

Los anexos recogen como utilizar photomodeler scanner para calibrar la cámara, un pequeño manual sobre matlab y como crear interfaces gráficas de usuario y la planificación del proyecto, en cuanto distribución de tiempo empleado y el presupuesto del mismo.

CAPÍTULO 1. INTRODUCCIÓN

1.1. Antecedentes

En el mercado actual encontramos disponibles varios productos de fotogrametría digital, entre estos, encontramos software que nos permite la generación de Modelos Digitales de Elevaciones así como Ortofotos digitales.

Durante los estudios de Ingeniería Técnica en Topografía se nos han enseñado varias aplicaciones y como usarlas para obtener estos productos.

Uno de estos programas que se usó en el tercer curso fue Orthoengine de PCI Geomatica, con este software aprendimos a crear los productos anteriores. Para ello, se nos facilitaban unos fotogramas aéreos en los que realizábamos la Orientación Interna con ayuda de los datos facilitados por el certificado de calibración de la cámara, para, posteriormente, realizar la Orientación Externa con ayuda de la información que nos proporciona el visor cartográfico del Instituto Geográfico Nacional (IGN).

Realizada la Orientación Externa, estábamos en condiciones de realizar las siguientes tareas que el software proporciona pasando a la generación una ortofotografía digital.

El profesor de la asignatura nos proporcionó un Modelo Digital de Elevaciones (MDE) de la zona para este propósito. La obtención de un MDE es un proceso que conlleva gran cantidad de tiempo en función del detalle con el que se quiera obtener, así para agilizar la tarea de generación de una Ortofoto se nos proporcionó el MDE.

Sabemos que este software es capaz de realizar la tarea de generar un MDE previamente generando imágenes epipolares, en clase se vio dicha tarea para una zona pequeña de las imágenes.

Orthoengine al igual que otros muchos programas presentan el inconveniente de no darnos total información sobre como realizan cada una de sus tareas, por ejemplo, en cuanto a correspondencia de imágenes se refiere para encontrar puntos homólogos.

En este proyecto presentamos una aplicación desarrollada con Matlab que permite generar un MDE a partir de fotogrametría convergente, explicando los pasos que se dan para obtener el mismo.

CAPÍTULO 2. FUNDAMENTOS TEÓRICOS

2.1. Fotogrametría Digital

La Asociación Americana de Fotogrametría y Teledetección (ASPRS) define la Fotogrametría como "..el arte, ciencia y tecnología de obtener información fidedigna de los objetos físicos y del medio ambiente mediante procesos de registro, medición e interpretación de imágenes fotográficas y de modelos de energía radiante electromagnética y otros fenómenos".

En la fotogrametría digital se hace uso de imágenes digitales que generalmente se obtienen con cámaras fotográficas digitales.

Podemos definir la imagen digital como una función discreta $f(x, y)$ donde las coordenadas x, y son variables espaciales y la función valor es la densidad. Normalmente, la imagen se representa como una matriz que contiene en cada celda un determinado nivel de gris.

Cuando trabajamos con imágenes en color tenemos tres canales, RGB, uno para el rojo, otro para el verde y otro para el azul. De esta forma, se forman tres matrices por canal con un nivel digital de intensidad.

A diferencia de la fotografía convencional con la fotografía digital esta se puede "tocar" en cuanto a que en ella podemos llevar a cabo de forma sencilla tareas como variación de brillo y contraste de la misma.

Actualmente, podemos elegir distintos tipos de cámaras digitales para realizar nuestros trabajos fotogramétricos pudiendo ser estas métricas o no métricas.

Vamos a analizar a continuación algunas diferencias que se dan entre distintos tipos de cámaras para posteriormente adentrarnos al tipo de tomas que se van a realizar en este proyecto.

2.1.1. Cámaras "amateur" no profesionales

Las cámaras fotográficas para la fotogrametría son tal vez los instrumentos fotogramétricos más importantes, ya que con ellas se toman las fotos de la que depende esta ciencia.

En fotogrametría se conocen dos tipos de cámaras: las cámaras métricas y las cámaras no métricas.

Las cámaras métricas digitales son las cámaras fabricadas específicamente para ser utilizadas con fines métricos y que por lo tanto comparten características fundamentales con una construcción robusta del conjunto lentes – sensor, una distancia focal fija, y unas lentes de alta calidad. Todas las demás cámaras se considerarán como no métricas.

Dentro de las cámaras digitales no métricas (también llamadas cámaras estándar o cámaras nóveles en la literatura científica) y desde el punto de vista de sus capacidades métricas y su uso en fotogrametría de objeto cercano se suele distinguir dos categorías:

- Cámaras profesionales ("high-grade" ó"highquality")
- Cámaras de consumo ("amateur" o "lowcost")

En este proyecto vamos hacer uso de estas últimas por el bajo coste que suponen al poder utilizar una cámara digital cualquiera sin tener grandes limitaciones por ello. Recogemos a continuación las ventajas y limitaciones de las mismas:

Sus ventajas:	Las limitaciones:
- Tamaño - Peso reducido - Objetivos intercambiables - Facilidad de compra; hace que se extienda su utilización	- Geometría interna desconocida - Sistema de lentes imperfecto - Ondulación del plano focal

Tabla 2-1. Ventajas y limitaciones de las cámaras convencionales

Se hace por tanto necesario determinar los parámetros de calibración que nos imponen las limitaciones.

2.1.2. Toma convergente

Con el fin de optimizar el ángulo de incidencia entre modelos y mejorar la geometría final es conveniente que las tomas se realicen ligeramente convergentes. Dependiendo de la cámara utilizada, un ángulo de 10-15º con respecto a la vertical es lo mas adecuado.

Ángulos mayores, pueden producir que la disparidad entre imágenes devenga en problemas de correlación posteriores.

A diferencia del caso normal donde cada toma es paralela entre si y perpendicular a la base, en la toma convergente tenemos que las direcciones de toma de cada fotograma no son paralelas entre sí, ni perpendiculares a la base entre las parejas de imágenes.

2.1.3. Toma múltiple

La reconstrucción del terreno o de un modelo digital de elevaciones implica combinar diferentes imágenes que nos permiten la búsqueda de puntos homólogos.

Cuando hemos realizado tareas previas de orientación de la imagen, con sus cálculos y tenemos parámetros de la orientación externa estamos en condiciones de restituir puntos.

El protocolo de adquisición de las imágenes impone el uso de puntos de apoyo a modo de dianas de puntería situadas en el terreno con el fin de que aparezcan en cada toma. Las tomas fotográficas se realizan desde una altura en la que el campo angular de la cámara capte toda la zona a representar.

Usando cámaras de bajo coste y para fotogrametría de objeto cercano, en función de la apertura angular de la cámara, esta distancia puede variar entre 0.65 y 1.60 m. Las tomas se realizan a mano alzada, sin necesidad de trípode.

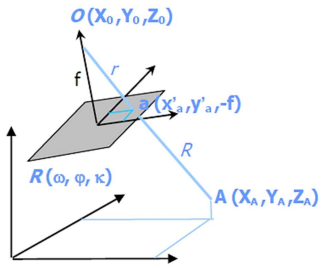
Las dimensiones que se pretenden abordar como ejemplo práctico con el software a elaborar, lo hacen ideal para que todas las fotografías lo abarquen con un 100% de recubrimiento. Es importante que entre una y otra toma exista una distancia base que ronde aproximadamente 1/4 ó 1/5 de la altura de la cámara.

2.2. Colinealidad

La condición de colinealidad establece que al orientar la fotografía al momento de la toma, se cumple que el punto de vista, el punto imagen y su correspondiente en el terreno, están alineados.

Dicho en términos geométricos: el vector $\vec{r}$ materializado entre el punto de vista y el punto de la fotografía, y el vector $\vec{R}$, materializado por el punto de vista y el mismo punto en el terreno, deben tener el mismo origen y ser colineales.

El modelo matemático es de la siguiente forma:



$$\begin{bmatrix} X_A - X_0 \\ Y_A - Y_0 \\ Z_A - Z_0 \end{bmatrix} = \lambda \cdot R \cdot \begin{bmatrix} x'_a \\ y'_a \\ -f \end{bmatrix}$$

Figura 2-1. Colinealidad

En este sistema, el vector $\vec{r}$ queda expresado como:

$$\vec{r} = (x'_a - x'_0)\vec{\imath} + (y'_a - y'_0)\vec{\jmath} + (z'_a - z'_0)\vec{k}$$

Y puesto que las coordenadas de los puntos extremos de este vector son $(x'_a, y'_a, 0)$ y $(0, 0, -f)$ se tendrá que

$$\vec{r} = x'_a\,\vec{\imath} + y'_a\,\vec{\jmath} + f\,\vec{k}$$

La relación entre el vector $\vec{r}$ en el sistema terrestre y en el sistema fotograma queda recogida mediante una matriz de rotación tridimensional. Dicha matriz es el resultado de aplicar al sistema imagen un giro sobre el eje x', otro sobre el eje y' y finalmente un giro sobre el eje f.

Existen varias formas de expresar matemáticamente la condición de colinealidad según reordenemos los elementos integrantes del mismo.

$$\begin{bmatrix} x'_a \\ y'_a \\ -f \end{bmatrix} = \frac{1}{\lambda} \cdot R^{-1} \cdot \begin{bmatrix} X_A - X_0 \\ Y_A - Y_0 \\ Z_A - Z_0 \end{bmatrix}$$

Los parámetros que intervienen en este modelo matemático son:

- La focal f
- Las coordenadas terreno del centro de proyección: $X_0\ Y_0\ Z_0$
- Las coordenadas terreno del punto en el terreno: $X_A\ Y_A\ Z_A$
- Las coordenadas placa del punto en la fotografía: $x'_a\ y'_a$
- Los tres giros que relacionan el sistema terreno con el sistema imagen: ω, φ, ϰ.
- El factor de escala λ que relaciona vector imagen y objeto.

Podemos evitar la aparición de λ si dividimos las dos primeras filas que nos da el sistema matricial entre la tercera fila, obteniendo una ecuación que nos permite obtener fotocoordenadas:

$$x'_a = -f\frac{r_{11}\cdot(X_A - X_O) + r_{12}\cdot(Y_A - Y_O) + r_{13}\cdot(Z_A - Z_O)}{r_{31}\cdot(X_A - X_O) + r_{32}\cdot(Y_A - Y_O) + r_{33}\cdot(Z_A - Z_O)}$$

$$y'_a = -f\frac{r_{21}\cdot(X_A - X_O) + r_{22}\cdot(Y_A - Y_O) + r_{23}\cdot(Z_A - Z_O)}{r_{31}\cdot(X_A - X_O) + r_{32}\cdot(Y_A - Y_O) + r_{33}\cdot(Z_A - Z_O)}$$

El modelo matemático de colinealidad será fundamental para calcular la orientación externa y para llevar a cabo la calibración de la cámara ya que este modelo es usado por Photomodeler en esta tarea.

2.3. Correspondencia basada en áreas

La correspondencia por áreas toma entidades compuestas por niveles de gris. La idea de este método es comparar la distribución de niveles de gris de una pequeña subimagen, denominada tesela imagen, con su correspondiente en la otra imagen. En la siguiente figura se ilustra el concepto y se introduce la terminología más frecuente.

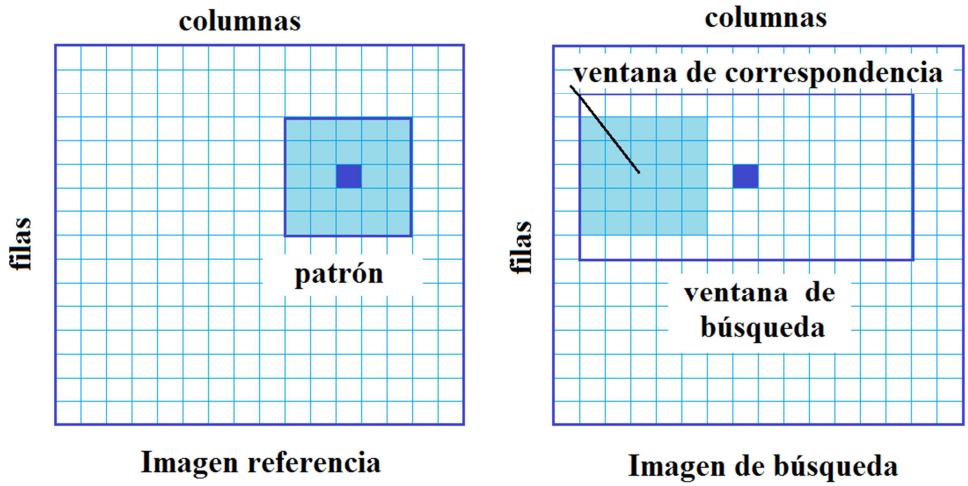


Figura 2-2. Correspondencia basada en áreas

El patrón (ventana patrón) es la tesela imagen que permanece en una posición fija en una de las imágenes. Se entiende por ventana de búsqueda la zona del espacio de búsqueda dentro de la que se comparan las teselas imagen (llamadas ventanas de correspondencia) con el patrón, comparación que se realiza utilizando distintos criterios para calcular el índice de semejanza. Los dos más conocidos son la correlación cruzada y la correspondencia por mínimos cuadrados. La posición y tamaño de la ventana de búsqueda puede determinarse o acotarse en función de alguna restricción geométrica.

La correspondencia entre dos píxeles se establece a partir de la correlación (grado de semejanza) de los niveles de gris correspondientes a los píxeles pertenecientes a sendas máscaras (matrices) centradas sobre los píxeles candidatos.

A partir de un píxel que actúa como referencia (dato) en una imagen (cuya máscara no se mueve) se busca en la otra imagen (o imágenes) el píxel (o píxeles) cuya máscara ofrece el mayor nivel de correlación. Esta máscara de correlación debe moverse por el espacio de búsqueda, con lo que se hace crucial poder acotar este espacio.

Tiene un carácter geométrico (no es invariante a la geometría de la máscara) y local (no se hace correspondencia de imágenes consideradas en su totalidad).

La técnica funciona adecuadamente siempre y cuando exista una regularidad geométrica entre las dos zonas de la imagen.

Resulta sensible a:

Pobreza de textura: no hay diferencias entre los niveles de gris (en un erial)

Patrones repetitivos: diversos objetos presentan formas radiométricas semejantes (ventanas de una fachada)

Variaciones de escala: Si el mismo objeto presenta imágenes a distinta escala la correlación ya no es significativa).

Rotaciones: Si el mismo objeto aparece rotado en distintas imágenes la correlación ya no es significativa).

Variaciones de relieve: Si la superficie del objeto forma ángulos distintos con los rayos perspectivos de las distintas imágenes la correlación ya no es significativa.

Estos factores hacen que esta herramienta sea semejante a la visión estereoscópica

Independientemente del método de cálculo del índice de semejanza, deben tratarse algunos puntos que a continuación se describen brevemente:

Posición del patrón: El aspecto de la elección del centro del patrón puede parecer trivial. En teoría, el centro del patrón podría situarse dentro de un área que fuera la mitad del tamaño del patrón y menor que la imagen, pero un análisis más cuidadoso indicaría que hay que ser más selectivo a la hora de aceptar dichos límites. La correspondencia por áreas puede fallar en determinadas condiciones. Sirvan como ejemplos: la colocación del patrón en zonas que presentan oclusión en la otra imagen, la selección de un área de baja relación señal ruido, la elección de un área con patrones repetidos, la ubicación en un área con líneas de ruptura, etc.

Tamaño del patrón: El tamaño del patrón y de la ventana de correspondencia son dos parámetros importantes. Al aumentar el tamaño, normalmente aumenta la unicidad de la función de niveles de gris, pero también lo hacen los problemas de distorsión geométrica por lo que debe encontrarse una solución de compromiso, por ejemplo calculando un índice de unicidad para distintos tamaños del patrón, que sirva también para controlar una ubicación útil del mismo.

Posición y tamaño de la ventana de búsqueda: La posición de la ventana de búsqueda es crucial, ya que la correspondencia por áreas necesita aproximaciones muy buenas. Su tamaño no obstante, no juega un papel importante porque la aproximación limita el tamaño a unos pocos píxeles. Una estrategia de correspondencia basada en el método jerárquico asegura la obtención de buenas aproximaciones.

Criterios de tolerancia: Deben analizarse los factores que aparecen en la medida de la semejanza entre el patrón y la ventana de correspondencia. El criterio de aceptación-rechazo cambia, incluso dentro de la misma imagen, por lo que los valores umbral u otros deberían definirse localmente, umbrales adaptativos.

Control de calidad: El control de calidad incluye una valoración de la precisión y de la exactitud de las posiciones homólogas o conjugadas. Además debe estudiarse la consistencia de los puntos correspondientes, incluyendo su adecuación a las expectativas o el conocimiento sobre el espacio objeto.

La idea es medir el grado de semejanza del patrón con la ventana de correspondencia calculando el factor de correlación.

Factor de correlación cruzado:

El coeficiente de correlación ρ se define como:

$$\rho = \frac{\sigma_{LR}}{\sigma_L \cdot \sigma_R}$$

Si ρ está normalizado se verifica -1≤ρ≤ 1. En la ecuación anterior los términos se definen como:

σ_{LR}, covarianza de las teselas de imagen L (izquierda) y R (derecha).

σ_{L}, desviación típica o estándar de la tesela de imagen L (patrón).

σ_{R}, desviación típica de la tesela de imagen R (ventana de correspondencia).

Si se introducen las funciones imagen $g_L(x,y)$ $g_R(x,y)$ para las teselas de imagen izquierda y derecha (en este caso el patrón y la ventana de correspondencia) y se denotan de la forma g_L , g_R se obtienen las siguientes ecuaciones de definición:

$$\overline{g_L} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} g_L(x_i, y_j)}{n \cdot m}$$

$$\overline{g_R} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} g_R(x_i, y_j)}{n \cdot m}$$

$$\sigma_L = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{m}(g_L(x_i, y_j) - \overline{g_L})^2}{n \cdot m - 1}}$$

$$\sigma_R = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{m}(g_R(x_i, y_j) - \overline{g_R})^2}{n \cdot m - 1}}$$

$$\sigma_{LR} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}(g_L(x_i, y_j) - \overline{g_L}) \cdot (g_R(x_i, y_j) - \overline{g_R})}{n \cdot m - 1}$$

El factor de correlación cruzado se determina dentro del espacio de búsqueda para cada posición f,c de la ventana de correspondencia. El problema siguiente es determinar qué posición u, v proporciona el máximo factor de correlación.

Factor de correlación máximo:

El factor de correlación máximo cruzado normalizado ρ toma valores en el intervalo [-1, 1]. Se obtiene un valor igual a la unidad si el patrón y la ventana de correlación son idénticos. Si no existe correlación entre las dos teselas de imagen, es decir, si no hay coincidencia en absoluto, entonces ρ=0. El valor ρ=-1 indicaría una correlación inversa, como en el caso de la diapositiva y el negativo de la misma imagen.

2.4. Metodología de la orientación interna

La orientación interna pretende conseguir la transformación entre dos sistemas de coordenadas bidimensionales, uno el de la imagen digital con coordenadas pixel dadas por la fila y columna y otro el de las fotocoordenadas.

Esta relación de sistemas de coordenadas se lleva a cabo relacionando el tamaño de la imagen en píxeles y el tamaño del sensor en milímetros.

Para el cálculo de la orientación interna se ha optado por una transformación afín compuesta por una traslación, un giro y dos factores de escala.

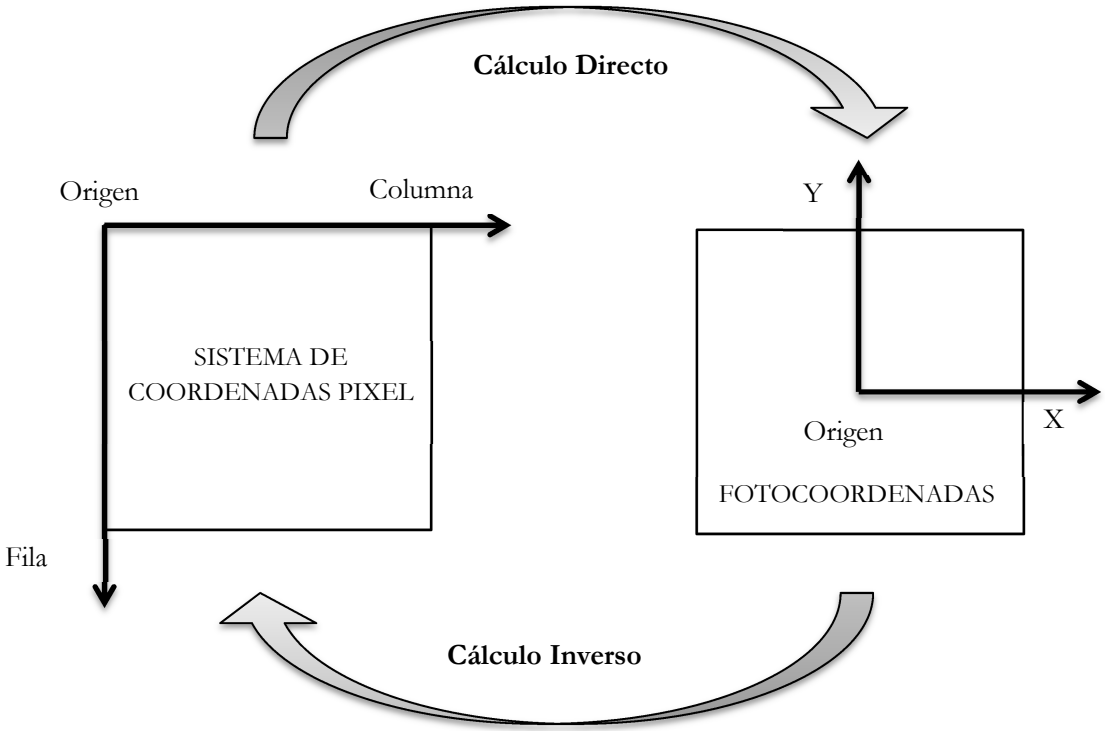


Figura 2-3. Relación entre sistemas de coordenadas en la orientación interna

Tenemos que tener en cuenta que en algunos casos necesitaremos un cálculo directo para pasar de coordenadas pixel a fotocoordenadas y otras veces un cálculo inverso como se ve en la figura.

El primer caso se da cuando marcamos elementos sobre la imagen por tanto en coordenadas píxel y queremos sus fotocoordenadas para trabajar con ellas, por ejemplo, en una orientación externa.

El segundo caso se da cuando queremos pasar datos calculados a la imagen para que se dibujen y sean vistos gráficamente.

Los parámetros que intervienen en una transformación afín para el cálculo directo quedan de la siguiente forma:

$$x = \lambda_{columnas} \cdot columna \cdot \cos\omega + \lambda_{columnas} \cdot fila \cdot \operatorname{sen}\omega + columna_T$$

$$y = -\lambda_{filas} \cdot columna \cdot \operatorname{sen}\omega + \lambda_{filas} \cdot fila \cdot \cos\omega + fila_T$$

El ángulo de giro ω que interviene téngase en cuenta que es de sentido antihorario por ser un giro matemático y no topográfico.

Si reparametrizamos del siguiente modo:

$$a = \lambda_{columnas} \cdot \cos\omega$$

$$b = \lambda_{columnas} \cdot \operatorname{sen}\omega$$

$$c = \lambda_{filas} \cdot \cos\omega$$

$$d = \lambda_{filas} \cdot \operatorname{sen}\omega$$

$$e = columna_T$$

$$f = fila_T$$

El sistema puede ser representado de forma matricial como:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} columna & fila & 0 & 0 & 1 & 0 \\ 0 & 0 & columna & fila & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}$$

Es un sistema que no está ajustado ya que tenemos 2 ecuaciones y 6 incógnitas, siendo necesarios por tanto 3 puntos para que este ajustado y alguno más para tener redundancia.

Como el sistema es de la forma:

$$A \cdot X = L$$

Es fácil resolver el sistema por mínimos cuadrados:

$$X = [A^T \cdot A]^{-1} \cdot A^T \cdot L$$

Con esto las incógnitas son los parámetros que nos permiten multiplicar posteriormente por la matriz de diseño A y así obtener fotocoordenadas.

Para el caso del cálculo inverso, podemos aplicar el mismo sistema matricial cambiando fotocoordenadas por coordenadas pixel con lo que se consiguen parámetros inversos denotados con un apostrofe.

$$\begin{bmatrix} columna \\ fila \end{bmatrix} = \begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a' \\ b' \\ c' \\ d' \\ e' \\ f' \end{bmatrix}$$

De nuevo se resolvería por mínimos cuadrados este sistema y así se obtendrían los parámetros inversos con al menos 3 puntos en ambos sistemas para estar ajustado y más en el caso de tener redundancia.

En este caso los parámetros son:

$$a' = \lambda_x \cdot \cos \omega$$

$$b' = \lambda_x \cdot \operatorname{sen} \omega$$

$$c' = \lambda_y \cdot \cos \omega$$

$$d' = \lambda_y \cdot \operatorname{sen} \omega$$

$$e' = x_T$$

$$f' = y_T$$

2.5. Metodología de la orientación externa

La orientación externa nos relaciona las fotocoordenadas de la imagen con los puntos del terreno. Se relacionan dos sistemas haciendo uso de colinealidad donde intervienen como incógnitas las coordenadas del punto de vista y los giros en el espacio.

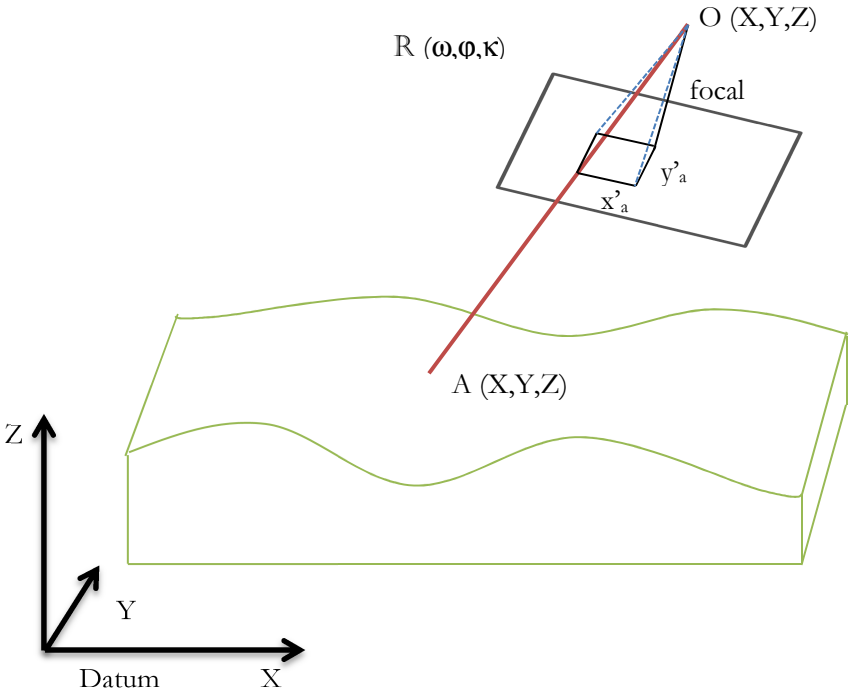


Figura 2-4. Orientación externa

La forma de expresar colinelidad matricialmente es:

$$\begin{bmatrix} x'_a \\ y'_a \\ -f \end{bmatrix} = \lambda \cdot \mathbb{R} \cdot + \begin{bmatrix} X_A - X_{O1} \\ Y_A - Y_{O1} \\ Z_A - Z_{O1} \end{bmatrix}$$

Reorgranizando el sistema y dividiendo la primera y segunda ecuación que se forman entre la tercera, somos capaces de formar una ecuación para las fotocoordenadas que no dependen del factor de escala pero si de los giros, de las coordenadas de los puntos de vista y de los puntos de apoyo.

$$x'_a = -f \frac{r_{11} \cdot (X_A - X_O) + r_{12} \cdot (Y_A - Y_O) + r_{13} \cdot (Z_A - Z_O)}{r_{31} \cdot (X_A - X_O) + r_{32} \cdot (Y_A - Y_O) + r_{33} \cdot (Z_A - Z_O)}$$

$$y'_a = -f \frac{r_{21} \cdot (X_A - X_O) + r_{22} \cdot (Y_A - Y_O) + r_{23} \cdot (Z_A - Z_O)}{r_{31} \cdot (X_A - X_O) + r_{32} \cdot (Y_A - Y_O) + r_{33} \cdot (Z_A - Z_O)}$$

Si expresamos las anteriores ecuaciones como se sigue:

$$F \approx x'_a \cdot [r_{31} \cdot (X_A - X_O) + r_{32} \cdot (Y_A - Y_O) + r_{33} \cdot (Z_A - Z_O)] + f \cdot [r_{11} \cdot (X_A - X_O) + r_{12} \cdot (Y_A - Y_O) + r_{13} \cdot (Z_A - Z_O)] = 0$$

$$G \approx y'_a \cdot [r_{31} \cdot (X_A - X_O) + r_{32} \cdot (Y_A - Y_O) + r_{33} \cdot (Z_A - Z_O)] + f \cdot [r_{21} \cdot (X_A - X_O) + r_{22} \cdot (Y_A - Y_O) + r_{23} \cdot (Z_A - Z_O)] = 0$$

Para resolver este sistema es necesario linealizar y realizar un cálculo iterativo mediante el desarrollo en serie de Taylor:

$$F \approx F_0 + \left(\frac{\partial F}{\partial X_O}\right) \cdot dX_O + \left(\frac{\partial F}{\partial Y_O}\right) \cdot dY_O + \left(\frac{\partial F}{\partial Z_O}\right) \cdot dZ_O + \left(\frac{\partial F}{\partial \omega}\right) \cdot d\omega + \left(\frac{\partial F}{\partial \varphi}\right) \cdot d\varphi + \left(\frac{\partial F}{\partial \kappa}\right) \cdot d\kappa$$

$$G \approx G_0 + \left(\frac{\partial G}{\partial X_O}\right) \cdot dX_O + \left(\frac{\partial G}{\partial Y_O}\right) \cdot dY_O + \left(\frac{\partial G}{\partial Z_O}\right) \cdot dZ_O + \left(\frac{\partial G}{\partial \omega}\right) \cdot d\omega + \left(\frac{\partial G}{\partial \varphi}\right) \cdot d\varphi + \left(\frac{\partial G}{\partial \kappa}\right) \cdot d\kappa$$

Para el valor F_0 y G_0 se toman valores aproximados de las incógnitas $X_O, Y_O, Z_O, \omega, \varphi, \kappa$. Estos valores aproximados pueden ser los valores nulos o para el caso de los puntos de vista el centroide de los puntos de apoyo.

Expresado matricialmente tenemos que calcular de forma iterativa soluciones hasta que esta converge:

$$\begin{bmatrix} \left(\frac{\partial F}{\partial X_O}\right)_0 & \left(\frac{\partial F}{\partial Y_O}\right)_0 & \left(\frac{\partial F}{\partial Z_O}\right)_0 & \left(\frac{\partial F}{\partial \omega}\right)_0 & \left(\frac{\partial F}{\partial \varphi}\right)_0 & \left(\frac{\partial F}{\partial \kappa}\right)_0 \\ \left(\frac{\partial G}{\partial X_O}\right)_0 & \left(\frac{\partial G}{\partial Y_O}\right)_0 & \left(\frac{\partial G}{\partial Z_O}\right)_0 & \left(\frac{\partial G}{\partial \omega}\right)_0 & \left(\frac{\partial G}{\partial \varphi}\right)_0 & \left(\frac{\partial G}{\partial \kappa}\right)_0 \end{bmatrix} \cdot \begin{bmatrix} dX_O \\ dY_O \\ dZ_O \\ d\omega \\ d\varphi \\ d\kappa \end{bmatrix} + \begin{bmatrix} F_0 \\ G_0 \end{bmatrix} = 0$$

Como tenemos 6 incógnitas y 2 ecuaciones por punto es necesario un mínimo de 3 puntos para que el sistema esté ajustado y más para tener redundancia.

Como el sistema es de la forma:

$$A \cdot X = L$$

Es fácil resolver el sistema por mínimos cuadrados:

$$X = [A^T \cdot A]^{-1} \cdot A^T \cdot L$$

El subíndice de las derivadas parciales se debe a que no se ha iterado aun. Así, tras un primer cálculo los valores obtenidos se actualizan como valores aproximados.

La forma de actualizar los valores es sumar a cada valor aproximado las correcciones correspondientes: d$X_O, dY_O, dZ_O, d\omega, d\varphi, d\kappa$

2.6. Restitución de puntos: Cálculo de coordenadas objeto

Acorde a la condición de colinealidad, las coordenadas modelo o terreno de un punto A quedan recogidas a través de la siguiente expresión:

$$\begin{bmatrix} X_A \\ Y_A \\ Z_A \end{bmatrix} = \lambda \cdot \mathbb{R} \cdot \begin{bmatrix} x'_a \\ y'_a \\ -f \end{bmatrix} + \begin{bmatrix} X_O \\ Y_O \\ Z_O \end{bmatrix}$$

donde el punto en cuestión depende de dos sumandos. El primero de ellos incorpora las fotocoordenadas del punto imagen, multiplicadas por una matriz de rotación que permiten orientar el fotovector a las condiciones de la toma, y que multiplicado por un escalar, permiten transformar sus dimensiones bien a la escala del terreno.

El segundo sumando permite fijar el origen del fotovector escalado y orientado, en el punto de vista del proyector. Esa traslación del vector, permite obtener las coordenadas tridimensionales del punto A terreno, siempre y cuando se conozca la orientación (matriz de rotación) y la posición (matriz de traslación) del fotograma.

Además, la expresión anterior presenta el inconveniente de que λ_1 también es desconocido. El factor de escala es único para cada punto a restituir, y por lo tanto inicialmente es una incógnita. Reparametrizando el sistema, para localizar todas las incógnitas en un mismo vector, se obtiene:

$$\begin{bmatrix} X_A \\ Y_A \\ Z_A \end{bmatrix} = \lambda_1 \cdot \mathrm{R}_1 \cdot \begin{bmatrix} x'_a \\ y'_a \\ -f \end{bmatrix} + \begin{bmatrix} X_{O1} \\ Y_{O1} \\ Z_{O1} \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 & -\mathrm{a_x} \\ 0 & 1 & 0 & -\mathrm{a_y} \\ 0 & 0 & 1 & -\mathrm{a_z} \end{bmatrix} \cdot \begin{bmatrix} X_A \\ Y_A \\ Z_A \\ \lambda_1 \end{bmatrix} = \begin{bmatrix} X_{O1} \\ Y_{O1} \\ Z_{O1} \end{bmatrix}$$

Se da la paradoja de disponer de tres ecuaciones para resolver cuatro incógnitas lo cual hace el sistema indeterminado. En el caso de disponer de otro fotograma en el que también aparezca la imagen del punto A, se plantearía una ecuación del mismo tipo

$$\begin{bmatrix} X_A \\ Y_A \\ Z_A \end{bmatrix} = \lambda_2 \cdot \mathrm{R}_2 \cdot \begin{bmatrix} x'_a \\ y'_a \\ -f \end{bmatrix} + \begin{bmatrix} X_{O2} \\ Y_{O2} \\ Z_{O2} \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 & -\mathrm{c_x} \\ 0 & 1 & 0 & -\mathrm{c_y} \\ 0 & 0 & 1 & -\mathrm{c_z} \end{bmatrix} \cdot \begin{bmatrix} X_A \\ Y_A \\ Z_A \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} X_{O2} \\ Y_{O2} \\ Z_{O2} \end{bmatrix}$$

que aporta otras tres ecuaciones con la adicción de una única incógnita más.

Fusionando ambas expresiones se construye un sistema con seis ecuaciones y cinco incógnitas que permite calcular las coordenadas tridimensionales de A, así como los factores de escala correspondientes.

$$\begin{bmatrix} 1 & 0 & 0 & -a_x & 0 \\ 0 & 1 & 0 & -a_y & 0 \\ 0 & 0 & 1 & -a_z & 0 \\ 1 & 0 & 0 & 0 & -c_x \\ 0 & 1 & 0 & 0 & -c_y \\ 0 & 0 & 1 & 0 & -c_z \end{bmatrix} \begin{bmatrix} X_A \\ Y_A \\ Z_A \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} X_{O1} \\ Y_{O1} \\ Z_{O1} \\ X_{O2} \\ Y_{O2} \\ Z_{O2} \end{bmatrix}$$

Donde:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = R_1 \begin{bmatrix} x_1 \\ y_1 \\ -f \end{bmatrix}$$

Siendo x_1, y_1 las fotocoordenadas en la imagen 1 y R_1 la matriz de rotación. Del mismo modo se establece que:

$$\begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} = R_2 \begin{bmatrix} x_2 \\ y_2 \\ -f \end{bmatrix}$$

Para aerotriangulación, vuelos con un recubrimiento mayor del tradicional 60%, o en fotogrametría convergente, es posible utilizar una expresión más generalista.

La expresión general para la restitución con n fotogramas es simple: incorporar tres ecuaciones por fotograma como dato, y un factor de escala como incógnita.

A modo de ejemplo, un punto en el terreno que tuviera imagen en tres fotogramas: dispondría de tres ecuaciones en cada fotograma, lo que resultaría en un total de nueve ecuaciones; y se resolverían seis incógnitas. Tres de ellas relativas a las coordenadas tridimensionales del punto en cuestión, y las otras tres consecuencia de los factores de escala para cada rayo proyectivo:

$$\begin{bmatrix} 1 & 0 & 0 & -a_x & 0 & 0 & 0 \\ 0 & 1 & 0 & -a_y & 0 & 0 & 0 \\ 0 & 0 & 1 & -a_z & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -c_x & 0 & 0 \\ 0 & 1 & 0 & 0 & -c_y & 0 & 0 \\ 0 & 0 & 1 & 0 & -c_z & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & -d_x & 0 \\ 0 & 1 & 0 & 0 & 0 & -d_y & 0 \\ 0 & 0 & 1 & 0 & 0 & -d_z & 0 \end{bmatrix} \begin{bmatrix} X_A \\ Y_A \\ Z_A \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} X_{O1} \\ Y_{O1} \\ Z_{O1} \\ X_{O2} \\ Y_{O2} \\ Z_{O2} \\ X_{O3} \\ Y_{O3} \\ Z_{O3} \end{bmatrix}$$

2.7. Generación de ortofotos

Se distinguen dos métodos para generar ortofotos disponiendo de los parámetros de la orientación externa de las fotografías correspondientes y de un MDE:

Método ascendente: En este caso para cada punto del MDE se obtendrá un nivel digital de la fotografía. Se hace uso de colinealidad y se denomina ascendente porque se pasa del MDE situado por debajo de la fotografía a esta misma. El resultado es una ortofoto del mismo tamaño que el MDE cuando el tamaño del MDE es inferior al de la imagen.

Sea un rayo proyectivo que pasa por un punto del MDE, corta a la fotografía y llega al centro de proyección, se trata de conseguir el nivel digital que tiene la fotografía en ese punto de corte:

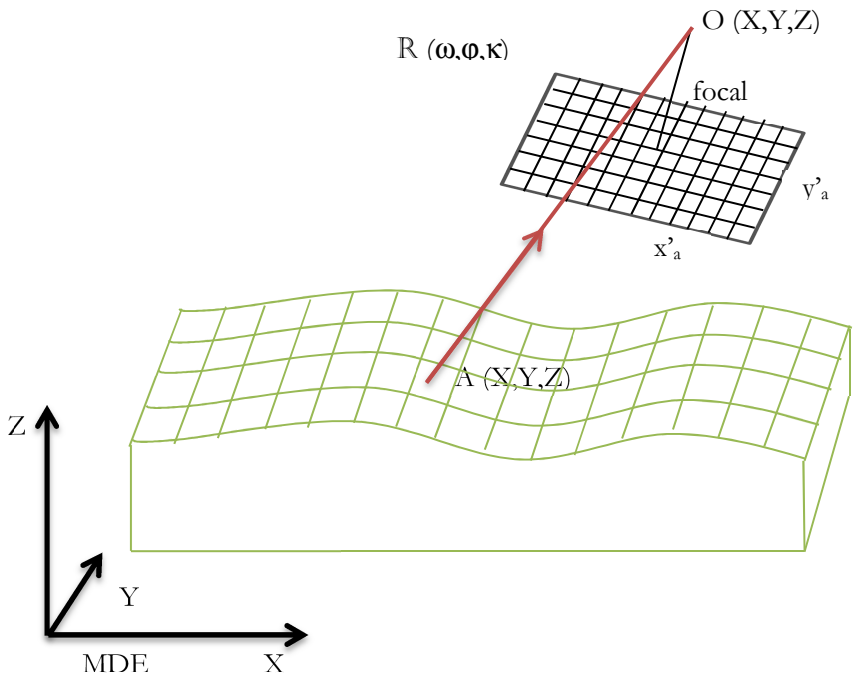
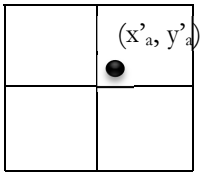


Figura 2-5. Método ascendente

Al trabajar en un espacio discreto puede darse el caso de que al situarnos en el centro de una celda del MDE y prolongar el rayo proyectivo que pasa por el centro de proyección este no corte en el centro de un píxel o celda imagen.

En este caso tenemos dos opciones para asignar un nivel digital a la ortofoto:



- Tomar el valor del vecino más próximo
- Realizar una interpolación bilineal o algún cálculo que pondere en función de la distancia al centro del pixel

Figura 2-6. Corte en el método ascendente

Tan solo debe aplicarse la ecuación que nos permite obtener fotocoordenadas en función de los puntos de apoyo y los parámetros de la orientación externa, y una vez obtenidas las fotocoordenadas, pasar a coordenadas píxel con una orientación interna en sentido inverso para recoger el nivel digital de la celda.

Esta tarea se realiza para cada punto del MDE tomando valores discretos.

Método descendente: En este caso para cada píxel o celda de la imagen obtendremos un valor de cota. Se hace uso de nuevo de colinealidad y es descendente por pasar de la imagen al MDE.

El tamaño de la ortofoto resultante no coincide con el tamaño del MDE ni con el de la imagen, ya que pueden darse situaciones en las que el rayo proyectivo no corta al MDE por quedar fuera la zona imagen de este, cuando el MDE, es de tamaño inferior a la imagen.

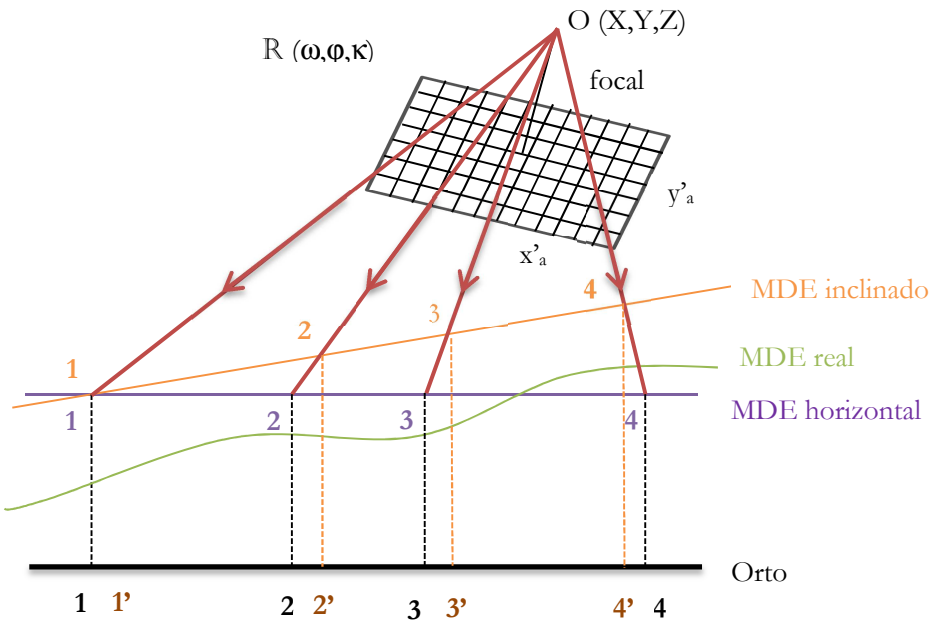


Figura 2-7. Método descendente

Según la figura 2-7 vemos como en función del MDE los puntos se proyectan de distinta forma en la ortofoto, véase diferencias en posición entre los puntos de un MDE horizontal y uno inclinado con sus proyecciones en la ortofoto.

Esto produce que el resultado, sea una nube de puntos no regular. Como la ortofoto tiene que ser un conjunto de celdas imagen este método implica interpolar cotas para obtener una malla regular.

2.8. El Modelo Digital de Elevaciones

Según Felicísimo (1994), un modelo digital de elevaciones (MDE) se define como una estructura numérica de datos que representa la distribución espacial de la altitud de la superficie del terreno. Un MDE puede describirse de forma genérica del modo siguiente:

$$z = f(x, y)$$

, donde z es la altitud del punto situado en la posición de coordenadas planimetricas x e y. Se trata pues, de una, función que relaciona la variable altura con respecto a su localización espacial. Los valores de x e y suelen corresponder con las abscisas y ordenadas de un sistema de coordenadas plano, habitualmente un sistema de proyección cartográfica.

La ecuación anterior representa una superficie o campo escalar en la que la altitud o cota es una variable continua. Dado que esta superficie está formada por un número infinito de puntos no es posible su modelización sin cierta pérdida de información, proceso equivalente al de generalización cartográfica en los mapas convencionales.

2.8.1. Estructuras de un MDE

De forma general, la unidad básica de información en un MDE es un valor de altitud, z, al que acompañan los valores correspondientes de x e y, expresados en un sistema de proyección geográfica. Cuando se definen las interrelaciones entre estas unidades elementales de información nos encontramos con varias variantes.

La clasificación general se divide en dos tipos fundamentales: estructura vectorial y ráster.

Los modelos vectoriales están basados en entidades (básicamente puntos y líneas) definidas por sus coordenadas. En los modelos ráster, los datos se interpretan como el valor medio de unidades elementales de superficie no nula que teselan el terreno con una distribución regular, sin solapamiento y con recubrimiento total del área representada.

Felicísimo (1994) establece la siguiente clasificación:

- Modelo vectorial: contornos: La estructura básica es el vector, compuesto por un conjunto de pares de coordenadas (x, y) que escribe la trayectoria de líneas isométricas (coincidiendo, por tanto, con las curvas de nivel o isohipsas del mapa topográfico convencional). El número de elementos de cada vector es variable y la reducción de éste a un único elemento permite incorporar cotas puntuales sin introducir incoherencias estructurales.

Una curva de nivel concreta queda definida, por tanto, mediante un vector ordenado de puntos que se sitúan sobre ella a intervalos adecuados (no necesariamente iguales) para garantizar la exactitud necesaria del modelo. La localización espacial de cada elemento es explícita, conservando los valores individuales de coordenadas. En el caso más sencillo, el MDE está constituido por el conjunto de las curvas de nivel que pasan por la zona representada, separadas generalmente por intervalos constantes de altitud.

Algunas opciones más avanzadas introducen líneas de rotura (*breaklines*), que permiten una mejor adaptación a algunos elementos del relieve (fallas, taludes), y que facilitan los tratamientos que necesitan asegurar la conectividad hidrológica.

- Modelo vectorial: redes de triángulos irregulares (TIN): Se compone de un conjunto de triángulos irregulares adosados, suele identificarse por las siglas de su denominación inglesa: triangulated irregular network, TIN (Peucker et al., 1978). Los triángulos se construyen ajustando un plano a tres puntos cercanos no colineales, y se adosan sobre el terreno formando un mosaico que puede adaptarse a la superficie con diferente grado de detalle, en función de la complejidad del relieve. Se trata de una estructura en la que el terreno queda representado por el conjunto de superficies planas que se ajustan a una estructura anterior de puntos.

  Los TIN pueden considerarse como una estructura derivada de otra anterior de puntos o líneas. Aunque la distribución original puede ser cualquiera (incluso puntos distribuidos aleatoriamente), es frecuente partir de una base de isohipsas (modelo vectorial) para generar la red de triángulos. Estos están internamente organizados en función de su vecindad mediante un conjunto de información bastante complejo que hace posible un manejo relativamente ágil y eficaz frente a alternativas menos estructuradas.

- Modelo ráster: matrices regulares: Esta estructura es el resultado de superponer una retícula sobre el terreno y extraer la altitud media de cada celda (aunque habitualmente se utiliza un valor puntual, asociado a cada nudo de la retícula o punto medio de la celda, con lo que esencialmente se construye un modelo vectorial de puntos). La retícula puede adoptar formas variadas pero la más utilizada es una red regular de malla cuadrada con filas y columnas equiespaciadas. En esta estructura, la localización espacial de cada dato está implícitamente determinada por su situación en la matriz, una vez definidos su origen y el intervalo entre filas y columnas. Las matrices de altitudes suelen ser generadas por interpolación a partir de un modelo previo de contornos o por métodos fotogramétricos.

Además de estas estructuras podemos encontrar más tipos en distintos libros y publicaciones siendo estas las más básicas.

Las alternativas prácticas se reducen básicamente a dos: matrices regulares y TIN. El modelo de contornos presenta serias dificultades de tratamiento directo; la consecuencia ha sido que, en el caso de los MDE, puede considerarse en la práctica como una estructura aceptable para la captación de información, pero no funcional para el tratamiento de los datos topográficos.

Podemos establecer que al elegir una estructura de MDE en general, la descripción vectorial es más adecuada para variables discretas, que por su naturaleza están limitadas por fronteras claras, mientras que las descripciones raster se adaptan mejor para la representación espacial de variables continuas, para las que no se pueden definir bordes de una forma neta, así como para otras con una naturaleza estadística o probabilística.

2.8.2. Procedimientos de obtención

Actualmente el procedimiento de obtención es el LiDAR aerotransportado, (un acrónimo del inglés Light Detection and Ranging o Laser Imaging Detection and Ranging), el cual forma parte de un sistema de altimetría laser.

Los sistemas de altimetría láser actuales se pueden considerar que son la unión de tres tecnologías: los sistemas de distanciometría láser (LiDAR), los sistemas de posicionamiento global mediante satélite (GPS) y los sistemas de referencia inercial de alta precisión (IMU).

Dichos sistemas requieren la utilización de un avión similar al empleado para la toma de fotogramas aéreos. Así mismo, comparte con la fotogrametría el diseño del plan de vuelo ejecutando pasadas paralelas y perpendiculares, siendo básico un buen diseño para el éxito de la operación. Un ejemplo de sensor digital aerotransportado es el LH ADS40.

El origen de un modelo digital de elevaciones puede estar entonces en la medida directa sobre la superficie real del terreno. Sea este el caso de los levantamientos de topografía clásica o de los sistemas de altimetría laser.

También es frecuente, sin embargo, el uso de métodos indirectos, que utilizan como base un conjunto de documentos (analógicos o digitales) elaborados previamente. Este sería el caso de obtener un MDE por procedimientos de fotogrametría o teledetección.

Atendiendo a la siguiente clasificación (Felicísimo, 1994):

Directos	Altimetría	Altímetros transportados por plataformas aéreas
	GPS	Global positioning system (sistema de localización mediante satélites)
	Topografía	Mediante estaciones topográficas con salida digital
Indirectos	Restitución	Origen digital: Imágenes digitales captadas por satélites (p. ej. SPOT) con diferentes ángulos de visión
	Digitalización	Manual: mediante tableros digitalizadores
		Automática: mediante scanners

Tabla 2-2. Resumen procedimientos de obtención de un MDE. (Felicísimo, 1994)

- Métodos directos: altímetros, GPS y estaciones topográficas: Algunos satélites disponen de altímetros que permiten el registro directo de los datos altimétricos en formato digital. Su mayor ventaja reside en que se trata de un método de captación remota de información por lo que la toma de los datos no está limitada por la accesibilidad de la zona.

  Con el GPS (siglas de su denominación en inglés: global positioning system) se utilizan un conjunto de satélites de referencia y, mediante métodos de triangulación, permiten obtener valores de las tres coordenadas espaciales para un lugar localizado sobre la superficie terrestre. Presenta algunas limitaciones que reducen su utilidad. Entre ellas, las dos principales son la necesidad de acceder físicamente al lugar de medida y el tiempo relativamente elevado que se precisa para realizar una toma de datos fiable. A estas dos circunstancias, que impiden en la práctica la adquisición del enorme número de datos que componen un MDE, deben añadirse algunos problemas secundarios relativos a la necesidad de condiciones favorables para las medidas (acceso visual directo y simultáneo a un mínimo de cuatro satélites, poca cubierta vegetal sobre la antena receptora, necesidad de una segunda estación de apoyo en funcionamiento simultáneo, etc.). Estas limitaciones convierten al método GPS más en un recurso de apoyo que en el sistema básico de captación de datos.

  Finalmente, las estaciones topográficas pueden generar y almacenar los resultados de sus medidas en formato digital. Algunos sistemas de información geográfica incorporan utilidades que permiten el tratamiento e incorporación de los datos en este tipo de formatos. A pesar de su utilidad, el método tiene problemas similares al anterior ya que la recogida de información exige la presencia física sobre el terreno.

- Métodos indirectos: restitución fotogramétrica: Estos métodos no necesitan acceder físicamente a la totalidad de la zona de estudio y la generación de datos se hace de forma relativamente rápida, cuestión básica cuando el volumen de información es muy elevado.

  En las operaciones de restitución analógica se utiliza como documento básico un conjunto de pares estereoscópicos de imágenes de la zona a estudiar. El trabajo se basa en métodos fotogramétricos que, examinando puntos homólogos en los pares estereoscópicos, deducen de su paralaje las cotas de referencia necesarias para reconstruir la topografía.

  Actualmente existe software que permite realizar esta tarea de forma automática grabando directamente los resultados en un formato digital compatible con sistemas de información geográfica. En este caso, los pares estereoscópicos son sustituidos por un conjunto de imágenes digitales que son procesados por el software para, mediante un proceso iterativo de cálculo de correlaciones, identificar los puntos homólogos y obtener altitudes o cotas. El acceso al terreno es necesario para establecer un conjunto de puntos de apoyo que permitan fijar valores de altitud en una escala absoluta.

  El método se utiliza también en la elaboración de cartografía analógica y sólo se diferencia en el formato de la salida de los datos. En la cartografía convencional, la salida se realiza sobre un soporte físico estable (minuta), mientras que en el otro caso se graba directamente en un soporte informático. Ambos tipos de productos, sin embargo, no son incompatibles y pueden ser generados paralelamente.

  Los pares estereoscópicos han sido hasta hace unos años exclusivamente fotogramas aéreos, tomados por cámaras de gran formato desde aviones en vuelo a diferentes altitudes. Actualmente, a estas fotografías se han sumado las imágenes digitales tomadas por sensores pancromáticos transportados por satélite. Se incorpora la posibilidad de obtener imágenes estereoscópicas mediante variaciones en el ángulo de vista, programables desde tierra.

  Finalmente, cabe mencionar la construcción de MDE a partir de datos tomados por los radares de apertura sintética (SAR). La Interferometría con Radar de Apertura Sintética es una técnica que puede aplicarse en muchas áreas diferentes. Implica utilizar un radar para registrar dos o más imágenes de exactamente la misma área en diferentes puntos temporales. Al comparar las imágenes, es posible detectar cualquier cambio que pueda haber ocurrido durante ese período particular de tiempo. La interferometría se puede conseguir con un único satélite o usando dos que van uno detrás del otro en la misma órbita.

- Métodos indirectos: digitalización de mapas topográficos: Los métodos fotogramétricos son utilizados generalmente por organismos estatales o por empresas especializadas. El elevado coste de los aparatos necesarios para la restitución hace difícil que pequeños equipos puedan abordar la construcción de los MDE por esta vía. La opción alternativa es la digitalización de los mapas topográficos prexistentes.

Cualquiera de los métodos anteriormente expuestos es una elección razonable si los medios disponibles lo permiten.

2.8.3. Etapa analógica y analítica para generar cartografía por fotogrametría

En la etapa analógica se utilizaban restituidores analógicos, se basa en la utilización de aparatos de restitución ópticos o mecánicos, donde el operador realizaba la alineación de las imágenes para crear un modelo estereoscópico debidamente nivelado y escalado. Por otro lado la confección de mapas, con información planialtimétrica, se realizaba con el principio de la marca flotante o graficadoras basadas en este principio. Se caracteriza por la casi ausencia de procedimientos de cálculo.

La idea es quitar la paralaje en el eje Y para medir la paralaje en X que nos da la profundidad de los objetos que vemos estereoscópicamente. Esto conlleva realizar la orientación interna y la orientación relativa por algunos de los métodos conocidos como son el método del proyector izquierdo, del proyector derecho o dejando la base fija operando sobre los puntos de Von Gruber.

Finalizada la orientación relativa se procede a la orientación absoluta con lo que damos escala al modelo obtenido así como lo situamos en un sistema de referencia acorde.

La realización de la cartografía con estos instrumentos era una tarea muy laboriosa que implicaba ir eliminado paralaje en X a cada uno de los puntos a restituir para obtener coordenadas tridimensionales en un sistema de referencia y pasar estos a un plano por medio de un trazador o coordinatógrafo.

Algunos de los restituidores podemos verlos en la siguiente imagen perteneciente al laboratorio de fotogrametría de la Escuela Politécnica Superior de Ávila:

Figura 2-8. Restituidores del laboratorio de la Escuela Politécnica Superior de Ávila

En la etapa analítica los cálculos que se realizan pasan de ser manuales a ser calculados con el ordenador a partir de la orientación relativa en el caso de la etapa semianalítica y a partir de la orientación interna en el caso de la etapa analítica propiamente dicha.

Un restituidor asistido es un restituidor analógico en el que la mesa trazadora o coordinatógrafo, asociado a los movimientos de exploración del modelo, y que permite recoger sobre una minuta cartográfica las coordenadas terreno correspondientes al modelo ha quedado sustituida por un modelo matemático, soportado por un procesador informático, cuyos datos proceden de codificadores colocados a lo largo de los carros guía instrumentales que posibilitan dicha exploración. La conversión A/D (analógico digital) queda localizada al nivel de las Coordenadas Modelo en el Método General de la Fotogrametría. Con este restituidor el producto final se permite que sea en formato digital y la orientación absoluta se aborda en el contexto matemático.

En la fotogrametría analítica el operador "sólo" debe proceder a observar los datos de la Orientación Interna (fiduciales) y, tras ella, identificar (medir Fotocoordenadas) la posición de los puntos homólogos, tanto de paso como de apoyo (habitualmente, la configuración de Von Gruber).

La fase de cálculo consiste en que las Fotocoordenadas, accesibles tras la Orientación Interna de cada uno de los fotogramas, son procesadas para efectuar la Orientación Externa del bloque. La herramienta es la Condición de Colinealidad.

Aunque el instrumento por excelencia para trabajar en este contexto es el estereocomparador codificado (trabajando tanto on-line como off-line) las posibilidades operativas abiertas hacen que pueda emplearse también:

- Un monocomparador, pues la aportación del estereocompaador es "sólo" el empleo de la visión estereoscópica en la identificación de puntos homólogos. En la medida en que se empleen puntos preseñalizados o puntos naturales de buena definición geométrica dicha identificación puede ser también monoscópica.

- Un restituidor analítico, pues este tipo de instrumento incluye las prestaciones del estereocomparador (los codificadores) además de aportar las prestaciones correspondientes a los restituidores (servomotores que permiten la eliminación automática de paralaje vertical tras la resolución de la Orientación Relativa).

En ambas etapas la elaboración de cartografía y pretender realizar un modelo digital de elevaciones cuando se dispone de los medios necesarios es una tarea costosa que conlleva que el operador tome cada uno de los puntos que definen la malla del modelo.

Es un proceso muy lento ya que en función de la densidad de puntos que se quieran tomar se demorará más o menos tiempo. A, demás, es un proceso sometido a los errores y equivocaciones que el operador cometa en la identificación de puntos homólogos. Será ya en la etapa digital donde se pretende automatizar esta tarea para que el operador intervenga lo menos posible con el empleo de software adecuado.

2.8.4. Etapa digital para generar MDE por fotogrametría

En fotogrametría digital la conversión A/D (analógico-digital) se produce a nivel de fotograma, es decir, la orientación interna y la identificación de puntos homólogos se hace de forma numérica por medio de software en el ordenador.

En esta etapa los ordenadores son capaces de visualizar imágenes de gran tamaño y explorarlas a gran velocidad con las técnicas de roaming, es decir, imagen móvil y cursor fijo. Con las cámaras digitales ya no es necesario escanear las imágenes ya que estas se capturan en formato digital.

Se sustituye el restituidor por la Estación Fotogramétrica Digital, esto es un equipo informático específico compuesto por dispositivos de video, de visión estereoscópica, CPU general, periféricos de entrada/salida básicos, dispositivos de medición estereoscópica y software fotogramétrico.

Pero quizás, lo más importante de esta etapa, es la capacidad de automatización de los procesos fotogramétricos. Mientras que en las técnicas anteriores, se procesan pocos puntos bien seleccionados, en esta técnica se procesan muchos puntos pero de calidad no contrastada. Se establece así una dicotomía entre calidad vs cantidad de puntos. La potencia de la automatización radica en la posibilidad de manejar, no unas decenas de puntos por fotograma, sino cientos o miles de puntos por fotograma. La limitación radica en que dicha selección no seguirá los criterios de sentido común y experiencia que aporta un operador fotogramétrico y, en consecuencia, muchos de ellos serán de muy baja eficacia geométrica y, aún más, muchos de ellos serán errores groseros, es decir, equivocaciones.

Esta técnica debe incorporar, en consecuencia, estrategias que aseguren que los puntos seleccionados verifiquen unos mínimos de calidad así como filtros que eliminen las equivocaciones.

La elaboración de MDE's en esta etapa se considera por tanto adecuada ya que podremos desarrollar algoritmos que sean implementados en un lenguaje de programación así como crear una interfaz de usuario. De esta forma, un usuario poco especializado con ayuda de un manual del software o indicaciones previas puede ser capaz de generar un modelo digital de elevaciones de forma automática en un tiempo relativamente corto.

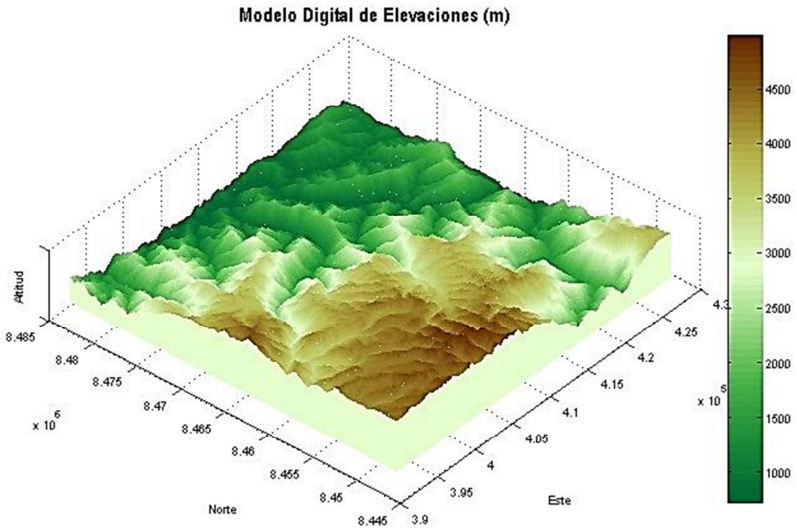


Figura 2-9. Ejemplo de MDE con vista en 3D (fuente: internet)

2.8.5. Aplicaciones del MDE

Vamos a ver a continuación algunos de los usos y aplicaciones que se puede dar al MDE generado. Los ejemplos que se muestran en las siguientes imágenes no han sido generados en este proyecto, son de otras fuentes.

Algunos autores mencionan que el terreno puede ser expresado mediante una superficie cuadrática de la forma:

$$z = \frac{rx^2}{2} + sxy + \frac{ty^2}{2} + px + qy + z_0$$

Al hacer que esta función se ajuste en la mayor medida posible a la superficie real del terreno tal y como ésta se encuentra recogida en el MDT con una menor o mayor precisión, resulta poco lógico, intuyéndose de antemano que la fidelidad para con el relieve real de tal función será poco menos que nula. No obstante, si en lugar de trabajar a nivel global con la totalidad de la malla lo hacemos a nivel local tomando el entorno inmediato de un punto, esta simplificación cobra mayor sentido y los parámetros que puedan calcularse a partir de la ecuación que se deduzca en dicha porción de la malla serán más representativos de las características reales del punto tomado.

Pendiente

Partiendo de la función matemática descrita anteriormente, y a la cual asimilábamos la forma del MDE en un entorno local de un punto dado, la pendiente puede calcularse a través de las primeras derivadas de dicha función.

Así la pendiente se define de la siguiente manera:

$$\nabla Z = \left(\frac{\partial z}{\partial x} , \frac{\partial z}{\partial y} \right), SLOPE = \arctan(|\nabla Z|)$$

La pendiente deseada representa el ángulo entre dicho vector gradiente y la vertical, el cual, haciendo uso de conceptos básicos de cálculo de ángulos entre vectores, se obtiene según la expresión

$$\gamma = \arctan\sqrt{p^2 + q^2}$$

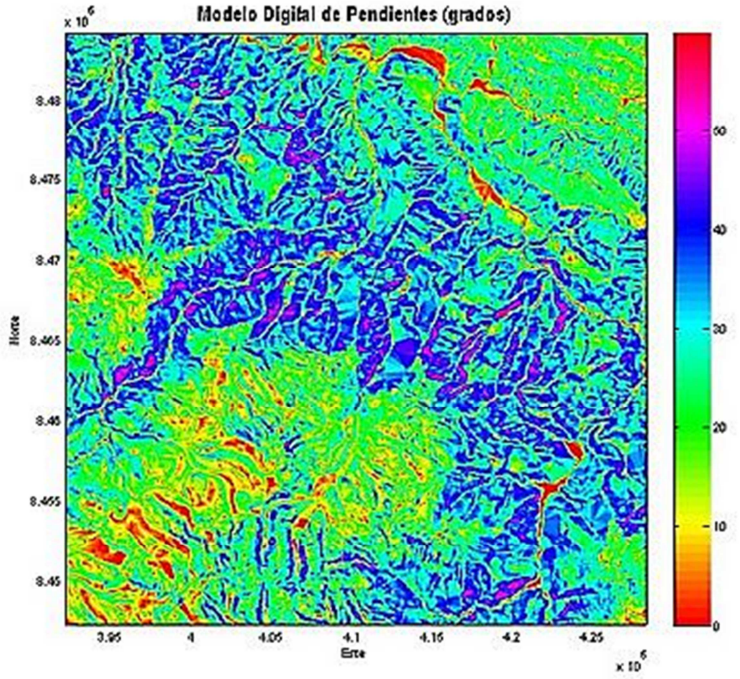


Figura 2-10. Modelo digital de pendientes

Orientación

La orientación en un punto puede definirse como el ángulo existente entre el vector que señala el norte y la proyección sobre el plano horizontal del vector gradiente (Felicísimo 1994).

La forma habitual de utilizar este parámetro es expresado en grados sexagesimales, de acuerdo a la siguiente expresión.

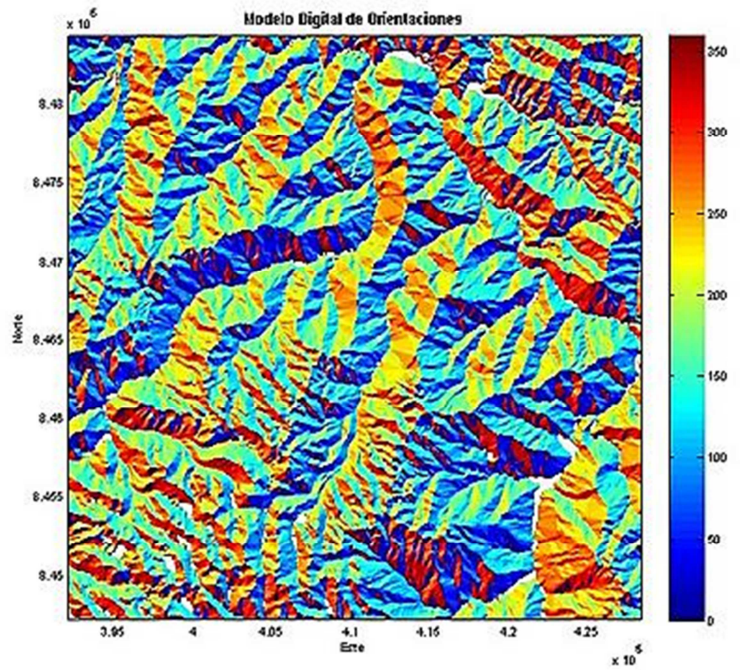


Figura 2-11. Modelo digital de orientaciones

Sombreado

El sombreado en una superficie se obtiene de una iluminación hipotética, mediante la determinación de los valores de iluminación para cada celda. Esto se logra mediante el establecimiento de una fuente de luz hipotética y el cálculo de los valores de iluminación de cada celda en relación a las celdas vecinas.

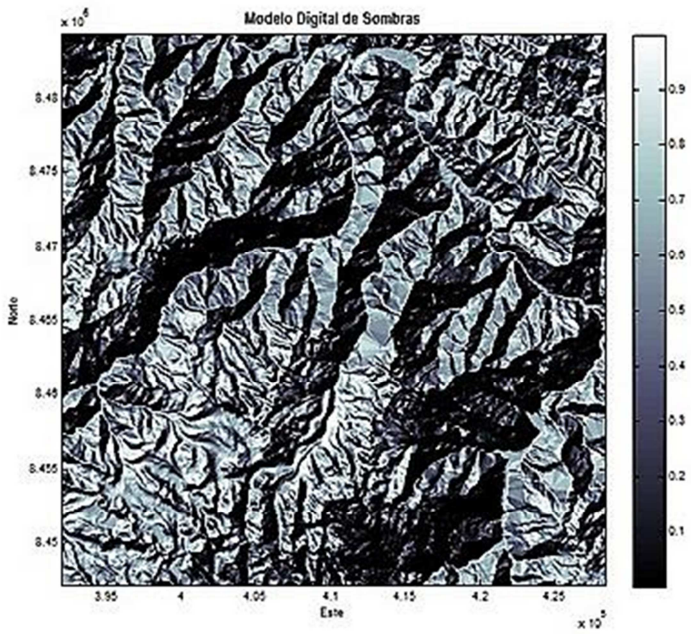


Figura 2-12. Modelo digital de sombreado

Rugosidad

Upton y Fingleton, citados por Felicísimo (1994) mencionan que las coordenadas rectangulares de un vector unitario perpendicular a la superficie en un punto i vienen dadas por las expresiones:

$$X_i = sen(\gamma_i)\cos(\phi_i)$$

$$Y_i = sen(\gamma_i)\mathrm{sen}(\phi_i)$$

$$Z_i = \cos(\gamma_i)$$

El módulo del vector suma de un conjunto de vectores es un indicador de agrupación y, por tanto, inversamente proporcional a la rugosidad. El módulo del vector suma se calcula, para un conjunto de n datos vecinos al punto problema (los 8 más próximos, por ejemplo) mediante la expresión:

$$R = \sqrt{\left(\sum x_i\right)^2 + \left(\sum y_i\right)^2 + \left(\sum z_i\right)^2}$$

Band, citado por Felicísimo (1994) menciona que resulta conveniente estandarizar el valor de R dividiéndolo por el tamaño muestral obteniendo así el módulo medio. El resultado puede variar entre los valores extremos de 0 (dispersión máxima) y 1 (alineamiento completo). El módulo medio es complementario del parámetro estadístico denominado varianza esférica.

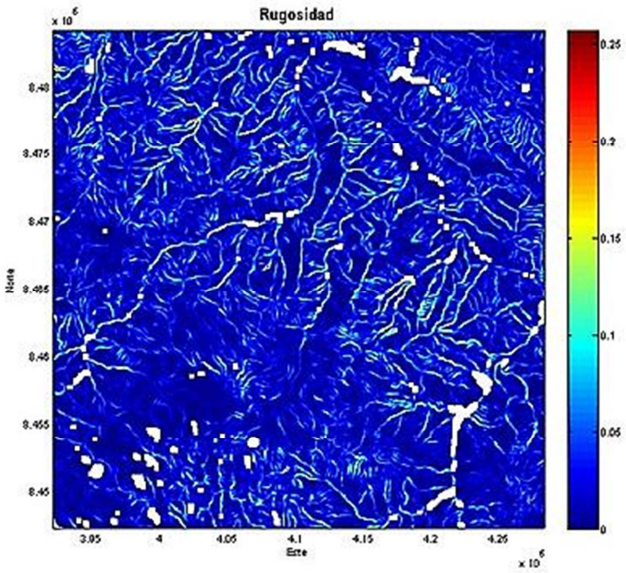


Figura 2-13. Modelo digital de rugosidad

Índice Elevación-Relieve

Se describe este índice según la siguiente expresión:

$$IR = (Zm - Zmin)(Zmax - Zmin)$$

Siendo Zm, Zmin, Zmax los valores medio, mínimo y máximo de elevación en la ventana de análisis 3x3 alrededor de cada celda.

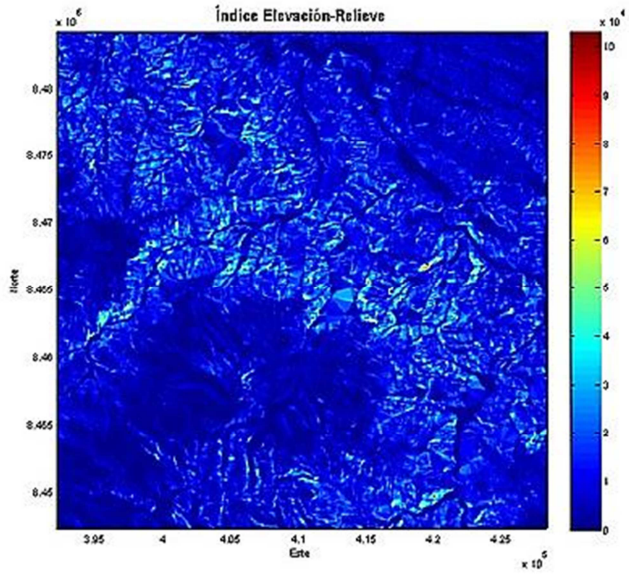


Figura 2-14. Índice de elevación-relieve

Índice Varianza Pendiente

Felicísimo (1994) describe este índice como el análisis mediante una ventana móvil sobre el modelo digital de pendientes. El cálculo de la varianza se realiza sobre los valores incluidos en ella y el resultado puede utilizarse como estimación de la rugosidad.

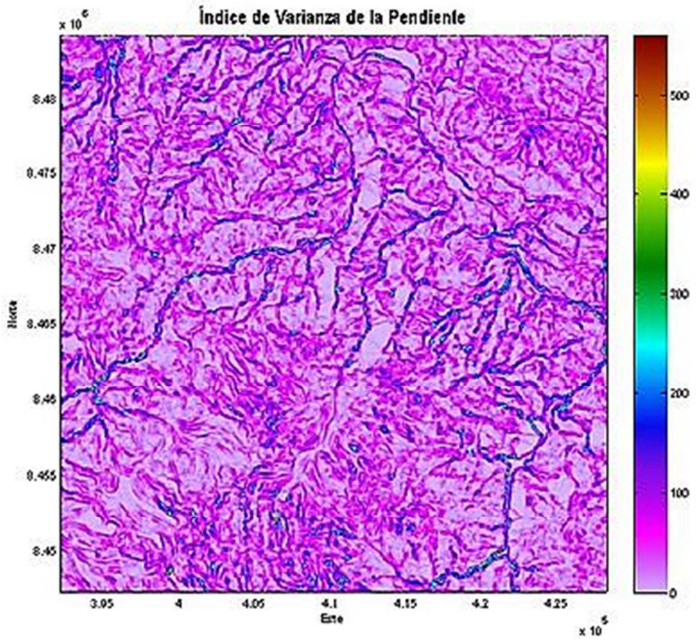


Figura 2-15. Índice de varianza pendiente

Curvatura

Felicísimo (1994) describe la curvatura en un punto, h, puede definirse como la tasa de cambio en la pendiente y depende, por tanto, de las derivadas de segundo grado de la altitud (es decir, de los cambios de pendiente en el entorno del punto).

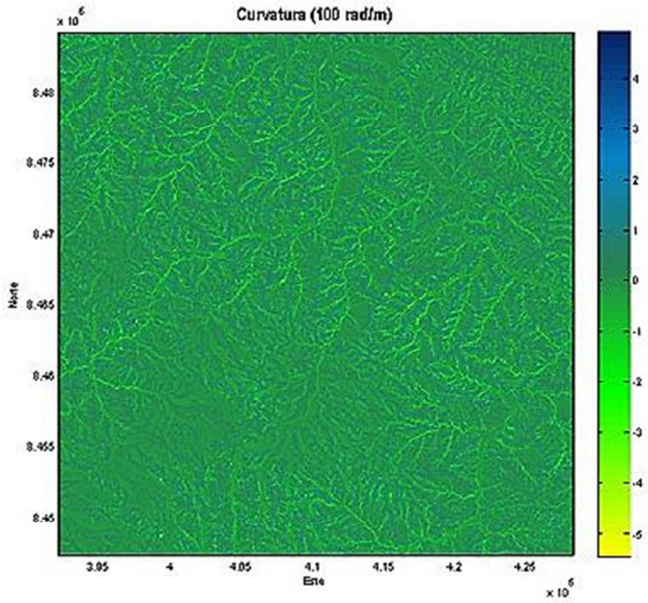


Figura 2-16. Curvatura

Acumulación de flujo

Cuando la proporción d de drenaje de una celda a sus vecinos (debe sumar 1) son conocidos, también la proporción de recepción r que drena hacia una celda. Las proporciones de recepción determinan que fracción de cada vecino son recibidas. La cantidad de masa (o volumen, área u otra propiedad) A que se acumula en la celda i está dada por la suma de A en cada celda vecina multiplicada por la respectiva fracción de recepción r más la masa u otra cualidad ingresante I en la celda misma.

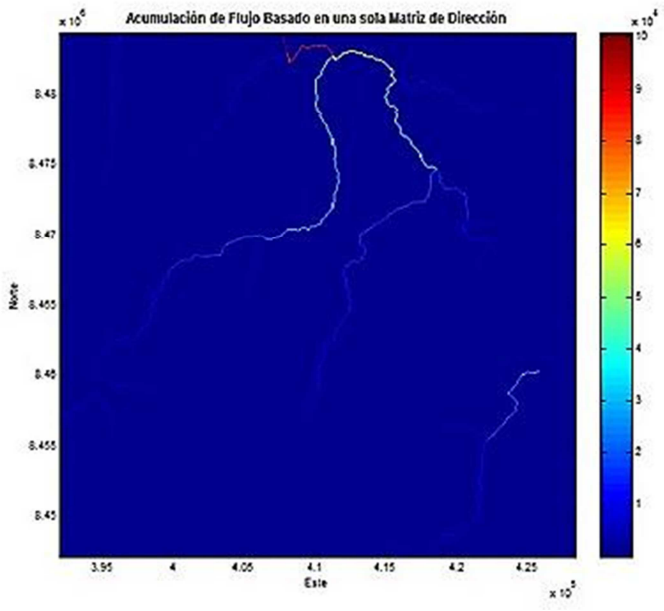


Figura 2-17. Flujo acumulado (Matriz simple)

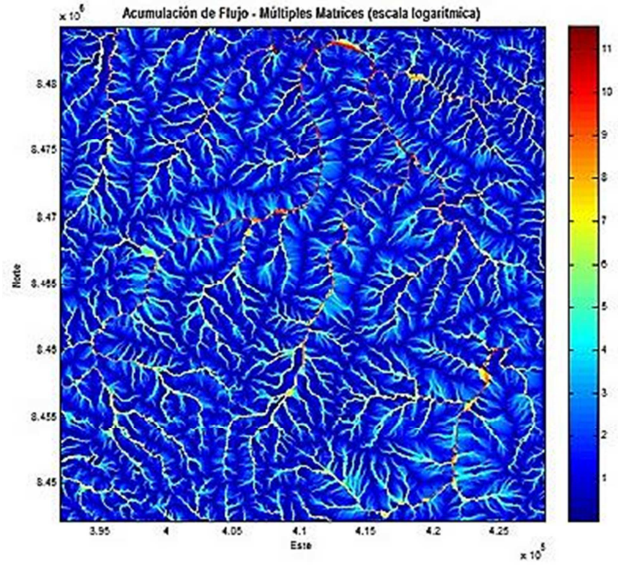


Figura 2-18. Flujo acumulado (Matriz múltiple)

Cuencas hidrográficas

La dirección de flujo es el elemento clave para la delimitación de las cuencas hidrográficas, es con el cual se da forma con toda precisión a las cuencas vertientes.

El conocimiento de esta dirección de flujo de cada celda nos permitirá deducir para cada una de ellas, y siguiendo el recorrido desde las mismas hacia aguas abajo, si pertenecen o no a la cuenca que pretendemos definir.

$$f_{CUE} : \mathrm{M} \times \mathrm{N} \longrightarrow \mathbb{N}$$

$$f_{CUE}(i,j) \begin{cases} = 0 & \text{si la celda no es una celda de cuenca} \\ \neq 0 & \text{si la celda es una celda de cuenca} \end{cases}$$

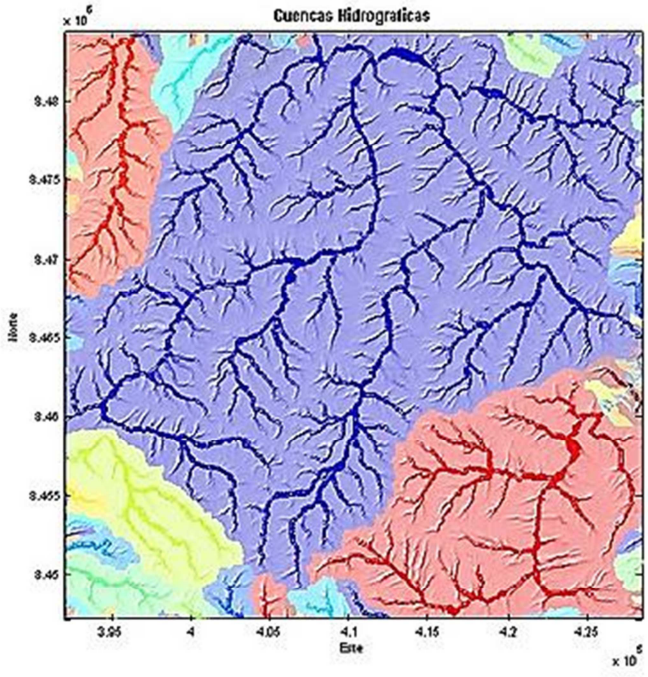


Figura 2-19. Cuencas hidrográficas

Índice de humedad

El índice de humedad para cada celda está representado mediante la siguiente expresión

$$weti = \log\left(\frac{1 + flowac}{\tan(slp)}\right)$$

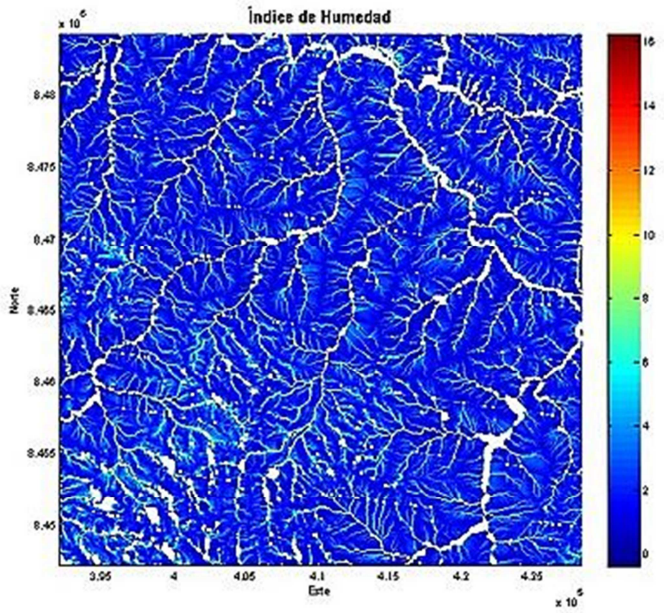


Figura 2-20. Índice de humedad

Índice de potencia de cauce

El índice de potencia de cauce es frecuentemente usado en geomorfología, ciencias del suelo y disciplinas relacionadas. Como una media de potencia de cauce es un indicador del potencial transporte de sedimentos y de erosión del agua. Se define mediante la siguiente expresión.

$$spi = flowac \cdot \tan(slp)$$

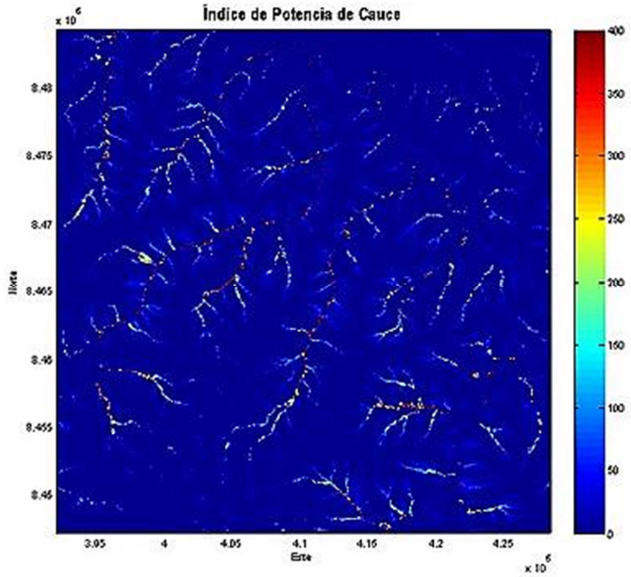


Figura 2-21. Índice de potencia de cauce

CAPÍTULO 3. INTRUMENTACIÓN

La instrumentación empleada esta compuesta principalmente por cámara fotográfica, un dispositivo denominado "CHISME" pudiéndose utilizar plantillas impresas con un sistema de coordenadas arbitrario, software de programación y de fotogrametría, una colección de imágenes y un equipo informático sobre el que ejecutar estos.

La cámara

La cámara con la que se han tomado fotos para los ejemplos realizados con "CHISME" es el modelo Panasonic GS200.

Sistema de lente	
Type	Zoom lens - 2.45 mm - 24.5 mm - f/1.8-2.8
Lens Aperture	F/1.8-2.8
Focal length [mm]	2.45 - 24.5 mm
Focal Length Equivalent to 35mm Camera [mm]	47.1 - 471 mm
Focus adjustment	Automatic, manual
Optical zoom	10 x
Filter size [mm]	37 mm
Lens manufacturer	Leica
Auto focus	TTL contrast detection
Zoom Adjustment	Electronic drive

Fotografías Digitales	
Grabación Fotos Digitales	Sí
Resolución Máxima [píxel]	1760x1320 píxel
Tipo Memoria Soportada	Secure Digital
Memoria Incluida [Mb]	8 Mb

Tabla 3-1. Especificaciones de la cámara

Las tomas se realizan sin uso de trípode a una altura más o menos similar entre ellas para que no se produzca variación de escala.

Al disponer de una lente Leica pensamos que la estabilidad de la misma puede ser buena, por otra parte vemos que la resolución es de 2.3 megapíxeles, que si bien, no es una gran resolución, para la distancia a la que se capturan las fotografías en los ejemplos mostrados, nos permite obtener un MDE de alta resolución.

Matlab

El software Matlab en su versión r2011a nos permite como entorno de desarrollo integrado implementar en este lenguaje los algoritmos y tareas que realizan el proyecto. Más información de su funcionamiento en el anexo sobre el Lenguaje de programación empleado Matlab.

OrthoEngine de PCI Geomatica

Este software en su versión 10, nos permite realizar algo similar a lo que se pretende en este proyecto con algunas diferencias en su funcionamiento.

El programa realiza la orientación interna y la orientación externa, pero, en la fase de elaboración del MDE realiza la tarea previa de construir imágenes epipolares, y sobre estas, se produce la correspondencia de puntos homólogos.

Dicho software es empleado para comparar resultados obtenidos, si bien, como consecuencia de no saber realmente como esta implementado el mismo y de emplear otra metodología es bastante probable que el resultado difiera del que hemos elaborado.

Para una mejor comparación y poder determinar cual software es más fiable habría sido recomendable utilizar un tercer software o más que nos permita discriminar. Sin embargo, se cree que al ser desconocidos en ellos sus implementaciones y algoritmos usados será difícil llevar a cabo una comparación a priori.

Por ello para examinar la calidad de los resultados se propone medir objetos regulares sobre el armazón o "CHASIS" y verificar diferencias.

Photomodeler Scanner

Con el programa Photomodeler Scanner en su Versión 6 calibramos la cámara.

Partimos de unos parámetros iniciales que se nos han proporcionado con otro software denominado BINGO, cuyos valores son:

$$f = 4.248\,mm$$

$$Ancho\ sensor = 3.382\,mm \qquad Alto\ sensor = 2.538\,mm$$

$$Desplazamiento\ del\ punto\ principal\ x = 0.003\,mm\ y = 0.032mm$$

A partir de estos, el programa tarda un tiempo en hacer los cálculos y finalizar con la tarea de calibración, finalmente nos muestra los resultados y nos permite imprimirlos a un fichero.

En el caso nuestro el informe es el que se recoge a continuación:

```
Precisions / Standard Deviations
      Camera Calibration Standard Deviations
         Camera1: Unnamed Default Camera
            Focal Length
               Value: 4.161390 mm
               Deviation: Focal: 0.001 mm
            Xp - principal point x
               Value: 1.734840 mm
               Deviation: Xp: 3.3e-004 mm
            Yp - principal point y
               Value: 1.210864 mm
               Deviation: Yp: 9.5e-004 mm
            Fw - format width
               Value: 3.384756 mm
               Deviation: Fw: 6.0e-005 mm
            Fh - format height
               Value: 2.538000 mm
            K1 - radial distortion 1
               Value: 1.505e-002
               Deviation: K1: 6.6e-005
            K2 - radial distortion 2
               Value: -5.948e-004
               Deviation: K2: 2.0e-005
            K3 - radial distortion 3
               Value: 0.000e+000
            P1 - decentering distortion 1
               Value: 4.646e-004
               Deviation: P1: 4.5e-006
            P2 - decentering distortion 2
               Value: -2.110e-004
               Deviation: P2: 1.2e-005
```

Figura 3-1. Informe de resultados

De aquí el valor que recogeremos será el de la focal de 4,16 mm redondeado con dos decimales y no introduciremos correcciones de distorsiones ni del punto principal, ya, que la relación tamaño sensor-tamaño imagen en pixeles nos da un tamaño de pixel muy pequeño. Veremos más adelante que el tamaño de GSD es inferior a 1 mm y no trabajaremos por debajo de este.

Equipo informático

El equipo utilizado para la elaboración del proyecto ha sido con plataforma de sistema operativo Windows en versión de 64 bits. El software se ha compilado tanto para la versión de 64 bits como para la versión de 32 bits.

Se obtiene mayor rendimiento del mismo con un procesador y tarjeta grafica potentes de ultima tecnología, estando estos realizados con procesador de 2,5 GHz y tarjeta grafica de 512 Mb así como una memoria RAM de 4 Gb.

CHISME

Para facilitar la elaboración de los modelos digitales de elevaciones con la aplicación se han hecho algunas pruebas con un dispositivo denominado CHISME (Chasis Imagen-Suelo para medidas elementales.

CHISME es un dispositivo de construcción sencilla formado por un marco rectangular de 60x60cm, realizado con láminas plano-paralelas de 5cm de ancho y 7mm de grosor. En cada una de las piezas laterales que conforman el marco, se ha dispuesto una plantilla de elementos de calibración del dispositivo: ocho dianas de puntería y ciertos puntos de enlace, permiten calibrar el chasis y los parámetros internos de la cámara. Álbumes cromáticos situados junto a las dianas anteriores permiten la calibración radiométrica de las imágenes adquiridas, facilitando la transformación entre los niveles de gris de la cámara y valores de radiancia absolutos. Finalmente, la incorporación de varios test ópticos de resolución, siguiendo una secuencia similar a códigos de barras, permite valorar la apreciación del píxel de la imagen adquirida acorde a las condiciones reales de la toma.

Las dianas cromáticas no han sido utilizadas en el presente proyecto pues no son necesarias para la obtención del MDE. Sí lo han sido los segmentos plano-paralelos que marcan la resolución de la cámara en las condiciones de medida con el chasis.

Las dimensiones del chasis lo hacen ideal para que todas las fotografías lo abarquen con un 100% de recubrimiento. Es importante que entre una y otra toma exista una distancia (base) que ronde aproximadamente 1/4 ó 1/5 de la altura de la cámara.

Nuestro marco de referencia empleado en los ejemplos esta formado por las dianas del armazón o “CHISME” o por una cuadrícula con coordenadas conocidas. En cualquier otro caso necesitaremos de puntos de apoyo.

Para el caso de "CHISME" se considera el sistema de coordenadas cartesiano local (x, y, z) definido por el plano del armazón con origen arbitrario y ejes perpendiculares con orientación en el sentido de los lados del mismo.

El eje de ordenadas, Y, es aquel que se ve vertical en todas las fotografías, y el eje de abscisas X, es el que se ve horizontal, haciendo todas las tomas con la misma orientación, siendo estas ligeramente convergentes.

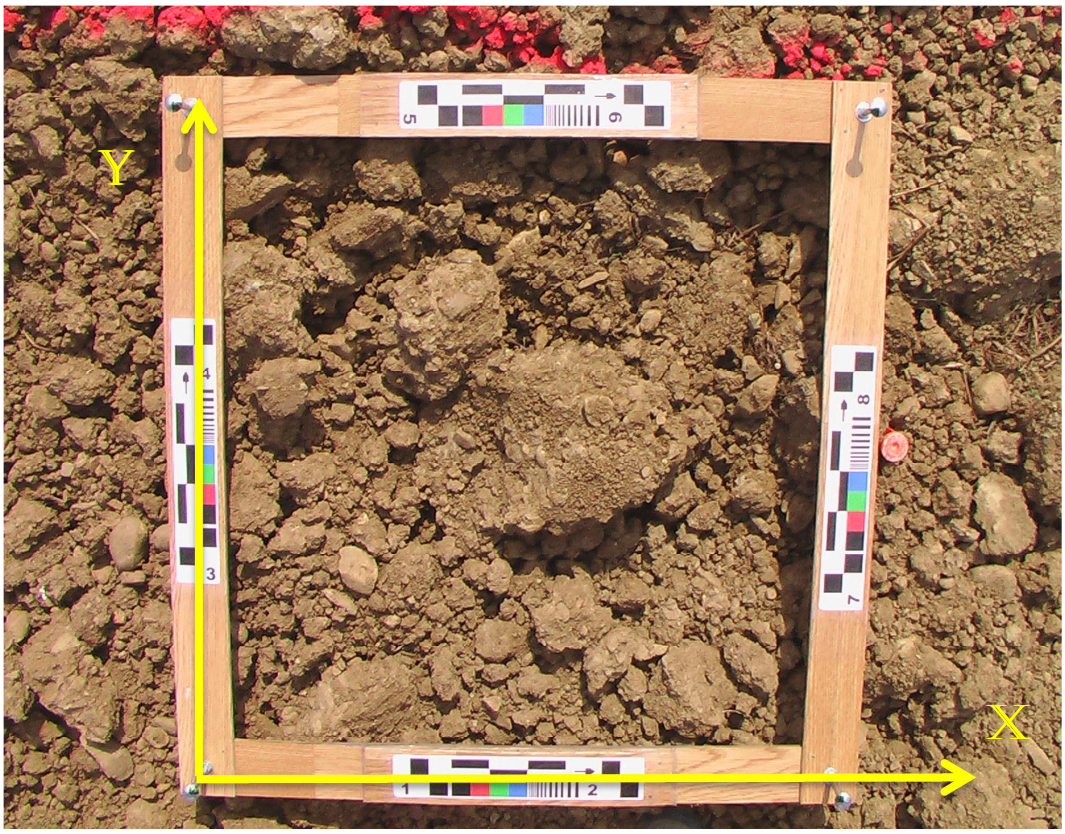


Figura 3-2. Sistema de coordenadas de CHISME

Para los ejemplos utilizados con la poca extensión de terreno abarcada resulta cómodo trabajar con sistema de coordenadas local y evitar el uso de un Datum o elipsoide, ya que para trabajos semejantes no hablamos de sistema de referencia propiamente geodésicos.

Aunque algunos ejemplos se realizan con este dispositivo "CHISME", también podemos realizar pruebas con otros elementos. Por ejemplo, sobre una malla imprimida en papel, en la que dotamos de coordenadas algunos de sus puntos para, posteriormente, colocar algunos objetos regulares sobre esta y realizar las correspondientes tomas fotográficas y procesarlas con el software.

Colección de imágenes

Se ha hecho uso de una colección de imágenes proporcionadas con el fin de obtener distintos MDE a modo de ejemplo.

Estas imágenes se corresponden con distintos suelos tomados con "CHISME" clasificados como rugosidad lisa, media y alta:

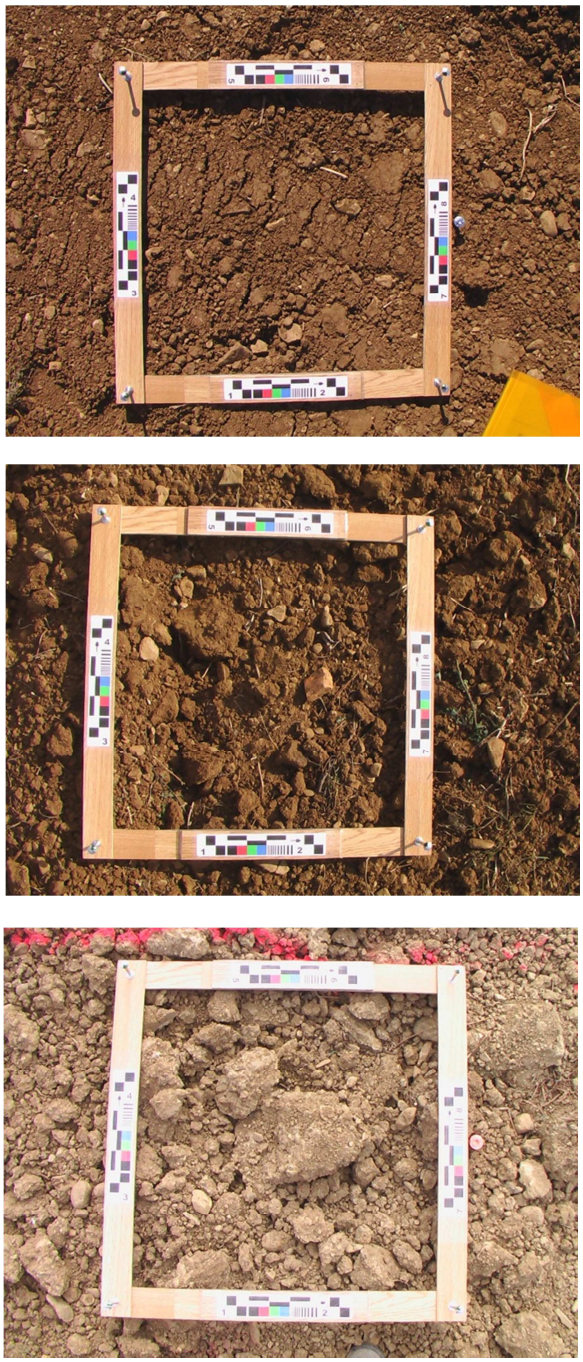


Figura 3-3. De arriba hacia abajo: zona lisa, media y rugosa

Las imágenes tienen una resolución de 1760x1320 píxeles considerando el ancho de 1760 píxeles y el alto de 1320 píxeles.

A parte de las imágenes que mostramos en la figura anterior tenemos 4 orientaciones de la misma foto teniendo en cuenta que estas son ligeramente convergentes.

Son ligeramente convergentes y, por ello, el giro de las mismas es bajo permaneciendo siempre los números de la diana situados en posiciones correspondientes de la imagen, esto es, si las dianas 1 y 2 están en la parte inferior de una imagen también lo están en el resto, al igual que si la diana 3 y 4 están a la izquierda también lo están en las otras:

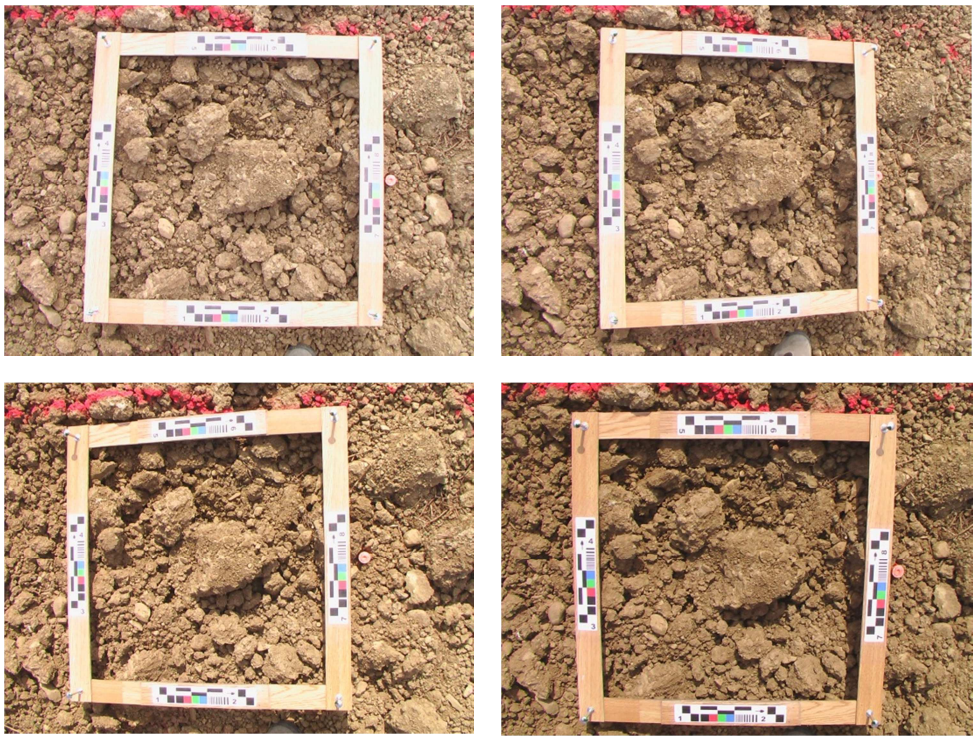


Figura 3-4. Distintas orientaciones de una misma zona rugosa

Para más información acerca de la huella del pixel o GSD, podemos calcular este para una fotografía y hacernos una idea de los valores con los que se van a trabajar en estos ejemplos.

Sea la primera fotografía de la figura anterior, la cual se ha tomado de forma más nadiral, tenemos que medir elementos en la realidad y en la imagen siendo factible hacerlo sobre las dianas puesto que tenemos sus coordenadas.

Por tanto, establecemos la relación entre los milímetros y píxeles para obtener el valor del GSD.

Las coordenadas de las dianas se nos han proporcionado, obteniendo los siguientes valores:

	X	Y	Z
1	1240	1000	1000
2	1439,971	999,839	1000
3	1004,334	1234,694	1000
4	1005,490	1434,673	1000
5	1249,069	1662,389	1000
6	1448,983	1663,998	1000
7	1662,904	1215,797	1000
8	1663,196	1415,763	1000

Tabla 3-2. Puntos de apoyo

Los valores vienen dados con tres decimales pero teniendo en cuenta que las unidades son en milímetros y no vamos a trabajar por debajo de estas podremos redondearlos al valor entero. No obstante, para el cálculo que se pretende, al ser aproximado no conlleva problemas usar los decimales.

Vamos a calcular las distancias por parejas 1-2, 3-4, 5-6 y 7-8:

$$D = \sqrt{\Delta x^2 - \Delta y^2}$$

Obtenemos los siguientes valores:

$$D_{12} = 199{,}9711$$

$$D_{34} = 199{,}9823$$

$$D_{56} = 199{,}9205$$

$$D_{78} = 199{,}9662$$

Redondeando estos valores tenemos 200 milímetros, es decir, 20 centímetros de separación por cada pareja de dianas.

Las coordenadas pixel de las dianas se han tomado con el comando $ginput$ de Matalb, obteniendo:

	Columna	Fila
1	714,145955451348	1161,88335287222
2	1031,37924970692	1166,52579132474
3	355,130715123095	785,845838218054
4	370,605509964830	479,444900351700
5	754,380422039859	143,641852286049
6	1057,68640093787	146,736811254396
7	1382,65709261430	830,722743259085
8	1381,10961313013	524,321805392731

Tabla 3-3. Coordenadas pixel

Calculamos distancias por parejas al igual que en las coordenadas terreno:

$$Dpixel_{12} = 317{,}2673$$

$$Dpixel_{34} = 306{,}0099$$

$$Dpixel_{34} = 303.3216$$

$$Dpixel_{34} = 306{,}4050$$

Para el cálculo del GSD establecemos la siguiente relación:

$$GSD = \frac{Distancia\ terreno}{Distancia\ imagen}$$

Obtenemos cuatro valores, una por imagen:

$$GSD1 = 0.6303, \quad GSD2 = 0.6535, \quad GSD3 = 0.6591, \quad GSD4 = 0.6526$$

Haciendo la media damos como valor aproximado:

$$GSD = 0.6490\ mm/pixel$$

Este es el valor medio que obtenemos para el tamaño de un pixel en la imagen seleccionada.

CAPÍTULO 4. EXPERIENCIA DE USUARIO

En este capítulo vamos a explicar el funcionamiento interno del software que se ha elaborado, dividiendo cada tarea a realizar en diferentes apartados, desde que arrancamos el programa y vemos la interfaz grafica de usuario de inicio, hasta que llegamos al resultado deseado.

Al mismo tiempo, se irán presentando los prototipos de las funciones que se han implementado.

4.1. Apertura del programa

El programa ha sido compilado para trabajar sobre el sistema operativo Microsoft Windows en cualquiera de las versiones de Windows XP, Vista o 7 de 32 y 64 bits.

La forma de iniciar el programa es sencilla si se hace doble clic en el icono que vemos a continuación:

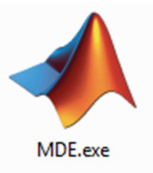


Figura 4-1. Icono del programa

La interfaz de usuario de inicio que vemos es la siguiente:

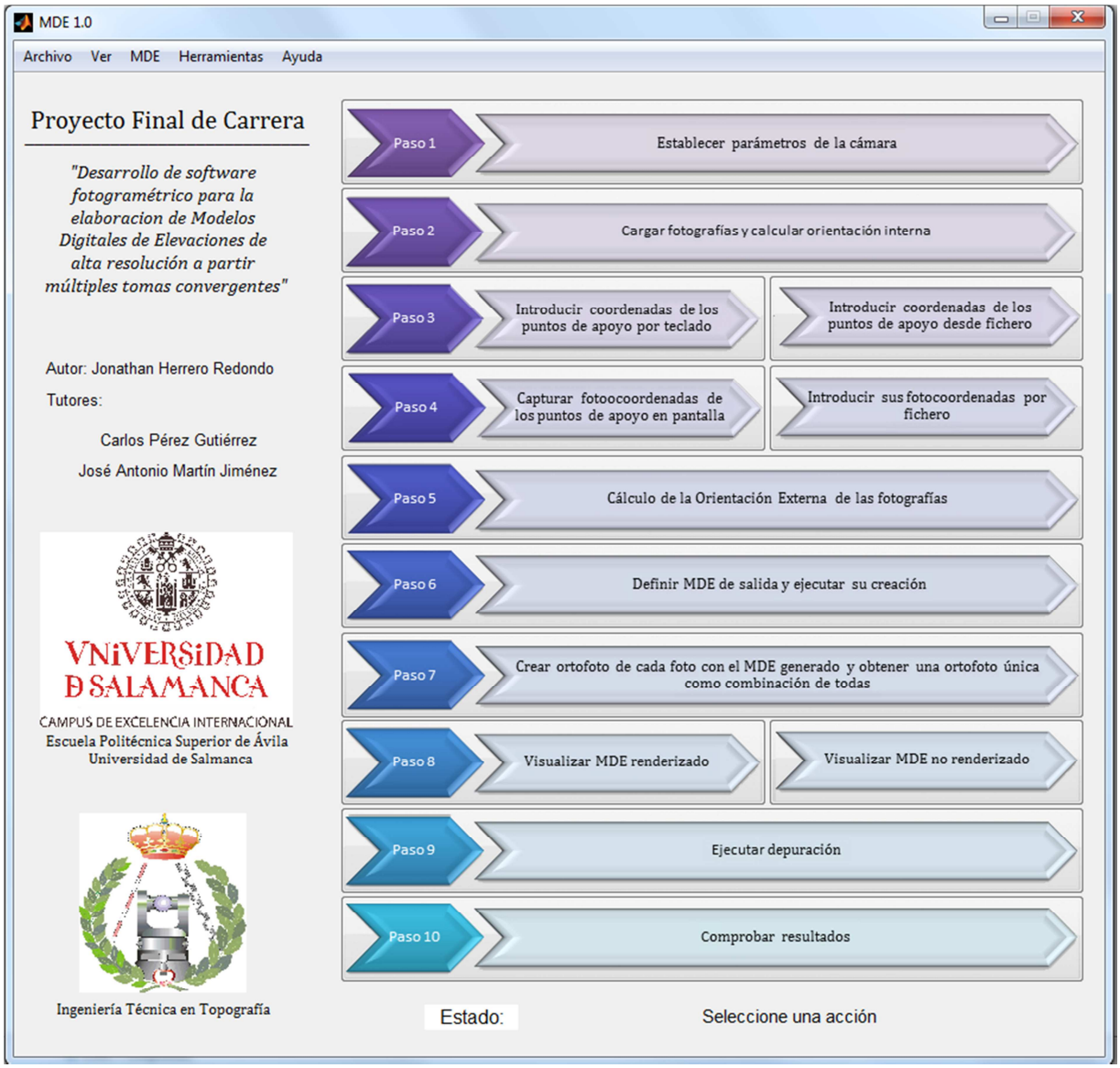


Figura 4-2. Ventana de la interfaz principal

Esta interfaz está estructurada en pasos o tareas que se pueden realizar desde la misma con hacer clic en dichos botones.

Por otra parte, encontramos una barra de menú la cual contiene las opciones de Archivo, Ver, MDE, Herramientas y Ayuda.

El menú Archivo se encarga de crear, abrir y guardar una nueva ejecución del programa y el menú Ver incluye algunas herramientas extra que nos permite, por ejemplo, proyectar líneas epipolares y verticales que muestran la perspectiva de la foto.

Con la opción MDE, de la barra de menú, podemos realizar varios de los pasos incluidos en la interfaz, estando algunas otras operaciones en la opción denominada Herramientas dentro del menú que nos permite realizar la orientación interna, la externa, o bien por ejemplo, realizar un perfil longitudinal si se cuenta con los datos necesarios.

Por último, en la opción de Ayuda se encuentra un "Acerca de..." en el que se da información sobre con que entorno de desarrollo se ha realizado el software, la fecha y que se trata de un proyecto final de carrera.

Obsérvese la barra de estado que encontramos en la parte inferior, esta nos informa sobre el progreso de la tarea actual.

4.2. Especificaciones de la cámara

Este constituye el primer paso al ejecutar nuestro programa, establecer los parámetros de la cámara. Estos parámetros son la focal, el ancho y el alto del sensor.

Mediante una interfaz se recogen estos datos y se almacenan para utilizarlos posteriormente. El aspecto de la venta es la siguiente:

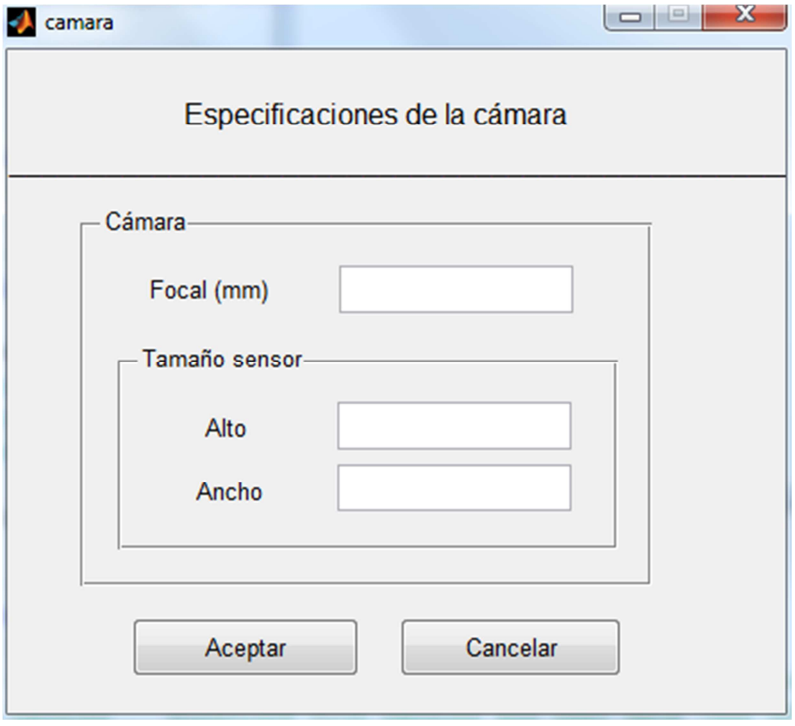


Figura 4-3. Interfaz de especificaciones de la cámara

Los parámetros que aquí introducimos de utilizar una cámara no métrica son los que obtenemos de la calibración con Photomodeler, por ejemplo.

4.3. Selección de las imágenes y orientación interna

El proyecto hace uso de varias imágenes para generar el Modelo Digital de Elevaciones, en concreto, se hace uso de cuatro imágenes de toma ligeramente convergente.

Con el propósito de cargar las mismas se muestra la siguiente interfaz, que nos permite, a la vez que cargamos las fotografías, ver estas en tamaño reducido ante la posibilidad de equivocarnos al elegirlas:

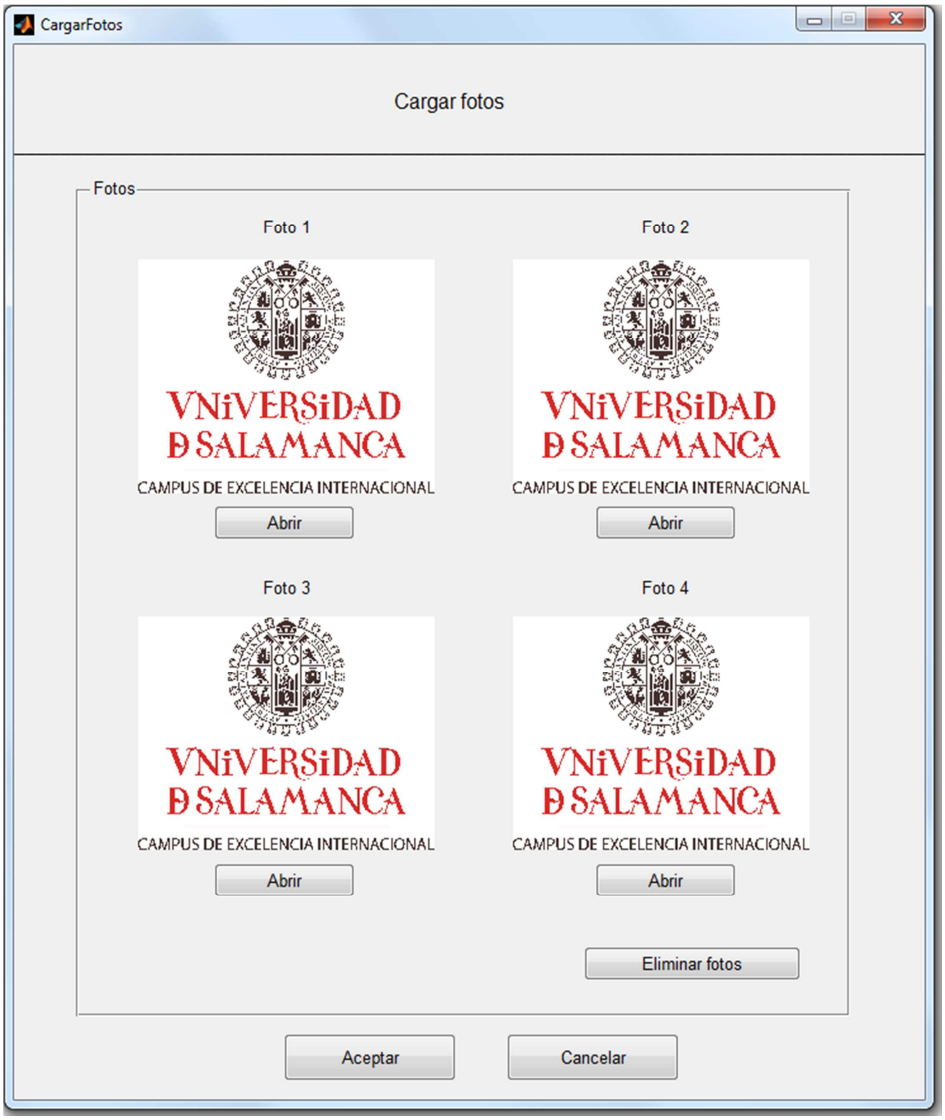


Figura 4-4. Interfaz cargar fotos

Al mismo tiempo que cargamos las fotografías y habiendo inicialmente introducido las dimensiones del sensor, podemos, realizar la orientación interna, si calculamos las dimensiones de las fotos, y aplicamos una transformación afín con las esquinas de ambos sistemas.

4.4. Introducción de los Puntos de Apoyo

Para la tarea de introducir puntos de apoyo se nos ofrecen dos alternativas:

- Introducir los puntos de apoyo por teclado
- Introducir los puntos de apoyo por fichero

Cuando realizamos esta tarea por teclado hemos de rellenar las casillas correspondientes a las coordenadas de los puntos de apoyo por medio de una interfaz, mientras que si realizamos la tarea por fichero, se supone, que esta tarea se ha realizado previamente por teclado y se ha guardado para utilizar estas coordenadas en varias ejecuciones posteriores del programa.

Para el caso de introducir estos por teclado tenemos la siguiente interfaz:

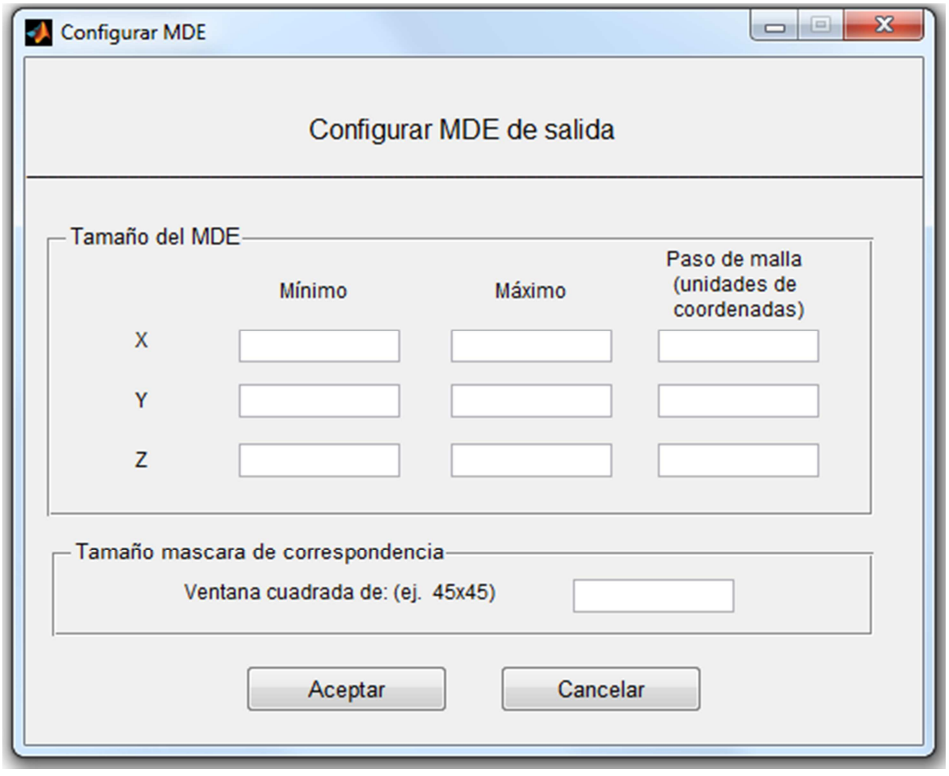


Figura 4-5. Interfaz para pedir puntos de apoyo

Se ha fijado que el número de puntos apoyo sea un total de ocho, ya, que inicialmente, fue este el número de dianas con el que se realizaron los ejemplos. Además, nos ofrece la suficiente redundancia, ya que tan solo necesitamos para que el sistema este ajustado, un total de tres puntos de apoyo.

Las incógnitas que se pretenden calcular de la orientación externa son las coordenadas del punto de vista y los giros de la orientación, es decir, 3 coordenadas y 3 giros que en total son 6 incógnitas. Por cada punto en terreno y fotograma podemos obtener 2 ecuaciones. Así, para el caso de 8 puntos tenemos un total de 16 ecuaciones. La redundancia que tenemos si restamos el número de incógnitas a las ecuaciones es por tanto de:

Conseguimos por tanto una redundancia de:

$$Redundancia = Ecuaciones - Incógnitas = 16 - 6 = 10$$

Para el caso de que queramos introducir estas por fichero la disposición de las coordenadas ha de ser por columnas con la siguiente cabecera y filas:

	X	Y	Z
1			
2			
3			
4			
5			
6			
7			
8			

Tabla 4-1. Tabla modelo para los puntos de apoyo

4.5. Captura/Medida de dianas

Un cuarto paso que se nos muestra en la interfaz principal es el que nos permite obtener las fotocoordenadas de las dianas. De nuevo tenemos dos alternativas:

- Capturar fotocoordenadas por pantalla
- Introducir fotocoordenadas por fichero

La forma de captura de fotocoordenadas por pantalla consiste en marcar sobre las imágenes con el puntero del ratón las dianas de las que conocemos sus coordenadas terreno. Es un proceso secuencial, se nos va mostrando un título de ventana diferente para cada punto.

Para una mejor precisión en la puntería de la medida de las mismas se muestra una subventana con zoom, véase un ejemplo de la misma:

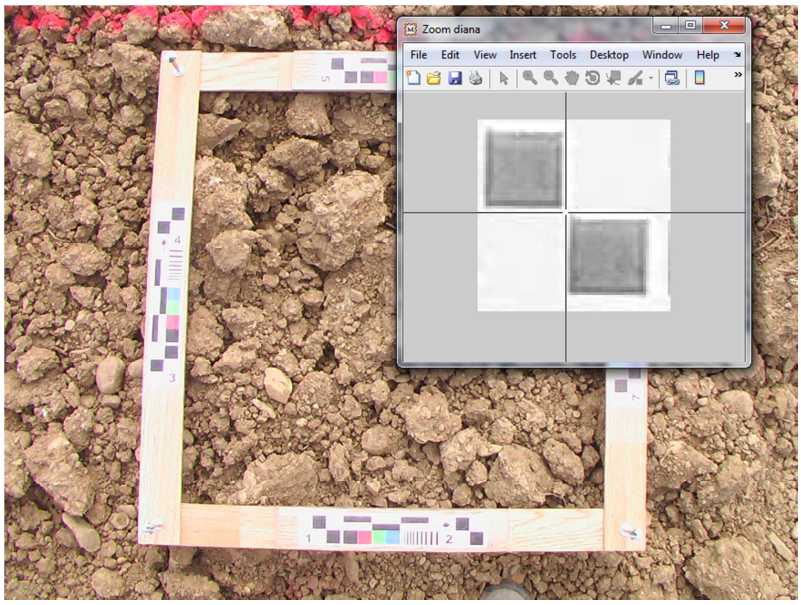


Figura 4-6. Captura de las fotocoordenadas por pantalla

Al disponer de los parámetros de la orientación interna con hacer puntería en las dianas y recoger coordenadas pixel la transformación a fotocoordenadas es trivial.

Esta tarea se realiza por medio de dos funciones una que abre las imágenes y nos permite pinchar sobre la imagen original y otra que se llama desde esta función y nos permite hacer zoom.

El prototipo de la primera función tiene dos argumentos de entrada:

```
function foto = Captura(foto, ParamOI)
```

Estos son los parámetros de la orientación interna y la fotografía. Como salida se obtiene de nuevo la foto que contendrá las fotocoordenadas. Esto se consigue formando un tipo de dato estructura que dispone de dos campos, uno el de las fotografías y otro el de sus fotocoordenadas.

Es decir, $foto(i).ima$ con el campo ima nos hace referencia a las imágenes y $foto(i).fotocoordenadas$ con el campo fotocoordenadas nos hace referencia a las fotocoordeandas. Para acceder a una fotografía concreta utilizaremos el índice entre paréntesis. Por ejemplo, si queremos acceder a la fotografía 3 pondremos $foto(3).ima$.

El prototipo de la función que hace zoom es el siguiente:

```
function [fotocoordendas]=MiZoom(imagen,EsqSI_x,EsqSI_y, ParamOI)
```

Dispone de 4 argumentos de entrada: la imagen, coordenadas de la esquina superior izquierda y derecha de la ventana de zoom y los parámetros de la orientación interna y como resultado devuelve las fotocoordenadas que se toman.

En el caso de introducir las fotocoordenadas por fichero estas se han debido guardar previamente por una captura manual. Por tanto, generamos una estructura $foto(i).fotocoordenadas$ que será guardada en un archivo y que al usuario se le va a pedir para importar las mismas.

4.6. Orientación externa

Al llegar a este punto en que disponemos de fotocoordenadas y coordenadas terreno con la suficiente redundancia estamos en condiciones de calcular la orientación externa por colinealidad, véase el apartado de la metodología de la orientación externa que encontramos en los fundamentos teóricos.

La forma de ejecutar esta acción es con el botón siguiente:

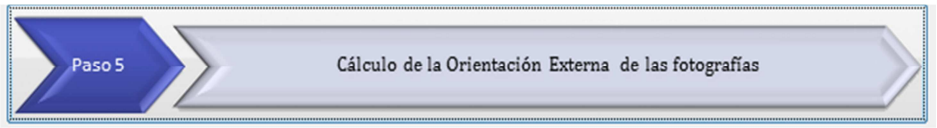


Figura 4-7. Botón calculo de la Orientación Externa

En la barra de estado se nos muestra información del avance según se finaliza la orientación de cada foto:

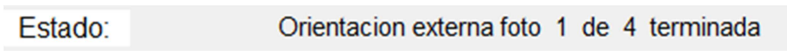


Figura 4-8. Barra estado progreso de la Orientación Externa

Como se ha descrito debemos obtener para cada foto las coordenadas del punto de vista o centro de proyección $X_o, Y_o \; y \; Z_o$ y los giros ω, φ y κ. Una vez más empleamos un función para esta tarea con el prototipo siguiente:

```
function [ParamOE, emcOE, informe] = CalculoExt(ptos_apoyo, foto, f, id)
```

Dispone de cuatro argumentos de entrada: los puntos de apoyo, la estructura de la foto con las imágenes y fotocoordenadas, la focal y un identificador de foto.

El identificador de foto sirve como índice para moverse dentro de la estructura foto. La salida que obtenemos son los parámetros de la orientación externa, el error medio cuadrático que se obtiene del cálculo por mínimos cuadrados y un informe de los cálculos iterativos hasta llegar a la solución.

La orientación externa requiere de un cálculo iterativo puesto que hay que linealizar por medio del desarrollo en serie de Taylor. El número de iteraciones se ha fijado en 15 ya que la mayoría de los casos convergen con esta cifra.

El informe nos muestra por filas las incógnitas que se van obteniendo por cada iteración. Veamos el siguiente ejemplo obtenido:

Iteración	XO(mm)	YO(mm)	ZO(mm)	Omega(grad)	Phi(grad)	Kappa(grad)	emc OE
0	1427,42129	1251,82645	2428,99349	-0,94231084	-14,9280458	1,29154937	12,0213222
1	1603,70952	1144,72242	2400,8323	-0,77239131	-8,16264429	1,23737037	14,6011784
2	1691,04438	1233,64113	2385,60612	-0,6248545	-12,0695324	1,22555177	4,97598839
3	1704,20215	1305,83892	2372,12579	-0,63598722	-12,7006353	1,2270753	3,06980901
4	1708,09562	1314,87539	2369,62813	-0,63503398	-12,8895412	1,22643868	2,63998708
5	1709,34925	1317,4404	2368,8563	-0,63503091	-12,9491929	1,22630725	2,51472321
6	1709,7619	1318,25716	2368,60495	-0,63503595	-12,9687854	1,22626589	2,47486505
7	1709,89862	1318,52509	2368,52193	-0,63503831	-12,9752735	1,22625238	2,46181093
8	1709,94402	1318,61376	2368,49438	-0,63503916	-12,9774274	1,22624792	2,45749336
9	1709,9591	1318,6432	2368,48524	-0,63503946	-12,9781431	1,22624644	2,45606063
10	1709,96411	1318,65297	2368,4822	-0,63503956	-12,9783809	1,22624594	2,45558467
11	1709,96578	1318,65622	2368,48118	-0,63503959	-12,97846	1,22624578	2,45542649
12	1709,96633	1318,6573	2368,48085	-0,6350396	-12,9784863	1,22624573	2,45537392
13	1709,96652	1318,65766	2368,48074	-0,6350396	-12,978495	1,22624571	2,45535644
14	1709,96658	1318,65778	2368,4807	-0,6350396	-12,9784979	1,2262457	2,45535064

Tabla 4-2. Informe de la orientación externa de un ejemplo

En el ejemplo, vemos como por cada iteración se llega a un valor que converge, más aun, teniendo en cuenta las unidades empleadas para el caso de milímetros pretender llegar a valores por debajo de esta resulta innecesario.

Los parámetros de la orientación externa se guardan en una estructura de la forma $ParamOE(id).ima$ donde id hace referencia a la foto. Para el caso anterior si esta fuese la fotografía número 3 tendríamos con $ParamOE(3).ima$:

1709,96658 1318,65778 2368,4807 -0,6350396 -12,9784979 1,2262457

Si quisiéramos recoger alguno de sus valores, por el ejemplo, el valor de la cota del punto de vista en otra variable, esta sería recogida en $ParamOE(3).ima(3)$:

2368,4807

Cuando se realiza la orientación externa, si queremos comprobar los resultados y por un suponer cambiar estos, podemos abrir desde el menú la opción:

$$Herramientas \rightarrow Orientación\ externa \rightarrow Ver/editar\ OE$$

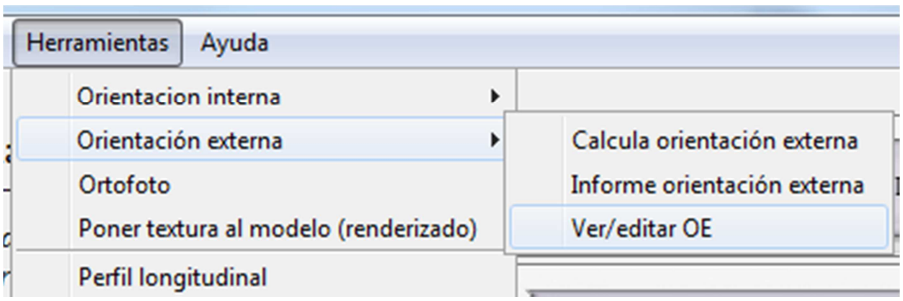


Figura 4-9. Menú herramientas para ver/editar OE

Llegamos a una interfaz como la siguiente, tomada de un ejemplo ejecutado:

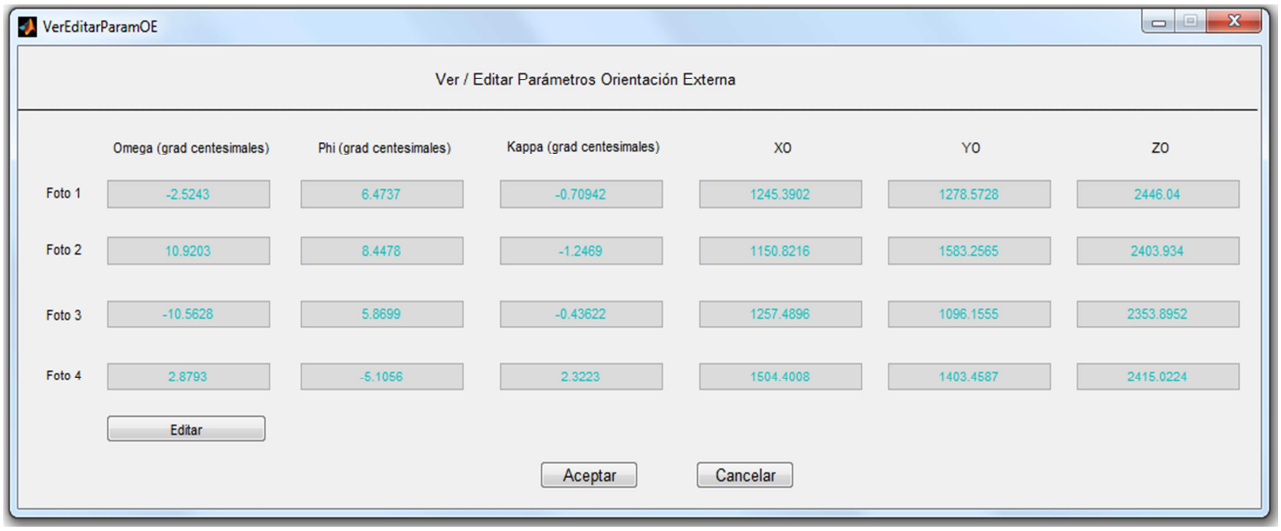


Figura 4-10. Panel de vista y edición de los parámetros de OE

Si queremos editar los valores tan solo hay que hacer clic en el botón editar y podremos rellenar cada casilla con el nuevo valor.

4.7. Creación del MDE

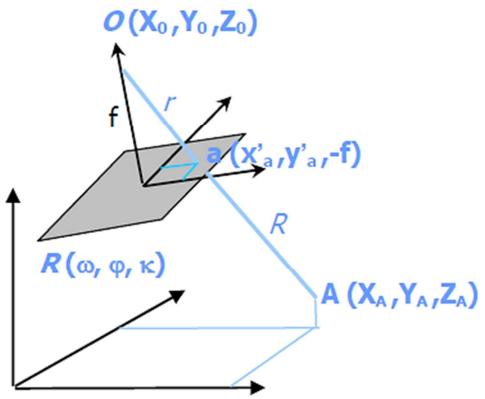
4.7.1. Primeros pasos: uso de la línea epipolar y vertical del lugar

Puesto que tenemos la orientación externa calculada estamos en condiciones de restituir puntos y esto se pretende llevarlo a cabo de forma automática, por ello, conviene analizar distintas metodologías.

Comencemos analizando un primer procedimiento que se pensó, el cual no se llevó a la práctica por algunos de sus inconvenientes. Este primer procedimiento se basa en utilizar dos elementos geométricos: la línea epipolar y la vertical del lugar.

Sea el plano epipolar el plano formado por la base y los dos fotovectores homólogos, es decir, la recta que une los centros de proyección de dos fotografías y los rayos proyectivos homólogos, la intersección de dicho plano con cada una de las fotografías da lugar a las rectas epipolares.

A la hora de analizar este método debemos pensar primero como proyectar las rectas epipolares. La forma de realizarlo es sencilla, supóngase que en una fotografía seleccionamos un punto y queremos ver su recta epipolar en el resto de fotografías, para ello se hace uso de colinealidad:



$$\begin{bmatrix} X_A \\ Y_A \\ Z_A \end{bmatrix} = \lambda \cdot R_1 \cdot \begin{bmatrix} x'_a \\ y'_a \\ -f \end{bmatrix} + \begin{bmatrix} X_{O1} \\ Y_{O1} \\ Z_{O1} \end{bmatrix}$$

Figura 4-11. Condición de colinealidad

Sea un punto en el terreno A, cuyas coordenadas terreno se desconocen y con fotocoordenadas en la foto 1 x'_a e y'_a, así como los parámetros de la orientación externa conocidos y por ello R_1 y X_{O1}, Y_{O1} y Z_{O1} conocidos. Sea un factor de escala desconocido λ, si aplicamos dos factores de escala arbitrarios, el punto A se prolongara más o menos en el rayo proyectivo estando siempre sobre el plano epipolar. Por tanto, asignando por ejemplo valores $\lambda_1=1$ y $\lambda_2=100$ tenemos dos valores para las coordenadas terreno del punto A:

$$\begin{bmatrix} X_{A1} \\ Y_{A1} \\ Z_{A1} \end{bmatrix} = \lambda_1 \cdot R_1 \cdot \begin{bmatrix} x'_a \\ y'_a \\ -f \end{bmatrix} + \begin{bmatrix} X_{O1} \\ Y_{O1} \\ Z_{O1} \end{bmatrix}$$

$$\begin{bmatrix} X_{A2} \\ Y_{A2} \\ Z_{A2} \end{bmatrix} = \lambda_2 \cdot R_1 \cdot \begin{bmatrix} x'_a \\ y'_a \\ -f \end{bmatrix} + \begin{bmatrix} X_{O1} \\ Y_{O1} \\ Z_{O1} \end{bmatrix}$$

Ahora, al utilizar la otra expresión de colineadad en la que desaparece el factor de escala, conseguimos obtener fotocoordenadas en las otras imágenes ya que disponemos de dos puntos de apoyo pertenecientes a la recta epipolar y sus parámetros de la orientación externa. Así ,con dos puntos podemos calcular la recta que pasa por ellos en el plano en dos dimensiones.

$$x''_a = -f \frac{r_{11} \cdot (X_A - X_O) + r_{12} \cdot (Y_A - Y_O) + r_{13} \cdot (Z_A - Z_O)}{r_{31} \cdot (X_A - X_O) + r_{32} \cdot (Y_A - Y_O) + r_{33} \cdot (Z_A - Z_O)}$$

$$y''_a = -f \frac{r_{21} \cdot (X_A - X_O) + r_{22} \cdot (Y_A - Y_O) + r_{23} \cdot (Z_A - Z_O)}{r_{31} \cdot (X_A - X_O) + r_{32} \cdot (Y_A - Y_O) + r_{33} \cdot (Z_A - Z_O)}$$

La línea epipolar constituye una restricción a la hora de buscar puntos homólogos ya que se puede ver que en una fotografía un punto A se proyecta sobre otra fotografía en un punto perteneciente a la línea epipolar.

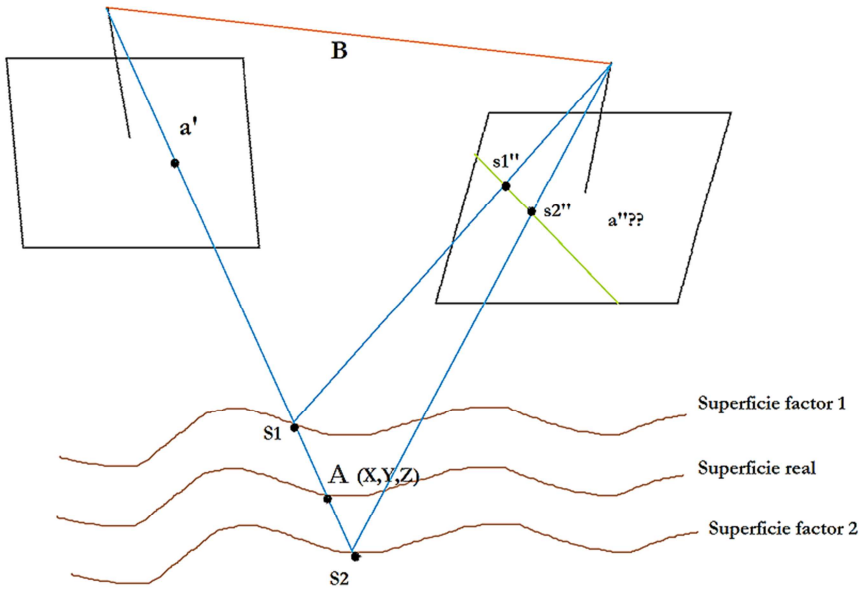


Figura 4-12. Proyección de la línea epipolar

En la figura 4-12 se puede ver que los puntos A, a', s1" y s2" pertenecen al plano epipolar. Los puntos s1" y s2" definen la línea epipolar y en un lugar de esta línea que desconocemos se encuentra el punto homologo a" que vendría dado por la intersección del rayo proyectivo que une A y el centro de proyección de la imagen derecha con la línea epipolar.

El proceso de búsqueda del punto homólogo continúa mediante el empleo de algún tipo de correspondencia, por ejemplo, correspondencia basada en áreas.

Además de la línea epipolar, podemos establecer otra restricción que consiste en añadir la proyección de la vertical del lugar en la imagen.

Una recta vertical en el espacio que es proyectada en las imágenes, fuga radialmente respecto del nadir. Los limites inferior y superior de esta recta, sean denotados como S e I y sus proyecciones s', i' y en la otra imagen s", i" resulta que s', s" pertenecen a las líneas epipolares al igual que lo hacen i', i".

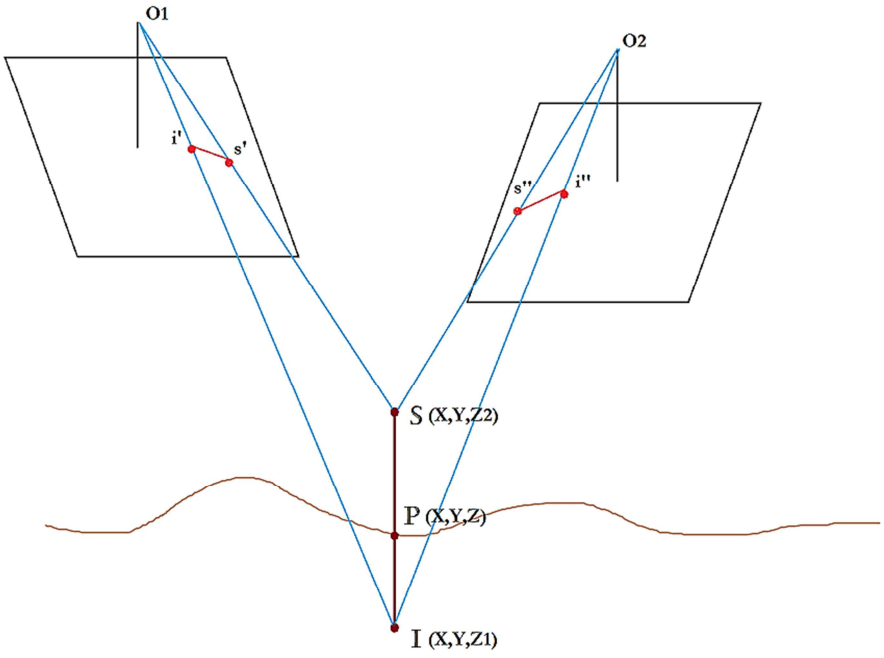


Figura 4-13. Proyección de la vertical del lugar

Sea un punto P cuyas coordenadas planimétricas coinciden con las del limite superior, S, y con las del limite inferior, I, si se desea conocer la cota del punto P, partiendo de una cota limite superior e inferior, el espacio de búsqueda se realiza en la proyección de estas verticales. Aplicando colinealidad podemos obtener las fotocoordenadas de los límites inferior y superior y buscar entre estos los puntos homólogos del punto P para posteriormente calcular su cota.

Los puntos proyectados de los límites inferior y superior no tendrán por qué ser homólogos ya que estos en el terreno intersectarán en distintas zonas. Así, al aplicar correlación para la búsqueda se comprueba que en los límites la correlación es baja.

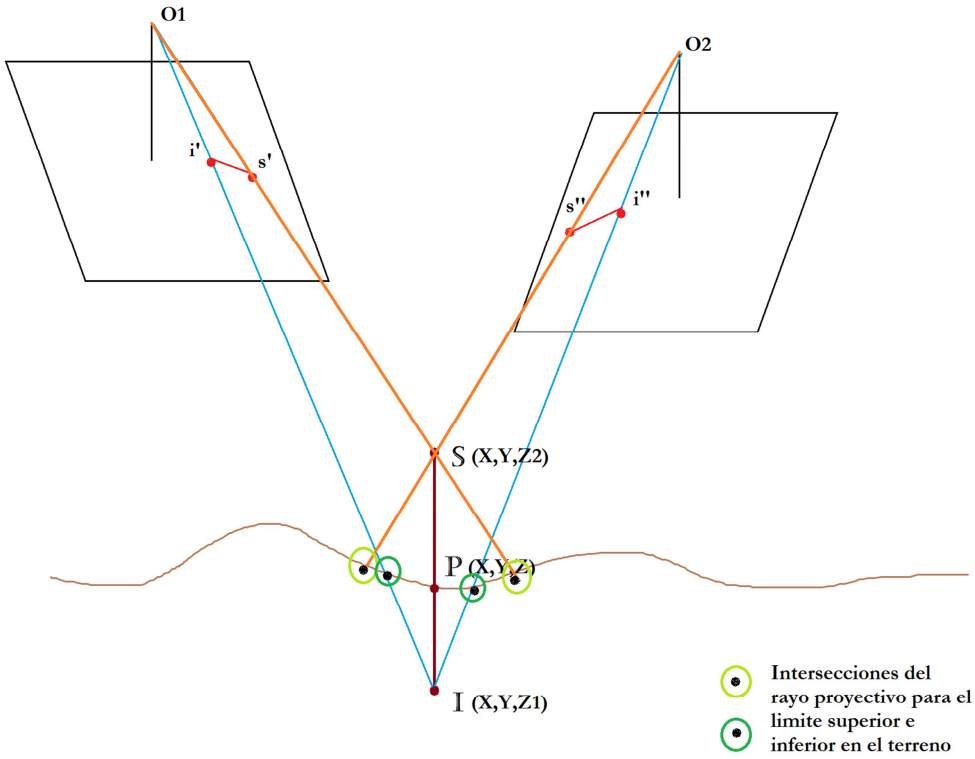


Figura 4-14. Intersección de los rayos proyectivos de los límites con el terreno

Analizadas estas dos restricciones parece útil combinar estos métodos para llegar a una búsqueda de puntos homólogos y así restituir puntos pertenecientes a una malla de coordenadas planimétricas con los que obtener un MDE.

Sin embargo, al aplicar este método se observa que los cálculos se realizan con gran lentitud y se está sometido a varios errores por combinación de métodos y sus errores inherentes.

Si combinamos los dos métodos la técnica que inicialmente se ideó consiste en proyectar las verticales del lugar en las dos imágenes, a continuación tomar uno de los limites de la vertical en su proyección de la imagen y trazar la línea epipolar que pasa por este punto cortando en la vertical del lugar.

Sobre este punto de corte se calcula el coeficiente de correlación llevando a cabo una correspondencia basada en áreas y almacenando este valor en una matriz.

Seguidamente se realiza lo mismo con otro punto de la proyección de la vertical tomando incrementos a nivel de pixel sobre esta recta. Los incrementos pueden ser enteros o con precisión subpíxel según se programe la función.

Al llevar a cabo este procedimiento hasta encontrar el máximo coeficiente de correlación, se observa un tiempo de ejecución muy alto debido principalmente al número de imágenes empleado y al tamaño de estas que puede hacer que sean varios los píxeles involucrados en una recta y que intervienen en el cálculo. Por tanto, si la imagen es de gran resolución y se utilizaran muchas imágenes este procedimiento sería lento y poco práctico.

Encontramos errores debidos a la intersección de la línea epipolar con la vertical del lugar. Los errores vendrán dados en consecuencia de la precisión con la que se realiza la orientación externa. Del mismo modo, encontramos errores debidos a las malas intersecciones que se producen entre rectas con ángulos pequeños siendo en algunos casos estas rectas casi paralelas, a lo que si sumamos errores de la orientación externa, los cortes de rectas de línea epipolar y vertical del lugar pueden darse fuera de los límites superior e inferior de la vertical del lugar.

En el software se ha incluido un menú que permite ver la proyección de la línea epipolar tomando una foto como referencia en la que se selecciona un punto. También se ha incluido en este menú la opción de ver la proyección de la vertical de lugar distribuidas dentro de la zona que envuelven las dianas de los puntos de apoyo.

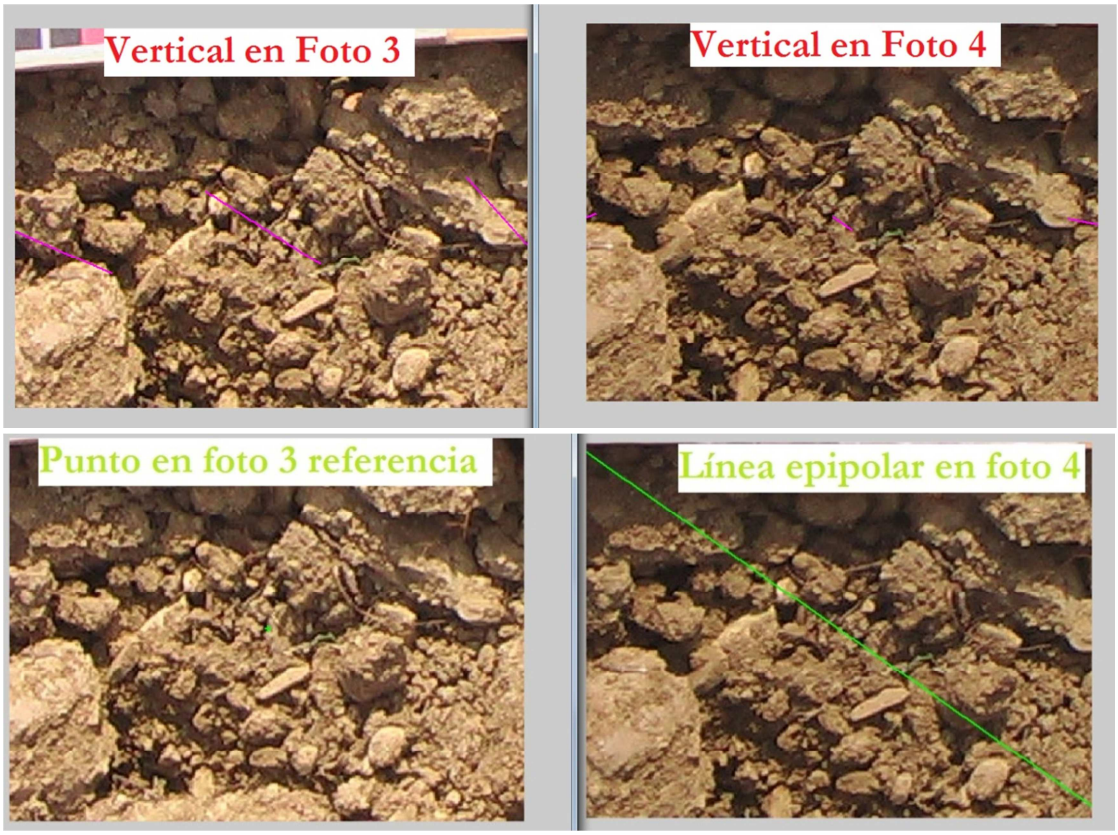


Figura 4-15. Posible error por paralelismo entre línea epipolar y vertical del lugar

4.7.2. Procedimiento de cálculo

Tras varias pruebas y con el objetivo de conseguir mejores resultados se ideó una mejor forma de calcular la cota de los puntos del MDE consiguiendo rapidez en el proceso y evitando ciertos errores al prescindir de algunos de los métodos.

Este método trabaja a nivel de objeto en vez de a nivel de imagen como se estaba haciendo anteriormente. Anteriormente, conocida la pendiente y ordenada sobre el origen de la recta que define la vertical del lugar proyectada, es decir, parámetros m y n de la ecuación de la recta, se puede colocar una máscara de correspondencia centrada en cada pixel que toca la recta. Sin embargo, al realizar esta tarea con un bucle con incrementos de una unidad de pixel o inferior, hay que tener en cuenta la inclinación de la recta.

Al tener en cuenta la inclinación de la recta quiere decir, que si tomamos incrementos de un pixel en las columnas o filas puede que alguna de estas tenga más de una fila o columna perteneciente a la recta. Véase el siguiente ejemplo:

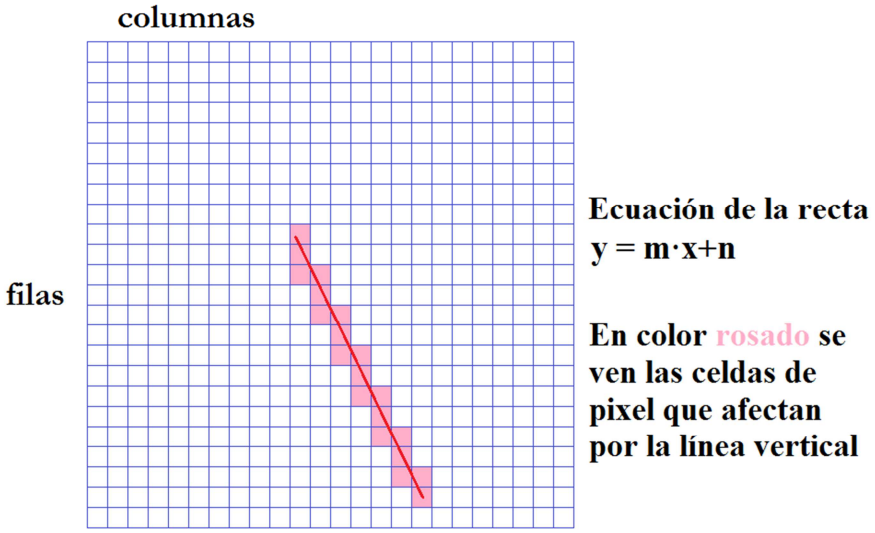


Figura 4-16. La recta en la imagen digital como caso discreto

En este ejemplo (figura 4-16), se observa que con la inclinación dada por la recta roja cada paso de columna lleva consigo tres filas de la recta y cada dos pasos de fila lleva también 2 columnas pertenecientes a la recta.

La máscara de correlación debería centrarse en cada una de las posiciones y esto no siempre es posible. Al trabajar con valores discretos lo normal es trabajar con posiciones enteras de fila y columna, si bien se puede trabajar con valores de pendiente no enteros,. Por ello, se debe redondear el valor para la fila y columna o lo que es lo mismo, los valores de y, x de la ecuación de la recta.

Conocido el problema de trabajar a nivel de imagen se dispone a trabajar a nivel de objeto, esto es con desplazamientos a lo largo de la vertical en el espacio y no según se proyecta en la imagen. Si anteriormente realizábamos desplazamientos en la unidad de pixel o inferior a esta, ahora, podemos realizar incrementos en milímetros, centímetros o la unidad que se considere oportuna.

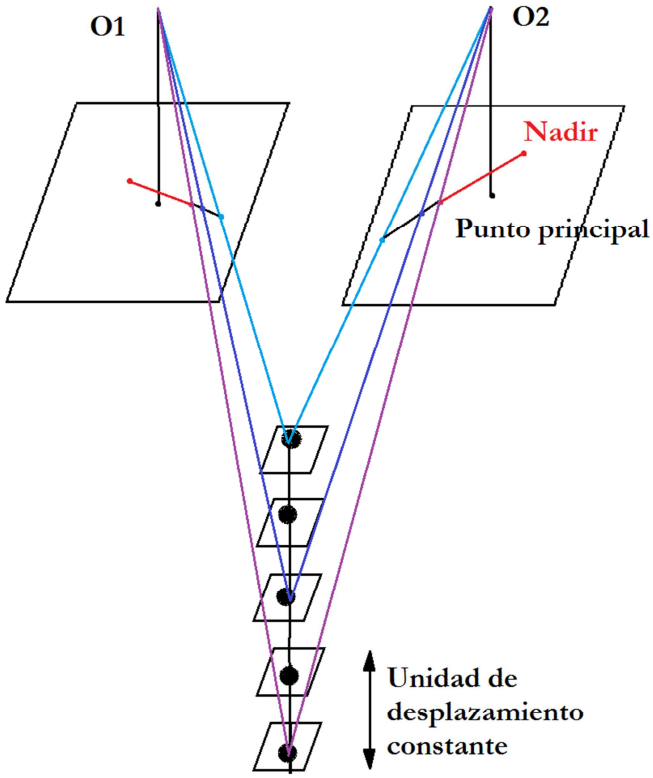


Figura 4-17. Desplazamientos en el espacio objeto

En el método utilizado la vertical se proyecta dada una unidad de desplazamiento constante. Por cada desplazamiento se proyecta el punto objeto sobre todas las imágenes de esas coordenadas terreno y se analiza el índice de correlación formando una máscara cuadrada centrada en valores enteros de la proyección.

Con este procedimiento evitamos el uso de la línea epipolar y evitamos cometer errores de centrado de máscara que se comenten al trabajar en el espacio imagen. Sin embargo, el método puede ser lento si se utilizan desplazamientos cortos con respecto a la longitud de la vertical.

Téngase en cuenta que esto se debe realizar para todas las fotografías pero en esta nueva variante evitamos trabajar con parejas de imágenes y la combinatoria que esto supone.

La cuestión surge en ¿Cómo trabajar con todas las imágenes a la vez, de modo, qué, se pueda establecer un método de correspondencia múltiple?

Sea el coeficiente de correlación:

$$\rho = \frac{\sigma_{xy}}{\sigma_x \cdot \sigma_y}$$

Dependiente de la covarianza entre dos variables x e y, y de las desviaciones típicas de x e y. Este valor esta normalizado entre -1 y 1, siendo el mejor de los casos el de la unidad positiva que indica que los patrones son iguales.

Los patrones se analizan de dos en dos ya que son dos variables las que podemos analizar. Para una imagen representada por una función $f1(x,y)$ y otra imagen $f2(x,y)$ tomamos máscaras que se desplazan a lo largo de una ventana de búsqueda tales que $g1(x,y)$ y $g2(x,y)$.

Ahora, si antes surgía una combinatoria por parejas de imágenes para buscar el mejor candidato por cada pareja de estas y formar varios modelos y elegir el más adecuado descartando aquellos que superan cierta tolerancia. En este caso, se nos forma una combinatoria de correspondencias ya que por cada máscara $g1(x,y)$ $g2(x,y)$ $g3(x,y)$ $g4(x,y)$ de cada una de las cuatro imágenes podremos calcular:

$$\frac{n!}{r! \cdot (n-r)!} = \frac{4!}{2! \cdot (4-2)!} = \frac{24}{4} = 6$$

Para n=4 máscaras y con r=2 parejas, puesto que se toman de dos en dos, tenemos un total de 6 combinaciones.

Son seis el número de factores de correlación que se forman por cada posición de la ventana de búsqueda. No todos estos factores tienen porque ser buenos, es decir, positivos, ya que en función de la convergencia de las tomas tendremos mejor o peor búsqueda.

Con los 6 factores de correlación que se obtienen por cada posición de la ventana de búsqueda debemos establecer un estimador que nos diga cual posición es la del homólogo o la mejor candidata para ella.

Se ha pensado aquí en utilizar dos criterios: la media y el máximo.

Para empezar, se hace uso de la media y se introduce en un informe para cada posición. La media hace que en caso de obtener valores buenos y malos estos queden reflejados, pues, el uso del máximo en un principio solo reflejaría los valores buenos (valores altos positivos de correlación) que pueden ser falsos positivos.

La función de máximo será utilizada en caso de encontrar valores de media iguales con el fin de elegir de ellos el que posea un máximo de correlación mayor.

A continuación, se muestra un fragmento del informe obtenido en una búsqueda para un punto de la malla del MDE con coordenadas planimétricas fijas:

X	Y	Z	col 1	fil 1	col 2	fil 1	col 3	fil 3	col 4	fil 4	ρ media	ρ máxima
1104	1105	926	525	985	568	989	461	999	511	1038	0,48776661	0,55994584
1104	1105	927	525	985	568	989	461	1000	511	1038	0,48045787	0,55994584
1104	1105	928	525	985	569	990	460	1000	511	1039	0,50546895	0,60142363
1104	1105	929	524	985	569	990	459	1000	510	1039	0,50076301	0,62806639
1104	1105	930	524	985	569	990	459	1000	510	1040	0,50662487	0,62806639
1104	1105	931	524	985	569	990	458	1000	510	1040	0,51956824	0,62806639
1104	1105	932	524	985	569	990	458	1001	509	1041	0,50412211	0,62806639
1104	1105	933	524	985	569	990	457	1001	509	1041	0,51577678	0,62806639
1104	1105	934	523	985	570	991	457	1001	509	1042	0,5097558	0,65233996
1104	1105	935	523	986	570	991	456	1001	509	1042	0,52126596	0,64896463
1104	1105	936	523	986	570	991	456	1001	508	1043	0,51476502	0,64896463
1104	1105	937	523	986	570	991	455	1002	508	1043	0,51250756	0,64896463
1104	1105	938	522	986	570	991	454	1002	508	1044	0,53281565	0,63521572
1104	1105	939	522	986	570	991	454	1002	508	1044	0,53281565	0,63521572
1104	1105	940	522	986	571	992	453	1002	507	1045	0,52392136	0,62581949
1104	1105	941	522	986	571	992	453	1002	507	1045	0,52392136	0,62581949
1104	1105	942	521	986	571	992	452	1003	507	1046	0,52367214	0,598516
1104	1105	943	521	986	571	992	452	1003	506	1046	0,50152003	0,598516
1104	1105	944	521	986	571	992	451	1003	506	1047	0,49576432	0,598516
1104	1105	945	521	986	571	992	450	1003	506	1047	0,49576432	0,598516
1104	1105	946	520	986	571	993	450	1004	506	1048	0,49382677	0,58916583
1104	1105	947	520	986	572	993	449	1004	505	1048	0,46618259	0,5821476
1104	1105	948	520	987	572	993	449	1004	505	1049	0,48139211	0,58773788
1104	1105	949	520	987	572	993	448	1004	505	1049	0,47875085	0,58023225
1104	1105	950	519	987	572	993	448	1004	505	1050	0,46771561	0,57037278
1104	1105	951	519	987	572	993	447	1005	504	1050	0,45245439	0,57751277
1104	1105	952	519	987	572	994	446	1005	504	1051	0,44132376	0,5434157
1104	1105	953	519	987	573	994	446	1005	504	1051	0,43140775	0,5434157
1104	1105	954	518	987	573	994	445	1005	503	1052	0,40571565	0,51498784
1104	1105	955	518	987	573	994	445	1005	503	1052	0,40571565	0,51498784
1104	1105	956	518	987	573	994	444	1006	503	1053	0,38857354	0,49468553

Tabla 4-3. Informe de correspondencia en el MDE

En la Tabla 4-3 se puede ver que para dos correlaciones medias que son máximas se obtienen al mismo tiempo dos correlaciones máximas iguales también con la diferencia para estas filas de una unidad en cota, siendo estas de milímetros. Esto quiere decir, que no se puede conseguir más precisión con las imágenes tomadas para el ejemplo propuesto ya que la ventana de búsqueda con estos incrementos situándose en posiciones exactas se sitúa con ±1 mm en la misma posición de fila columna.

Para las dos filas de la tabla, con valores de fila en imagen 1 (fil1) columna en imagen 1 (col1) y sus correspondientes en las otras imágenes, vemos que estas coinciden en ambas filas de la tabla.

Pero entonces, ¿Cuál es la cota que asignaremos a nuestro MDE para las coordenadas planimétricas X=1104 Y=1105, Z=938 ó Z=939?

El resultado que se asigna está implementado en el software para que se examine el máximo de correlación en caso de igualdad de correlación media máximo. Si la primera correlación máxima encontrada, es mayor que la segunda, nos quedamos con la primera y sino con la segunda. En caso de igualdad al utilizar el operador “>” nos quedamos con la segunda. Esto tiene su lógica ya que como queremos tomar valores de cota con precisión no inferior a la que se dan los puntos de apoyo, de tomarse la media entre ambos y redondear por encima de 0,5 como entero, tomaríamos el valor mayor de Z=939.

Véase un pequeño fragmento de código que se encarga de esta decisión:

```
        if length(pos)>1
            if mcorrel2(pos(1))>mcorrel2(pos(2))
                pos=pos(1);
            else
                pos=pos(2);
            end
        end

        X(I,J)=informeaux(I,J).foto(pos,1);
        Y(I,J)=informeaux(I,J).foto(pos,2);
        Z(I,J)=informeaux(I,J).foto(pos,3);
```

Figura 4-18. Fragmento de código para elegir el mejor candidato entre correlaciones máximas

En este fragmento vemos como las coordenadas terreno de este punto pasan a ser las situadas en esta fila y columna. Esto supone rapidez en cálculo ya que no tenemos que calcular la coordenadas terreno con las filas y columnas de cada imagen aplicando el conocido sistema aplicado a la obtención de coordenadas terreno.

Para el caso del par fotogramétrico el sistema es de la forma:

$$\begin{bmatrix} 1 & 0 & 0 & -a_x & 0 \\ 0 & 1 & 0 & -a_y & 0 \\ 0 & 0 & 1 & -a_z & 0 \\ 1 & 0 & 0 & 0 & -c_x \\ 0 & 1 & 0 & 0 & -c_y \\ 0 & 0 & 1 & 0 & -c_z \end{bmatrix} \begin{bmatrix} X_A \\ Y_A \\ Z_A \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} X_{O1} \\ Y_{O1} \\ Z_{O1} \\ X_{O2} \\ Y_{O2} \\ Z_{O2} \end{bmatrix}$$

Donde:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = R_1 \begin{bmatrix} x_1 \\ y_1 \\ -f \end{bmatrix}$$

Siendo x_1, y_1 las fotocoordenadas en la imagen 1 y R_1 la matriz de rotación. Del mismo modo se establece que:

$$\begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} = R_2 \begin{bmatrix} x_2 \\ y_2 \\ -f \end{bmatrix}$$

Al tener 4 imágenes podemos ampliar el sistema como se sigue:

$$\begin{bmatrix} 1 & 0 & 0 & -a_x & 0 & 0 & 0 \\ 0 & 1 & 0 & -a_y & 0 & 0 & 0 \\ 0 & 0 & 1 & -a_z & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -c_x & 0 & 0 \\ 0 & 1 & 0 & 0 & -c_y & 0 & 0 \\ 0 & 0 & 1 & 0 & -c_z & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & -d_x & 0 \\ 0 & 1 & 0 & 0 & 0 & -d_y & 0 \\ 0 & 0 & 1 & 0 & 0 & -d_z & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -e_x \\ 0 & 1 & 0 & 0 & 0 & 0 & -e_y \\ 0 & 0 & 1 & 0 & 0 & 0 & -e_z \end{bmatrix} \begin{bmatrix} X_A \\ Y_A \\ Z_A \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix} = \begin{bmatrix} X_{O1} \\ Y_{O1} \\ Z_{O1} \\ X_{O2} \\ Y_{O2} \\ Z_{O2} \\ X_{O3} \\ Y_{O3} \\ Z_{O3} \\ X_{O4} \\ Y_{O4} \\ Z_{O4} \end{bmatrix}$$

Siendo ahora los parámetros d y e los correspondientes a la tercera y cuarta imagen.

En Matlab programar algo así no es costoso y su ejecución es bastante rápida, sin embargo, podemos prescindir de este procedimiento de cálculo con lo anterior explicado, algo, que no sucedía de trabajar a nivel de imagen.

Al finalizar estos pasos se obtiene una matriz que para cada posición X,Y que corresponden a una posición de la matriz contiene su valor de cota Z.

Cuando ejecutamos esta tarea por la interfaz gráfica, se nos muestra la siguiente ventana en la que se solicitan datos relativos al tamaño del MDE y el paso de malla correspondiente a los incrementos en planimetría y también en altimetría para pasar de un punto a otro.

También se ofrece un recuadro con el que especificar el tamaño de la máscara de correspondencia. En esta vemos que se refleja el ejemplo de 45 que es el utilizado con las fotos tomadas para realizar pruebas.

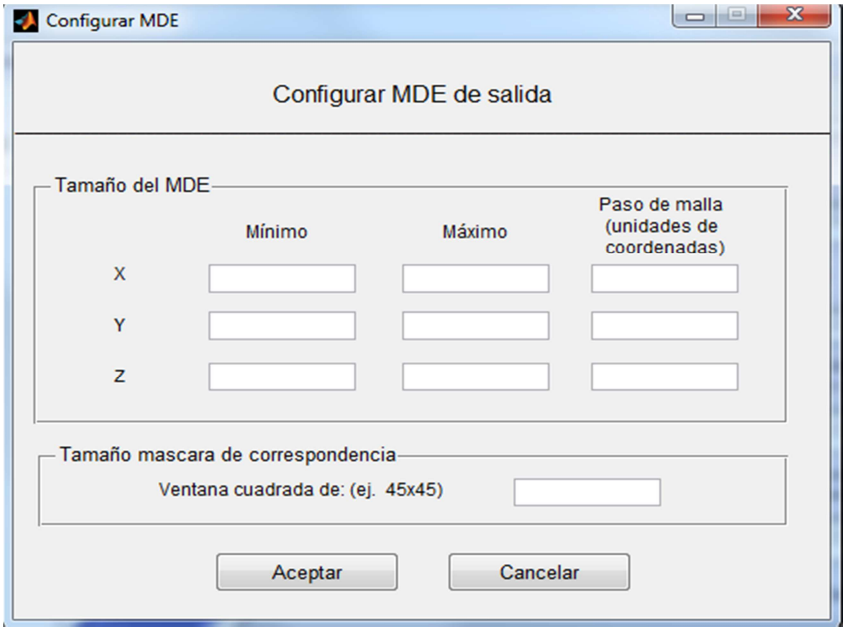


Figura 4-19. Ventana interfaz para configurar el MDE

4.8. Ortofoto y textura del MDE

En continuación a lo que se realiza en este proyecto, se encuentra la función que nos permite renderizar el modelo o lo que es lo mismo dotarlo de una textura acorde a la realidad.

Esta tarea lleva consigo la elaboración de una ortofoto con cada foto utilizada, empleando uno de los métodos conocidos como método ascendente o descendente. La ortofoto que se va crear será una ortofoto simple que no se corresponde con una "true orto", ya, que en estas últimas, debemos llevar a cabo un análisis de visibilidad que quedan en un principio fuera de los objetivos que se pretenden en el proyecto.

La metodología empleada ha sido la del método ascendente ya que de esta forma no interpolamos cotas, pues será el nivel digital el que tome un tipo de interpolación pudiéndose tomar el vecino más cercano.

Estas tareas las encontramos en la interfaz de inicio con los siguientes botones:

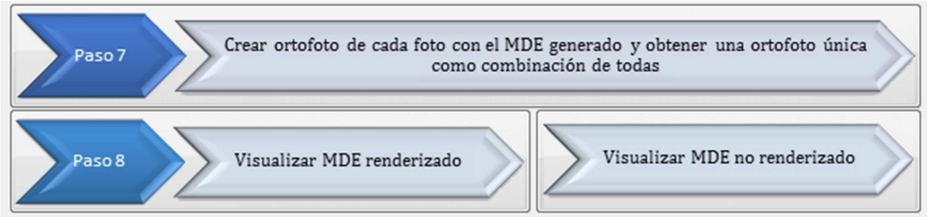


Figura 4-20. Botón para crear ortofoto y visualizar MDE

4.9. Depuración del MDE

Finalizando con los pasos que se incluyen en la interfaz de inicio se encuentra la que nos permite depurar el MDE.

Al depurar MDE conseguimos eliminar aquellos picos o sumideros que creemos que el MDE no tiene. La aparición de estos se debe a falsos positivos en la búsqueda por correlación en zonas donde la convergencia puede ser mayor.

Se ha optado por un filtro de suavizado para llevar a cabo la depuración. Este filtro consiste en una máscara de mediana de tamaño 15x15. Este filtro funciona de forma selectiva en aquellas zonas en las que supera una cierta tolerancia.

El método operativo en la selección es el que se describe a continuación:

- Calculamos la media y la desviación típica de la máscara de tamaño 15x15, es decir, 225 valores.
- Se fija la tolerancia en $T = 2.5 \cdot \sigma$, siendo σ la desviación típica. Esta fórmula es empleada cuando se realizan varias medidas a un mismo observable, por ejemplo medir la cota de un mismo punto varias veces obteniendo su error medio cuadrático. Esta relacionada con la curva de Gauss indicando el margen en el que se produce un punto de inflexión de la misma. Para el entorno de 15x15 se considera factible su aplicación suponiendo que no existen cambios de pendientes abruptos.
- Comprobamos que el valor central se encuentra dentro del margen dado por:

  $$\text{media} - 2.5 \cdot \sigma \geq \text{valor central} \geq \text{media} -+ 2.5 \cdot \sigma$$

  Si el valor está fuera de este margen se aplica el filtro o máscara centrada en este punto en caso contrario se deja el valor calculado.

  Sera elección del usuario si quiere o no aplicar el filtro en función de los resultados obtenidos viendo si estos se ajustan a la realidad o no.

4.10. Perfil longitudinal

Dentro del menú de herramientas encontramos la opción que nos permite ver el perfil longitudinal entre dos puntos que se piden al usuario. Siguiendo las indicaciones de la siguiente figura llegamos a esta opción:

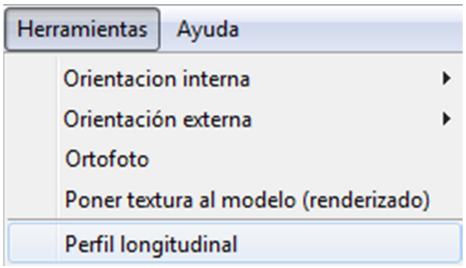


Figura 4-21. Menú herramientas para el perfil longitudinal

Para poder ejecutar esta acción hemos de tener creado nuestro MDE y haber obtenido una ortofoto, ya que será a partir de esta con la que seamos capaces de situar los puntos que definen el inicio y fin del mismo.

Se muestra a continuación un ejemplo del mismo:

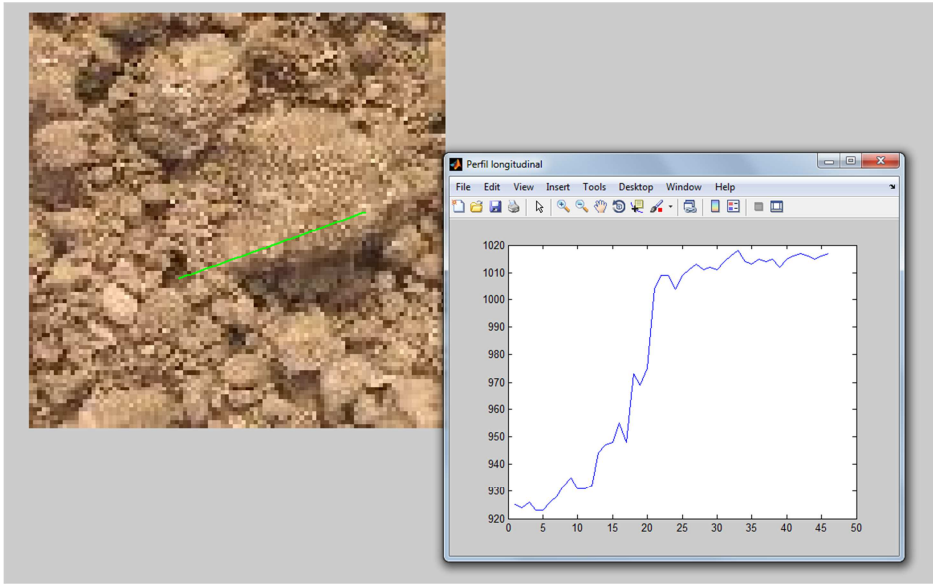


Figura 4-22. Ejemplo de un perfil longitudinal

CAPÍTULO 5. ALGORITMIA

5.1. Operaciones previas

Para elaborar un programa como el que queremos tenemos que pensar cuales son las operaciones necesarias hasta llegar al resultado final, así como cuales son los datos de entrada y cuales son los datos de salida.

Conviene realizar un diagrama de flujo explicativo que contenga lo anterior citado, así una forma puede ser la siguiente:

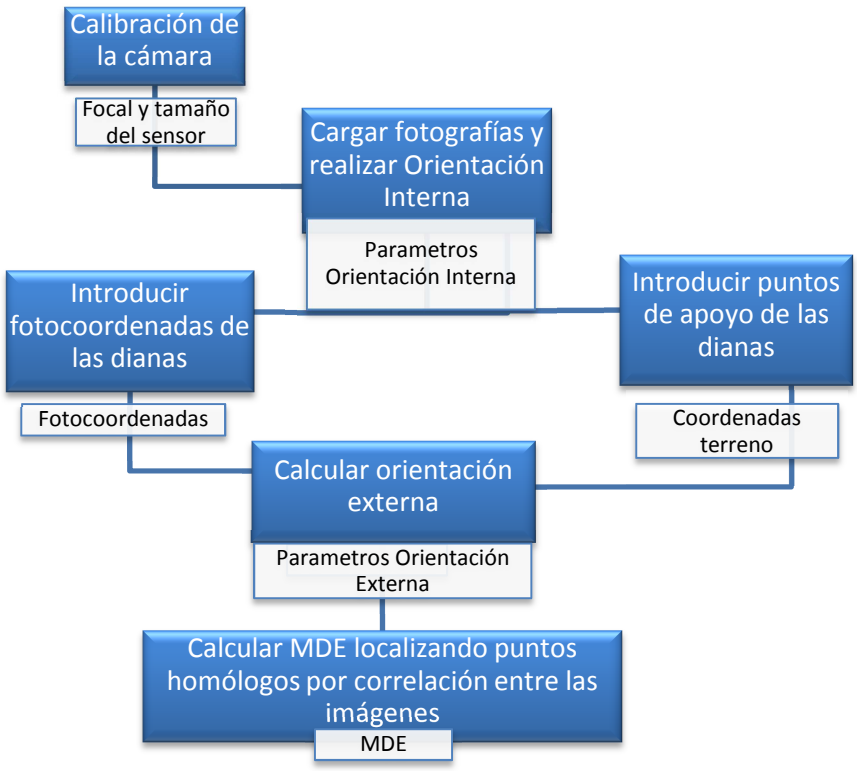


Figura 5-1. Diagrama de flujo de las operaciones previas

Analizando este diagrama podemos observar que al calibrar la cámara nos quedamos con la focal y el tamaño del sensor y a continuación en recuadro azul de operaciones tenemos que cargar las fotografías y realizar la orientación interna para llegar al resultado, que en este caso, son los parámetros de la orientación interna. Los parámetros de orientación interna nos permitirán pasar de coordenadas píxel a fotocoordenadas (caso directo) y de fotocoordenadas a coordenadas píxel (caso inverso).

Tras la orientación interna queremos realizar la orientación externa, que implica como datos de entrada las fotocoordenadas y las coordenadas terreno de las 8 dianas. Mediante dos operaciones previas introducimos estos datos.

Con los datos calculamos la orientación externa que nos facilitan los parámetros de orientación externa. Los parámetros de la orientación externa nos permiten proyectar en la fotografía líneas epipolares y verticales del lugar.

Estos elementos geométricos nos permiten restringir el espacio de búsqueda de los puntos homólogos, para aplicar una correspondencia basada en áreas sobre esta zona.

El MDE es el resultado al que se quiere llegar a modo de rejilla regular donde cada coordenada terreno X e Y con paso de malla especificado por el usuario contiene un valor de cota Z.

Posteriormente, se verá mas detenidamente esta última operación de búsqueda por correlación. Antes, veamos como implementar en Matlab las operaciones previas.

5.1.1. Parámetros de la cámara y carga de imágenes

Como lo primero es pedir al usuario la focal y las dimensiones del sensor tenemos dos formas de hacer esto: por comandos desde Matlab ejecutando las secuencias oportunas o mediante una interfaz gráfica de usuario.

En ambos casos la solución es sencilla, ya, que tan solo tenemos que crear las variables focal, alto y ancho del sensor, pedir al usuario estos valores por uno de los métodos anteriores y almacenar su resultado en la variable. Veamos el ejemplo para una secuencia:

```
function [f, foto, alto, ancho, altosensor, anchosensor]=CargaFoto()

%% CARGA DE FOTOS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Función CargaFoto
% Pide el nombre de las fotografias a cargar para realizar el
% MDE asi como la focal, única para todas ellas, alto y ancho del sensor
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Focal, ancho y alto

f=input('Dime la focal ');
%Ejemplo f=4.248;

altosensor=input('Dime alto ');

anchosensor=input('Dime ancho ');

% Selección de imagen

respuesta = 'S';
i=1;

while(respuesta == 's' || respuesta == 'S')

 [nombre, dir] = uigetfile({'*.jpg';'*.tiff';'*.mat';'*.*'},'Seleccione
imagen 1');
  ruta=[dir nombre];
  foto(i).ima=importdata(ruta);
 [alto ancho canal]=size(foto(i).ima);
  i=i+1;
  respuesta=input('Cargar otra imagen (S/N) ','s');
  while(respuesta ~= 's' && respuesta ~= 'S' && respuesta ~= 'n' &&
respuesta ~= 'N')
        respuesta=input('Cargar otra imagen (S/N) ','s');
  end
end
```

Figura 5-2. Función cargar foto

En la secuencia anterior se puede ver que no hay argumentos de entrada en la función *CargaFoto*() y hay 6 parámetros de salida que son: f, foto, alto, ancho, altosensor, anchosensor. No debe confundirse el alto con altosensor ya que uno hace referencia a la fotografía y otro al sensor.

En la función encontramos un bucle *while* que nos dice si queremos seguir cargando mas fotografías. En la interfaz grafica este bucle está limitado a 4 fotografías y de usar esta secuencia también ha de ser para 4 fotografías.

Recogiendo la misma función que no se utiliza en la interfaz gráfica pero que ilustra como realizar la tarea de cargar las fotos y recoger el valor de la focal, alto y ancho de la misma tenemos lo siguiente:

```
function [f, foto, alto, ancho]=CargaFoto()
CARGA DE FOTOS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Función CargaFoto
% Pide el nombre de las fotografías a cargar para realizar el MDE así como
% la focal, única para todas ellas
% Es un ejemplo de función que realiza lo mismo que se hace por medio de la
% interfaz
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Focal

prompt={'Valor de la focal:'};
name='Cargar foto'; % Título de la ventana
numlines=1;
defaultanswer={''};
options.Resize='on';
options.WindowStyle='normal';
options.Interpreter='tex';
f=str2double(cell2mat(inputdlg(prompt,name,numlines,defaultanswer,options)));

% Selección de imagen

for i=1:1:4
     if i==1
        [nombre, dir] =
uigetfile({'*.jpg';'*.tiff';'*.mat';'*.*'},'Seleccione imagen 1');
        ruta=[dir nombre];
        foto(i).ima=importdata(ruta);
     end
     if i==2
        [nombre, dir] =
uigetfile({'*.jpg';'*.tiff';'*.mat';'*.*'},'Seleccione imagen 2');
        ruta=[dir nombre];
        foto(i).ima=importdata(ruta);
     end
     if i==3
        [nombre, dir] =
uigetfile({'*.jpg';'*.tiff';'*.mat';'*.*'},'Seleccione imagen 3');
        ruta=[dir nombre];
        foto(i).ima=importdata(ruta);
     end
     if i==4
        [nombre, dir] =
uigetfile({'*.jpg';'*.tiff';'*.mat';'*.*'},'Seleccione imagen 4');
        ruta=[dir nombre];
        foto(i).ima=importdata(ruta);
     end
end
[alto ancho canal]=size(foto(i).ima);


end
```

Figura 5-3. Función focal, alto, ancho

5.1.2. Función captura de coordenadas de las dianas

Veamos a continuación la función que nos permite capturar en pantalla las coordenadas pixel que se transforman a fotocoordenadas:

```
function foto = Captura(foto, ParamOI)
FUNCION CAPTURA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Esta función permite pinchar sobre los puntos de apoyo de la imagen y
% devolver sus fotocoordenadas aplicando la función MiZoom. Tiene como
% argumentos de entrada la estructura foto y el array de los parámetros de la
% orientación interna
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for j=1:1:length(foto)

fotocoordenadas=[];

  for i=1:8
    titulo=['Foto ' num2str(j) ', Punto ' num2str(i)];
    a=figure('Name',titulo,'NumberTitle','off');

    imshow(foto(j).ima);
    [col fil]=ginput(1);
    subimagen=foto(j).ima(fil-35:fil+35,col-35:col+35);
    [fotocoordenadastemp]=MiZoom(subimagen,col-35,fil-35, ParamOI);
    fotocoordenadas=[fotocoordenadas;fotocoordenadastemp];
    close;
  end
foto(j).fotocoordenadas=fotocoordenadas;

end
```

Figura 5-4. Función captura

Tenemos como argumento de entrada los parámetros de la orientación interna que se calculan con otra función y además dentro de esta función se llama a otra función denominada MiZoom, estas funciones se exponen como se sigue:

```
function [fotocoordendas]=MiZoom(imagen,EsqSI_x,EsqSI_y, ParamOI)
FUNCION MIZOOM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% La función MiZoom a partir de una subimagen y los valores de las esquinas de
% esta junto con los parámetros de la orientación interna nos devuelve las
% fotocoordenadas
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


titulo=['Zoom diana'];
a=figure('Name',titulo,'NumberTitle','off');

imshow(imagen,'InitialMagnification',300); % Zoom de 3 veces mayor

[col, fil]=ginput(1);

col_mejorada=col+EsqSI_x;
fil_mejorada=fil+EsqSI_y;

A1=[];

    A1=[col_mejorada fil_mejorada 0 0 1 0
        0 0 col_mejorada fil_mejorada 0 1];


puntos = A1*ParamOI;

xp=[];

for i=1:2:length(puntos)
    xptemp=[puntos(i)];
    xp=[xp;xptemp];
end

yp=[];

for i=2:2:length(puntos)
    yptemp=[puntos(i)];
    yp=[yp;yptemp];
end

fotocoordendas = [xp,yp];

close
```

Figura 5-5. Función MiZoom

```
function [ParamOI, emc]=CalculaOI(alto, ancho, altosensor, anchosensor,
modo)
ORIENTACIÓN INTERNA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Función CalculaOI
% Calcula la orientación interna entre dos sistemas de coordenadas
% bidimensionales pixel a fotocoordenadas
%
% Datos de entrada caso directo
% - Sistema 1: Coordenadas pixel
% - Sistema 2: fotocoordenadas
% - Alto de la imagen
% - Ancho de la imagen
% En el modo inverso el sistema 1 y 2 se intercambian
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

switch modo
    case 'directo'

        Sist1=[1 alto
               ancho alto
               ancho 1
               1 1];

        Sist2=[-anchosensor/2 -altosensor/2
               anchosensor/2 -altosensor/2
               anchosensor/2 altosensor/2
               -anchosensor/2 altosensor/2];

    case 'inverso'

        Sist2=[1 alto
               ancho alto
               ancho 1
               1 1];

        Sist1=[-anchosensor/2 -altosensor/2
               anchosensor/2 -altosensor/2
               anchosensor/2 altosensor/2
               -anchosensor/2 altosensor/2];
end

% Transformación de coordenadas instrumentales
Col=Sist1(:,1);
Fil=Sist1(:,2);

% Continúa……
```

Figura 5-6. Función orientación interna

Continuación:

```
% ……Continuación

% Calcular matriz A
A1=[];
b=[];

        for i=1:length(Sist1)
            A1temp=[Col(i)  Fil(i) 0  0 1 0
                    0 0 Col(i) Fil(i) 0 1];

            btemp=[Sist2(i,1)
                   Sist2(i,2)];

            A1=[A1;A1temp];
            b=[b;btemp];
        end


ParamOI=A1\b; % Resolución por mínimos cuadrados


residuos=A1*ParamOI-b;
```

Figura 5-7. Función orientación interna

5.1.3. Función para la Orientación Externa

Tras la orientación interna y al disponer de fotocoordenadas de las dianas y de los puntos de apoyo se aplica el cálculo de la orientación externa con la siguiente función:

```
function [ParamOE, emcOE, informe] = CalculoExt(ptos_apoyo, foto, f, id)
Cálculo de la ORIENTACIÓN EXTERNA de la imagen
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Esta función calcula la orientación externa con el propósito de obtener
% los parametros de la misma siendo estos las coordenadas del punto de
% vista y los giros omega, phi y kappa
%
% Se hace uso de colinealidad y se linealiza la siguiente expresión para
% mediante un cálculo iterativo obtener el resultado cuando aquel converge
%
% F = xa*(r13*(XA-XO)+r23*(YA-YO)+r33*(ZA-ZO))+f*(r11*(XA-XO)+r21*(YA- …
% YO)+r31*(ZA-ZO))
% G = ya*(r13*(XA-XO)+r23*(YA-YO)+r33*(ZA-ZO))+f*(r12*(XA-XO)+r22*(YA- …
% YO)+r32*(ZA-ZO))
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
```

Figura 5-8. Función orientación externa

```
% Reasignación

XPA = ptos_apoyo(:,1);
YPA = ptos_apoyo(:,2);
ZPA = ptos_apoyo(:,3);

xp=foto(id).fotocoordenadas(:,1);
yp=foto(id).fotocoordenadas(:,2);

informe = [];

emc=1;
iteracion=0;

while (emc>=1 && iteracion<15)

    if iteracion == 0



        % Primera aproximacion a los valores de F y G
        % equivalente a:
        % Faprox=subs(F,{ome,phi,kap, XO, YO, …
        % ZO},{omeaprox,phiaprox,kapaprox, XOaprox, YOaprox, ZOaprox});
        % Gaprox=subs(G,{ome,phi,kap, XO, YO, …
        % ZO},{omeaprox,phiaprox,kapaprox, XOaprox, YOaprox, ZOaprox});

        XO=mean(XPA);
        YO=mean(YPA);
        ZO=mean(ZPA);

        ome = 0;
        phi = 0;
        kap = 0;
    end

   [R]=r(ome,phi,kap); % Función de la matriz de rotación

for i=1:1:length(xp)

    F(i) = (xp(i)*(R(1,3)*(XPA(i)-XO)+R(2,3)*(YPA(i)-YO)+R(3,3)*(ZPA(i)-…
ZO))+f*(R(1,1)*(XPA(i)-XO)+R(2,1)*(YPA(i)-YO)+R(3,1)*(ZPA(i)-ZO)));
    G(i) = (yp(i)*(R(1,3)*(XPA(i)-XO)+R(2,3)*(YPA(i)-YO)+R(3,3)*(ZPA(i)-…
ZO))+f*(R(1,2)*(XPA(i)-XO)+R(2,2)*(YPA(i)-YO)+R(3,2)*(ZPA(i)-ZO)));

end


% Trasponemos los vectores para tener el vector columna de los términos
independientes

Faprox = F';
Gaprox = G';
```

Figura 5-9. Función orientación externa

```
% Matriz de diseño para la orientación externa

j=1;
for i=1:2:(2*length(xp))

    % La matriz de diseño lleva desarrolladas las derivadas parciales de
    % las funciones F y G respecto de las incógnitas a calcular, se
    % muestran desarrolladas para poder compilar el programa ya que en
    % modo simbólico no funciona:
    %     syms XO YO ZO ome phi kap
    %
    %     Aext(i,1) = diff(F(j),XO);
    %     Aext(i,2) = diff(F(j),YO);
    %     Aext(i,3) = diff(F(j),ZO);
    %     Aext(i,4) = diff(F(j),ome);
    %     Aext(i,5) = diff(F(j),phi);
    %     Aext(i,6) = diff(F(j),kap);
    %
    %     Aext(i+1,1) = diff(G(j),XO);
    %     Aext(i+1,2) = diff(G(j),YO);
    %     Aext(i+1,3) = diff(G(j),ZO);
    %     Aext(i+1,4) = diff(G(j),ome);
    %     Aext(i+1,5) = diff(G(j),phi);
    %     Aext(i+1,6) = diff(G(j),kap);

    Aext(i,1) = (-xp(j)*(sin(kap)*sin(ome)-cos(kap)*cos(ome)*sin(phi))-…
f*cos(kap)*cos(phi));

    Aext(i,2) = (f*cos(phi)*sin(kap)-…
xp(j)*(cos(kap)*sin(ome)+cos(ome)*sin(kap)*sin(phi)));

    Aext(i,3) = (-f*sin(phi)-xp(j)*cos(ome)*cos(phi));

    Aext(i,4) = (-
xp(j)*((cos(ome)*sin(kap)+cos(kap)*sin(ome)*sin(phi))*(XO-…
XPA(j))+(cos(kap)*cos(ome)-sin(kap)*sin(ome)*sin(phi))*(YO-YPA(j))-…
cos(phi)*sin(ome)*(ZO-ZPA(j))));

    Aext(i,5) = (xp(j)*(cos(ome)*sin(phi)*(ZO-…
ZPA(j))+cos(kap)*cos(ome)*cos(phi)*(XO-XPA(j))-
cos(ome)*cos(phi)*sin(kap)*(YO-YPA(j)))-f*(cos(phi)*(ZO-ZPA(j))-
cos(kap)*sin(phi)*(XO-… XPA(j))+sin(kap)*sin(phi)*(YO-YPA(j))));

    Aext(i,6) = (f*(cos(kap)*cos(phi)*(YO-YPA(j))+cos(phi)*sin(kap)*(XO-…
XPA(j)))-xp(j)*((cos(kap)*sin(ome)+cos(ome)*sin(kap)*sin(phi))*(XO-XPA(j))-
… (sin(kap)*sin(ome)-cos(kap)*cos(ome)*sin(phi))*(YO-YPA(j))));(-…
f*(cos(ome)*sin(kap)+cos(kap)*sin(ome)*sin(phi))-yp(j)*(sin(kap)*sin(ome)-…
cos(kap)*cos(ome)*sin(phi)));


    Aext(i+1,1) = (-f*(cos(kap)*cos(ome)-sin(kap)*sin(ome)*sin(phi))-…
yp(j)*(cos(kap)*sin(ome)+cos(ome)*sin(kap)*sin(phi)));

    Aext(i+1,2) = (-f*(cos(kap)*cos(ome)-sin(kap)*sin(ome)*sin(phi))-…
yp(j)*(cos(kap)*sin(ome)+cos(ome)*sin(kap)*sin(phi)));
```

Figura 5-10. Función orientación externa

```
    Aext(i+1,3) = (f*cos(phi)*sin(ome)-yp(j)*cos(ome)*cos(phi));

    Aext(i+1,4) = (f*((sin(kap)*sin(ome)-cos(kap)*cos(ome)*sin(phi))*(XO-…
XPA(j))+(cos(kap)*sin(ome)+cos(ome)*sin(kap)*sin(phi))*(YO-…
YPA(j))+cos(ome)*cos(phi)*(ZO-ZPA(j)))-…
yp(j)*((cos(ome)*sin(kap)+cos(kap)*sin(ome)*sin(phi))*(XO-…
XPA(j))+(cos(kap)*cos(ome)-sin(kap)*sin(ome)*sin(phi))*(YO-YPA(j))-…
cos(phi)*sin(ome)*(ZO-ZPA(j))));

    Aext(i+1,5) = (yp(j)*(cos(ome)*sin(phi)*(ZO-…
ZPA(j))+cos(kap)*cos(ome)*cos(phi)*(XO-XPA(j))-
cos(ome)*cos(phi)*sin(kap)*(YO-… YPA(j)))-f*(sin(ome)*sin(phi)*(ZO-
ZPA(j))+cos(kap)*cos(phi)*sin(ome)*(XO-… XPA(j))-
cos(phi)*sin(kap)*sin(ome)*(YO-YPA(j))));

    Aext(i+1,6) = (-f*((cos(kap)*cos(ome)-sin(kap)*sin(ome)*sin(phi))*(XO-…
XPA(j))-(cos(ome)*sin(kap)+cos(kap)*sin(ome)*sin(phi))*(YO-YPA(j)))-…
yp(j)*((cos(kap)*sin(ome)+cos(ome)*sin(kap)*sin(phi))*(XO-XPA(j))-…
(sin(kap)*sin(ome)-cos(kap)*cos(ome)*sin(phi))*(YO-YPA(j))));

    j=j+1;
end

% Vector de términos independientes

L=[];
for i=1:1:(length(xp))

   Ltemp=[Faprox(i);Gaprox(i)];
   L=[L;Ltemp];
end

% Resolución por mínimos cuadrados

Xini=Aext\(-L);

XO = XO + Xini(1);
YO = YO + Xini(2);
ZO = ZO + Xini(3);
ome = ome + Xini(4);
phi = phi + Xini(5);
kap = kap + Xini(6);

[fil, col]=size(Aext);
red=fil-col;;
res=Aext*Xini+L;
emcOE=sqrt(res'*res/red);

informeOEtemp=[iteracion, XO, YO, ZO, ome*200/pi, phi*200/pi, kap*200/pi,
emcOE];
informe = [informe;informeOEtemp];

iteracion=iteracion+1;
end
ParamOE=[XO, YO, ZO, ome, phi, kap];
end
```

Figura 5-11. Función orientación

Dentro de esta función se hace uso de la función que calcula la matriz de rotación a partir de los ángulos omega, phi y kappa:

```
function [R]=r(ome,phi,kap)
Matriz de rotación
R=[cos(phi)*cos(kap) sin(ome)*sin(phi)*cos(kap)+cos(ome)*sin(kap) -
cos(ome)*sin(phi)*cos(kap)+sin(ome)*sin(kap)

        -cos(phi)*sin(kap) -sin(ome)*sin(phi)*sin(kap)+cos(ome)*cos(kap)
cos(ome)*sin(phi)*sin(kap)+sin(ome)*cos(kap)

        sin(phi) -sin(ome)*cos(phi) cos(ome)*cos(phi)];
```

Figura 5-12. Función orientación externa

5.2. Creación del MDE

En forma de diagrama o esquema se recoge el siguiente resumen explicativo de la elaboración del MDE:

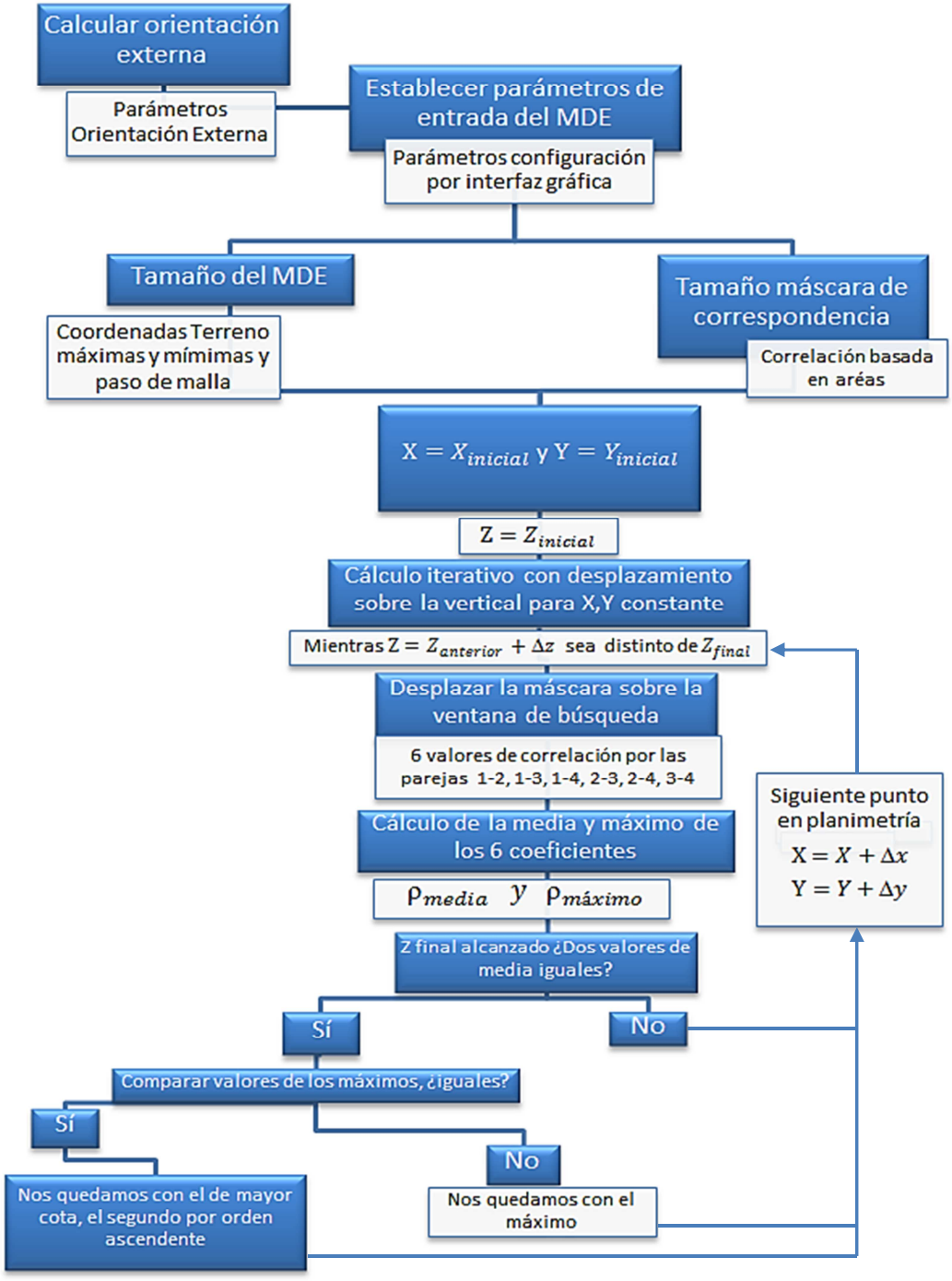


Figura 5-13. Diagrama para la elaboración del MDE

5.2.1. MDE por correspondencia basada en areas

Continuando con la exposición se muestra ahora una de las funciones mas importantes, como es la búsqueda de puntos homólogos por correlación y el cálculo de coordenadas terreno:

```
function [informeaux X Y Z] = MDEcorrespondencia(ParamOE, f, foto, altosensor,
anchosensor, I1, I2, J1, J2, Xini, Xmas, Yfinal, Ymas, Zmin, Zmax, mas,
TamMascara, handles)
Crear MDE por CORRESPONDENCIA BASADA EN AREAS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% En esta función se persigue obtener el MDE en base a los argumentos de
% entrada que son los parámetros de la orientación externa, la focal, el
% alto y ancho del sensor, unos índices para el recorrido del MDE y el
% tamaño de la máscara
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Tamaño de la imagen

[alto ancho canales] = size(foto(1).ima);

% Reasignación de variables

XO1=ParamOE(1).ima(1);
YO1=ParamOE(1).ima(2);
ZO1=ParamOE(1).ima(3);
ome1=ParamOE(1).ima(4);
phi1=ParamOE(1).ima(5);
kap1=ParamOE(1).ima(6);

XO2=ParamOE(2).ima(1);
YO2=ParamOE(2).ima(2);
ZO2=ParamOE(2).ima(3);
ome2=ParamOE(2).ima(4);
phi2=ParamOE(2).ima(5);
kap2=ParamOE(2).ima(6);

XO3=ParamOE(3).ima(1);
YO3=ParamOE(3).ima(2);
ZO3=ParamOE(3).ima(3);
ome3=ParamOE(3).ima(4);
phi3=ParamOE(3).ima(5);
kap3=ParamOE(3).ima(6);

XO4=ParamOE(4).ima(1);
YO4=ParamOE(4).ima(2);
ZO4=ParamOE(4).ima(3);
ome4=ParamOE(4).ima(4);
phi4=ParamOE(4).ima(5);
kap4=ParamOE(4).ima(6);
```

Figura 5-14. Función crear MDE por Correspondencia

```
% Reserva de memoria para la variable del informe de búsqueda

infor=[];
infor.foto=[];
informeaux=[];

for I=I1:1:I2
    for J=J1:1:J2
        informeaux(I,J).foto=[];
    end
end

% Reasignación para las matrices de rotación

R1=r(ome1,phi1,kap1);
R2=r(ome2,phi2,kap2);
R3=r(ome3,phi3,kap3);
R4=r(ome4,phi4,kap4);


% Puntos terreno para las fotocoordenadas dadas
Yfinal0=Yfinal+1;
Xini0=Xini-1;
progressbar % Inicio de una barra de progreso
for I=I1:1:I2
    I

       drawnow
       indice1=num2str(I)
       indice2=num2str(I2)

    set(handles.estadoactual,'String',['Localizando a nivel de imagen...  '…
indice1 '  de  ' indice2])
     drawnow

       progressbar(I/I2) % Actualización de la barra de progreso

    % Recorrido por los puntos del MDE de salida
    if I==I1
        Yfinal=Yfinal0-1;
    else
        Yfinal=Yfinal-Ymas;
    end
    Xini=Xini0;
```

Figura 5-15. Función crear MDE por Correspondencia

```
for J=J1:1:J2
        J;
        Xini=Xini+Xmas;

            for zi=Zmin:mas:Zmax;

                % Cálculo de las fotocoordenadas

                ftx1 = -f*(R1(1,1)*(Xini-XO1)+R1(2,1)*(Yfinal-…
YO1)+R1(3,1)*(zi-ZO1))/(R1(1,3)*(Xini-XO1)+R1(2,3)*(Yfinal-YO1)+R1(3,3)*(zi-
ZO1));
                fty1 = -f*(R1(1,2)*(Xini-XO1)+R1(2,2)*(Yfinal-…
YO1)+R1(3,2)*(zi-ZO1))/(R1(1,3)*(Xini-XO1)+R1(2,3)*(Yfinal-YO1)+R1(3,3)*(zi-
ZO1));

                ftx2 = -f*(R2(1,1)*(Xini-XO2)+R2(2,1)*(Yfinal-…
YO2)+R2(3,1)*(zi-ZO2))/(R2(1,3)*(Xini-XO2)+R2(2,3)*(Yfinal-YO2)+R2(3,3)*(zi-
ZO2));
                fty2 = -f*(R2(1,2)*(Xini-XO2)+R2(2,2)*(Yfinal-…
YO2)+R2(3,2)*(zi-ZO2))/(R2(1,3)*(Xini-XO2)+R2(2,3)*(Yfinal-YO2)+R2(3,3)*(zi-
ZO2));

                ftx3 = -f*(R3(1,1)*(Xini-XO3)+R3(2,1)*(Yfinal-YO3)+R3(3,1)*(zi-
ZO3))/(R3(1,3)*(Xini-XO3)+R3(2,3)*(Yfinal-YO3)+R3(3,3)*(zi-ZO3));
                fty3 = -f*(R3(1,2)*(Xini-XO3)+R3(2,2)*(Yfinal-…
YO3)+R3(3,2)*(zi-ZO3))/(R3(1,3)*(Xini-XO3)+R3(2,3)*(Yfinal-YO3)+R3(3,3)*(zi-
ZO3));

                ftx4 = -f*(R4(1,1)*(Xini-XO4)+R4(2,1)*(Yfinal-…
YO4)+R4(3,1)*(zi-ZO4))/(R4(1,3)*(Xini-XO4)+R4(2,3)*(Yfinal-YO4)+R4(3,3)*(zi-
ZO4));
                fty4 = -f*(R4(1,2)*(Xini-XO4)+R4(2,2)*(Yfinal-…
YO4)+R4(3,2)*(zi-ZO4))/(R4(1,3)*(Xini-XO4)+R4(2,3)*(Yfinal-YO4)+R4(3,3)*(zi-
ZO4));

                % Orientación interna en modo inverso para obtener
                % coordenadas píxel

                [ParamOI, emcOI]=CalculaOI(alto, ancho, altosensor, …
anchosensor, 'inverso');

            colfil=[ftx1 fty1 0 0 1 0
                    0 0 ftx1 fty1 0 1
                    ftx2 fty2 0 0 1 0
                    0 0 ftx2 fty2 0 1
                    ftx3 fty3 0 0 1 0
                    0 0 ftx3 fty3 0 1
                    ftx4 fty4 0 0 1 0
                    0 0 ftx4 fty4 0 1]*ParamOI;
```

Figura 5-16. Función crear MDE por Correspondencia

```
                col1=round(colfil(1));
                fil1=round(colfil(2));
                col2=round(colfil(3));
                fil2=round(colfil(4));
                col3=round(colfil(5));
                fil3=round(colfil(6));
                col4=round(colfil(7));
                fil4=round(colfil(8));

                % Máscara de correspondencia en la posición de cada foto

                mascara1=foto(1).ima(fil1-TamMascara:fil1+TamMascara,col1-…
TamMascara:col1+TamMascara);
                mascara2=foto(2).ima(fil2-TamMascara:fil2+TamMascara,col2-…
TamMascara:col2+TamMascara);
                mascara3=foto(3).ima(fil3-TamMascara:fil3+TamMascara,col3-…
TamMascara:col3+TamMascara);
                mascara4=foto(4).ima(fil4-TamMascara:fil4+TamMascara,col4-…
TamMascara:col4+TamMascara);

                % Factores de correlación por combinación de máscaras

                correlacion1=corr2(mascara1,mascara2);
                correlacion2=corr2(mascara1,mascara3);
                correlacion3=corr2(mascara1,mascara4);
                correlacion4=corr2(mascara2,mascara3);
                correlacion5=corr2(mascara2,mascara4);
                correlacion6=corr2(mascara3,mascara4);

                % Organización de los resultados en vectores y añadiendo la
                % media y el máximo de correlación

correlaciones=[correlacion1,correlacion2,correlacion3,correlacion4,…
correlacion5,correlacion6];
                correlacionmax=max(correlaciones);
                correlacion=median(correlaciones);

                        datos=[Xini Yfinal zi col1 fil1 col2 fil2 col3 fil3…
col4 fil4 correlacion correlacionmax];
                        infor.foto=[infor.foto;datos];
            end
        informeaux(I,J)=infor;
        infor=[];
        infor.foto=[];
        mcorrel2=informeaux(I,J).foto(:,13);
        mcorrel=informeaux(I,J).foto(:,12);

        % Búsqueda del máximo de media de correlación
        pos=find(mcorrel==max(mcorrel));

        % Comparativa en caso de haber mas de uno igual
        if length(pos)>1
            if mcorrel2(pos(1))>mcorrel2(pos(2))
                pos=pos(1);
            else
                pos=pos(2);
            end
        end
```

Figura 5-17. Función crear MDE por Correspondencia

```
        % Coordenadas terreno finales
        X(I,J)=informeaux(I,J).foto(pos,1);
        Y(I,J)=informeaux(I,J).foto(pos,2);
        Z(I,J)=informeaux(I,J).foto(pos,3);

        % Liberación de memoria
        mcorrel=[];
        mcorrel2=[];
        pos=[];

end

end
```

Figura 5-18. Función crear MDE por Correspondencia

5.2.2. Depuración

Para depurar datos del MDE se ha hecho la siguiente función:

```
function Zmed=depuracion(Z1, anchoMDE, altoMDE)
DEPURACIÓN DEL MDE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% La depuración se lleva a cabo con una ventana de 15x15, 225 elementos
% cuando se supera una cierta tolerancia. Así, si se da el caso de superar
% la misma se aplica un filtro de suavizado que aplica la mediana
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
I=0;
J=0;
for i=1+7:1:altoMDE-7
    I=I+1;
    J=0;
    for j=1+7:anchoMDE-7
        J=J+1;
        Z2=Z1(i-7:i+7,j-7:j+7);

Z2=[Z2(1,:),Z2(2,:),Z2(3,:),Z2(4,:),Z2(5,:),Z2(6,:),Z2(7,:),Z2(8,:),Z2(9,:
),Z2(10,:),Z2(11,:),Z2(12,:),Z2(13,:),Z2(14,:),Z2(15,:)];
        desZ2=std(Z2);
        medZ2=mean(Z2);
        if (Z1(i,j)<medZ2-2.5*desZ2) ||  (Z1(i,j)>medZ2+2.5*desZ2)
        Zmed(I,J)=median(Z2);
        else
            Zmed(I,J)=Z1(i,j);
        end
    end
end
```

Figura 5-19. Función depurar MDE

5.2.3. Ortofoto

La generación de la ortofoto está implementada con otra función:

```
function [orto orto1 orto2 orto3 orto4]=ortofoto(X, Y, Z, foto, f,
Param_OE, altosensor, anchosensor)
CREAR ORTOFOTO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% La función crea una ortofo por el método ascendente en la cual tenemos
% como parámetros de entrada los de la orientación externa, las
% coordenadas terreno en matrices X, Y, Z, la focal, fotografías y el alto
% y ancho del sensor.
% Como salida se devuelve una ortofo realizada con cada fotografía usando
% los distintos parámetros de la orientación externa y una fotografía que
% realiza el promedio del nivel digital en los distintos canales RGB
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Tamaño de foto y del modelo
[alto ancho canales] = size(foto(1).ima);
[alto2 ancho2] = size(Z);

% Resasignación

XO1=Param_OE(1).ima(1);
YO1=Param_OE(1).ima(2);
ZO1=Param_OE(1).ima(3);
ome1=Param_OE(1).ima(4);
phi1=Param_OE(1).ima(5);
kap1=Param_OE(1).ima(6);

XO2=Param_OE(2).ima(1);
YO2=Param_OE(2).ima(2);
ZO2=Param_OE(2).ima(3);
ome2=Param_OE(2).ima(4);
phi2=Param_OE(2).ima(5);
kap2=Param_OE(2).ima(6);

XO3=Param_OE(3).ima(1);
YO3=Param_OE(3).ima(2);
ZO3=Param_OE(3).ima(3);
ome3=Param_OE(3).ima(4);
phi3=Param_OE(3).ima(5);
kap3=Param_OE(3).ima(6);

XO4=Param_OE(4).ima(1);
YO4=Param_OE(4).ima(2);
ZO4=Param_OE(4).ima(3);
ome4=Param_OE(4).ima(4);
phi4=Param_OE(4).ima(5);
kap4=Param_OE(4).ima(6);

R1=r(ome1,phi1,kap1);
R2=r(ome2,phi2,kap2);
R3=r(ome3,phi3,kap3);
R4=r(ome4,phi4,kap4);
```

Figura 5-20. Función crear ortofoto

```
% Recorrido por el MDE
t=alto2+1;
for i=1:1:alto2
    t=t-1;
    for j=1:1:ancho2

        % Cálculo de fotocoordenadas en foto 1
ftx1 = -f*(R1(1,1)*(X(i,j)-XO1)+R1(2,1)*(Y(i,j)-YO1)+R1(3,1)*(Z(i,j)-…
ZO1))/(R1(1,3)*(X(i,j)-XO1)+R1(2,3)*(Y(i,j)-YO1)+R1(3,3)*(Z(i,j)-ZO1));
fty1 = -f*(R1(1,2)*(X(i,j)-XO1)+R1(2,2)*(Y(i,j)-YO1)+R1(3,2)*(Z(i,j)-…
ZO1))/(R1(1,3)*(X(i,j)-XO1)+R1(2,3)*(Y(i,j)-YO1)+R1(3,3)*(Z(i,j)-ZO1));

[ParamOI, emc]=CalculaOI(alto, ancho, altosensor, anchosensor, 'inverso');

                colfil=[ftx1 fty1 0 0 1 0
                        0 0 ftx1 fty1 0 1]*ParamOI;

                  % Niveles digitales obtenidos por el vecino más próximo con
                  % la función round() que redondea
                ND1=foto(1).ima(round(colfil(2)),round(colfil(1)),1);
                ND2=foto(1).ima(round(colfil(2)),round(colfil(1)),2);
                ND3=foto(1).ima(round(colfil(2)),round(colfil(1)),3);

                % Asignación de canales RGB en foto 1
                orto1(t,j,1)=ND1;
                orto1(t,j,2)=ND2;
                orto1(t,j,3)=ND3;
    end
end
1
% Recorrido por el MDE
t=alto2+1;
for i=1:1:alto2
    t=t-1;
    for j=1:1:ancho2

        % Cálculo de fotocoordenadas en foto 2
ftx2 = -f*(R2(1,1)*(X(i,j)-XO2)+R2(2,1)*(Y(i,j)-YO2)+R2(3,1)*(Z(i,j)-…
ZO2))/(R2(1,3)*(X(i,j)-XO2)+R2(2,3)*(Y(i,j)-YO2)+R2(3,3)*(Z(i,j)-ZO2));
fty2 = -f*(R2(1,2)*(X(i,j)-XO2)+R2(2,2)*(Y(i,j)-YO2)+R2(3,2)*(Z(i,j)-…
ZO2))/(R2(1,3)*(X(i,j)-XO2)+R2(2,3)*(Y(i,j)-YO2)+R2(3,3)*(Z(i,j)-ZO2));

[ParamOI, emc]=CalculaOI(alto, ancho, altosensor, anchosensor, 'inverso');

                colfil=[ftx2 fty2 0 0 1 0
                        0 0 ftx2 fty2 0 1]*ParamOI;

                % Niveles digitales obtenidos por el vecino más próximo con la
                % función round() que redondea
                ND1=foto(2).ima(round(colfil(2)),round(colfil(1)),1);
                ND2=foto(2).ima(round(colfil(2)),round(colfil(1)),2);
                ND3=foto(2).ima(round(colfil(2)),round(colfil(1)),3);
```

Figura 5-21. Función crear ortofoto

```
% Asignación de canales RGB en foto 2
                orto2(t,j,1)=ND1;
                orto2(t,j,2)=ND2;
                orto2(t,j,3)=ND3;
    end
end
2
% Recorrido por el MDE
t=alto2+1;
for i=1:1:alto2
    t=t-1;
    for j=1:1:ancho2

        % Cálculo de fotocoordenadas en foto 3
ftx3 = -f*(R3(1,1)*(X(i,j)-XO3)+R3(2,1)*(Y(i,j)-YO3)+R3(3,1)*(Z(i,j)-…
ZO3))/(R3(1,3)*(X(i,j)-XO3)+R3(2,3)*(Y(i,j)-YO3)+R3(3,3)*(Z(i,j)-ZO3));
fty3 = -f*(R3(1,2)*(X(i,j)-XO3)+R3(2,2)*(Y(i,j)-YO3)+R3(3,2)*(Z(i,j)-…
ZO3))/(R3(1,3)*(X(i,j)-XO3)+R3(2,3)*(Y(i,j)-YO3)+R3(3,3)*(Z(i,j)-ZO3));

[ParamOI, emc]=CalculaOI(alto, ancho, altosensor, anchosensor, 'inverso');

                colfil=[ftx3 fty3 0 0 1 0
                        0 0 ftx3 fty3 0 1]*ParamOI;

               % Niveles digitales obtenidos por el vecino más próximo con la
                % función round() que redondea
                ND1=foto(3).ima(round(colfil(2)),round(colfil(1)),1);
                ND2=foto(3).ima(round(colfil(2)),round(colfil(1)),2);
                ND3=foto(3).ima(round(colfil(2)),round(colfil(1)),3);

                % Asignación de canales RGB en foto 3
                orto3(t,j,1)=ND1;
                orto3(t,j,2)=ND2;
                orto3(t,j,3)=ND3;
    end
end
3
% Recorrido por el MDE
t=alto2+1;
for i=1:1:alto2
    t=t-1;
    for j=1:1:ancho2

        % Cálculo de fotocoordenadas en foto 4
ftx4 = -f*(R4(1,1)*(X(i,j)-XO4)+R4(2,1)*(Y(i,j)-YO4)+R4(3,1)*(Z(i,j)-…
ZO4))/(R4(1,3)*(X(i,j)-XO4)+R4(2,3)*(Y(i,j)-YO4)+R4(3,3)*(Z(i,j)-ZO4));
fty4 = -f*(R4(1,2)*(X(i,j)-XO4)+R4(2,2)*(Y(i,j)-YO4)+R4(3,2)*(Z(i,j)-…
ZO4))/(R4(1,3)*(X(i,j)-XO4)+R4(2,3)*(Y(i,j)-YO4)+R4(3,3)*(Z(i,j)-ZO4));
```

Figura 5-22. Función crear ortofoto

```
[ParamOI, emc]=CalculaOI(alto, ancho, altosensor, anchosensor, 'inverso');

                colfil=[ftx4 fty4 0 0 1 0
                        0 0 ftx4 fty4 0 1]*ParamOI;

                % Niveles digitales obtenidos por el vecino más próximo con la
                % función round() que redondea
                ND1=foto(4).ima(round(colfil(2)),round(colfil(1)),1);
                ND2=foto(4).ima(round(colfil(2)),round(colfil(1)),2);
                ND3=foto(4).ima(round(colfil(2)),round(colfil(1)),3);

                % Asignación de canales RGB en foto 4
                orto4(t,j,1)=ND1;
                orto4(t,j,2)=ND2;
                orto4(t,j,3)=ND3;
    end
end
4
% Recorrido del MDE y asignación de la media del nivel digital por canal RGB
for k=1:1:alto2
    for w=1:1:ancho2
        aux1=[orto1(k,w,1);orto2(k,w,1);orto3(k,w,1);orto4(k,w,1)];
        aux2=[orto1(k,w,2);orto2(k,w,2);orto3(k,w,2);orto4(k,w,2)];
        aux3=[orto1(k,w,3);orto2(k,w,3);orto3(k,w,3);orto4(k,w,3)];

        orto(k,w,1)=uint8(round(mean(aux1)));
        orto(k,w,2)=uint8(round(mean(aux2)));
        orto(k,w,3)=uint8(round(mean(aux3)));
    end
end
```

Figura 5-23. Función crear ortofoto

5.3. Funciones adicionales

5.3.1. Perfil longitudinal

Hay una función que se considera tiene salida gráfica. Se trata de la función con la que podemos graficar un perfil longitudinal:

```
function perfilLongitudinal(orto, Z)

figure('Name','Perfil longitudinal: Señalar extremos','NumberTitle','off')
imshow(orto);
hold on

perfil=[];

[col1 fil1]=ginput(1);

[col2 fil2]=ginput(1);

if fil2==fil1
    m1=(fil2-fil1)/(col2-col1);
    n1=fil1;
    x1=col1:1:col2;
    y1=n1;
    fila=round(y1);
    colum=round(x1);
    for i=1:1:length(fila)
            perfil=[perfil,Z(fila(i),colum(i))];
    end
    plot(x1,y1,'-g','LineWidth',2)
end
if col2==col1
    m2=(col2-col1)/(fil2-fil1);
    n2=col1;
    y1=fil1:1:fil2;
    x1=n2;
    fila=round(y1);
    colum=round(x1);
    for i=1:1:length(fila)
            perfil=[perfil,Z(fila(i),colum(i))];
    end
    plot(x1,y1,'-g','LineWidth',2)


else
    m=(fil2-fil1)/(col2-col1);
    n=fil1;

        if col1<col2 && abs(m)<1

            x1=col1:1:col2;
            y1=m*(x1-col1)+n;
            fila=round(y1);
            colum=round(x1);
```

Figura 5-24. Función perfil longitudinal

```
            for i=1:1:length(fila)
            perfil=[perfil,Z(fila(i),colum(i))];
            end
            plot(x1,y1,'-g','LineWidth',2)

        end
        if col1>col2 && abs(m)<1

            x1=col1:-1:col2;
            y1=m*(x1-col1)+n;
            fila=round(y1);
            colum=round(x1);
            for i=1:1:length(fila)
            perfil=[perfil,Z(fila(i),colum(i))];
            end
            plot(x1,y1,'-g','LineWidth',2)

        end
        if fil1<fil2 && abs(m)>=1

            y1=fil1:1:fil2;
            x1=(y1-n)/m+col1;
            fila=round(y1);
            colum=round(x1);
            for i=1:1:length(fila)
            perfil=[perfil,Z(fila(i),colum(i))];
            end
            plot(x1,y1,'-g','LineWidth',2)

        end
        if fil1>fil2 && abs(m)>=1

            y1=fil1:-1:fil2;
            x1=(y1-n)/m+col1;
            fila=round(y1);
            colum=round(x1);
            for i=1:1:length(fila)
            perfil=[perfil,Z(fila(i),colum(i))];
            end
            plot(x1,y1,'-g','LineWidth',2)

        end

        figure('Name','Perfil longitudinal','NumberTitle','off')

        plot(perfil,'DisplayName','Perfil
longitudinal','YDataSource','perfil');

end
```

Figura 5-25. Función perfil longitudinal

5.3.2. Función para crear líneas epipolares

Esta función es la que nos permite ejecutar la herramienta auxiliar para visualizar líneas epipolares a partir de un punto referencia en una de las imágenes.

```
function [col fil xx1 yy1 xx2 yy2 xx3 yy3 xx4 yy4] =
LineaEpipolar(ParamOE, f, alto, ancho, foto, id, altosensor,
anchosensor)

% Reasignacion de variables

XO1=ParamOE(1).ima(1);
YO1=ParamOE(1).ima(2);
ZO1=ParamOE(1).ima(3);
ome1=ParamOE(1).ima(4);
phi1=ParamOE(1).ima(5);
kap1=ParamOE(1).ima(6);

XO2=ParamOE(2).ima(1);
YO2=ParamOE(2).ima(2);
ZO2=ParamOE(2).ima(3);
ome2=ParamOE(2).ima(4);
phi2=ParamOE(2).ima(5);
kap2=ParamOE(2).ima(6);

XO3=ParamOE(3).ima(1);
YO3=ParamOE(3).ima(2);
ZO3=ParamOE(3).ima(3);
ome3=ParamOE(3).ima(4);
phi3=ParamOE(3).ima(5);
kap3=ParamOE(3).ima(6);

XO4=ParamOE(4).ima(1);
YO4=ParamOE(4).ima(2);
ZO4=ParamOE(4).ima(3);
ome4=ParamOE(4).ima(4);
phi4=ParamOE(4).ima(5);
kap4=ParamOE(4).ima(6);

R1=r(ome1,phi1,kap1);
R2=r(ome2,phi2,kap2);
R3=r(ome3,phi3,kap3);
R4=r(ome4,phi4,kap4);

titulo=['Foto referencia ' num2str(id)];
    a=figure('Name',titulo,'NumberTitle','off');


imshow(foto(id).ima);
[col fil]=ginput(1);

[ParamOI, emcOI]=CalculaOI(alto, ancho, altosensor, anchosensor,
'directo');
```

Figura 5-26. Función línea epipolar

```
subimagen=foto(id).ima(fil-35:fil+35,col-35:col+35);
[FT]=MiZoom(subimagen,col-35,fil-35, ParamOI);
 close
 [ParamOI, emcOI]=CalculaOI(alto, ancho, altosensor, anchosensor,
'inverso');
 cf=[FT(1) FT(2) 0 0 1 0
      0 0 FT(1) FT(2) 0 1]*ParamOI;
  col=cf(1);
  fil=cf(2);

x1=FT(1);
y1=FT(2);


%% Puntos terreno para las fotocoordenadas dadas

if id==1

XYZ1=R1*[x1;y1;-f]+[XO1;YO1;ZO1];
XYZ2=100*R1*[x1;y1;-f]+[XO1;YO1;ZO1];

elseif id==2

XYZ1=R2*[x1;y1;-f]+[XO2;YO2;ZO2];
XYZ2=100*R2*[x1;y1;-f]+[XO2;YO2;ZO2];

elseif id==3

XYZ1=R3*[x1;y1;-f]+[XO3;YO3;ZO3];
XYZ2=100*R3*[x1;y1;-f]+[XO3;YO3;ZO3];

elseif id==4

XYZ1=R4*[x1;y1;-f]+[XO4;YO4;ZO4];
XYZ2=100*R4*[x1;y1;-f]+[XO4;YO4;ZO4];

end


%% Fotocoordenadas en la segunda imagen para las coordenadas terreno

ftx11 = -f*(R1(1,1)*(XYZ1(1)-XO1)+R1(2,1)*(XYZ1(2)-
YO1)+R1(3,1)*(XYZ1(3)-ZO1))/(R1(1,3)*(XYZ1(1)-XO1)+R1(2,3)*(XYZ1(2)-
YO1)+R1(3,3)*(XYZ1(3)-ZO1));
fty11 = -f*(R1(1,2)*(XYZ1(1)-XO1)+R1(2,2)*(XYZ1(2)-
YO1)+R1(3,2)*(XYZ1(3)-ZO1))/(R1(1,3)*(XYZ1(1)-XO1)+R1(2,3)*(XYZ1(2)-
YO1)+R1(3,3)*(XYZ1(3)-ZO1));

ftx12 = -f*(R1(1,1)*(XYZ2(1)-XO1)+R1(2,1)*(XYZ2(2)-
YO1)+R1(3,1)*(XYZ2(3)-ZO1))/(R1(1,3)*(XYZ2(1)-XO1)+R1(2,3)*(XYZ2(2)-
YO1)+R1(3,3)*(XYZ2(3)-ZO1));
fty12 = -f*(R1(1,2)*(XYZ2(1)-XO1)+R1(2,2)*(XYZ2(2)-
YO1)+R1(3,2)*(XYZ2(3)-ZO1))/(R1(1,3)*(XYZ2(1)-XO1)+R1(2,3)*(XYZ2(2)-
YO1)+R1(3,3)*(XYZ2(3)-ZO1));
```

Figura 5-27. Función línea epipolar

```
ftx21 = -f*(R2(1,1)*(XYZ1(1)-XO2)+R2(2,1)*(XYZ1(2)-
YO2)+R2(3,1)*(XYZ1(3)-ZO2))/(R2(1,3)*(XYZ1(1)-XO2)+R2(2,3)*(XYZ1(2)-
YO2)+R2(3,3)*(XYZ1(3)-ZO2));
fty21 = -f*(R2(1,2)*(XYZ1(1)-XO2)+R2(2,2)*(XYZ1(2)-
YO2)+R2(3,2)*(XYZ1(3)-ZO2))/(R2(1,3)*(XYZ1(1)-XO2)+R2(2,3)*(XYZ1(2)-
YO2)+R2(3,3)*(XYZ1(3)-ZO2));

ftx22 = -f*(R2(1,1)*(XYZ2(1)-XO2)+R2(2,1)*(XYZ2(2)-
YO2)+R2(3,1)*(XYZ2(3)-ZO2))/(R2(1,3)*(XYZ2(1)-XO2)+R2(2,3)*(XYZ2(2)-
YO2)+R2(3,3)*(XYZ2(3)-ZO2));
fty22 = -f*(R2(1,2)*(XYZ2(1)-XO2)+R2(2,2)*(XYZ2(2)-
YO2)+R2(3,2)*(XYZ2(3)-ZO2))/(R2(1,3)*(XYZ2(1)-XO2)+R2(2,3)*(XYZ2(2)-
YO2)+R2(3,3)*(XYZ2(3)-ZO2));

ftx31 = -f*(R3(1,1)*(XYZ1(1)-XO3)+R3(2,1)*(XYZ1(2)-
YO3)+R3(3,1)*(XYZ1(3)-ZO3))/(R3(1,3)*(XYZ1(1)-XO3)+R3(2,3)*(XYZ1(2)-
YO3)+R3(3,3)*(XYZ1(3)-ZO3));
fty31 = -f*(R3(1,2)*(XYZ1(1)-XO3)+R3(2,2)*(XYZ1(2)-
YO3)+R3(3,2)*(XYZ1(3)-ZO3))/(R3(1,3)*(XYZ1(1)-XO3)+R3(2,3)*(XYZ1(2)-
YO3)+R3(3,3)*(XYZ1(3)-ZO3));

ftx32 = -f*(R3(1,1)*(XYZ2(1)-XO3)+R3(2,1)*(XYZ2(2)-
YO3)+R3(3,1)*(XYZ2(3)-ZO3))/(R3(1,3)*(XYZ2(1)-XO3)+R3(2,3)*(XYZ2(2)-
YO3)+R3(3,3)*(XYZ2(3)-ZO3));
fty32 = -f*(R3(1,2)*(XYZ2(1)-XO3)+R3(2,2)*(XYZ2(2)-
YO3)+R3(3,2)*(XYZ2(3)-ZO3))/(R3(1,3)*(XYZ2(1)-XO3)+R3(2,3)*(XYZ2(2)-
YO3)+R3(3,3)*(XYZ2(3)-ZO3));

ftx41 = -f*(R4(1,1)*(XYZ1(1)-XO4)+R4(2,1)*(XYZ1(2)-
YO4)+R4(3,1)*(XYZ1(3)-ZO4))/(R4(1,3)*(XYZ1(1)-XO4)+R4(2,3)*(XYZ1(2)-
YO4)+R4(3,3)*(XYZ1(3)-ZO4));
fty41 = -f*(R4(1,2)*(XYZ1(1)-XO4)+R4(2,2)*(XYZ1(2)-
YO4)+R4(3,2)*(XYZ1(3)-ZO4))/(R4(1,3)*(XYZ1(1)-XO4)+R4(2,3)*(XYZ1(2)-
YO4)+R4(3,3)*(XYZ1(3)-ZO4));

ftx42 = -f*(R4(1,1)*(XYZ2(1)-XO4)+R4(2,1)*(XYZ2(2)-
YO4)+R4(3,1)*(XYZ2(3)-ZO4))/(R4(1,3)*(XYZ2(1)-XO4)+R4(2,3)*(XYZ2(2)-
YO4)+R4(3,3)*(XYZ2(3)-ZO4));
fty42 = -f*(R4(1,2)*(XYZ2(1)-XO4)+R4(2,2)*(XYZ2(2)-
YO4)+R4(3,2)*(XYZ2(3)-ZO4))/(R4(1,3)*(XYZ2(1)-XO4)+R4(2,3)*(XYZ2(2)-
YO4)+R4(3,3)*(XYZ2(3)-ZO4));

 %% Paso de fotocoordenadas a coordenadas píxel

[ParamOI, emcOI]=CalculaOI(alto, ancho, altosensor, anchosensor,
'inverso');
```

Figura 5-28. Función línea epipolar

```
colfil=[ftx11 fty11 0 0 1 0
         0 0 ftx11 fty11 0 1
         ftx12 fty12 0 0 1 0
         0 0 ftx12 fty12 0 1
         ftx21 fty21 0 0 1 0
         0 0 ftx21 fty21 0 1
         ftx22 fty22 0 0 1 0
         0 0 ftx22 fty22 0 1
         ftx31 fty31 0 0 1 0
         0 0 ftx31 fty31 0 1
         ftx32 fty32 0 0 1 0
         0 0 ftx32 fty32 0 1
         ftx41 fty41 0 0 1 0
         0 0 ftx41 fty41 0 1
         ftx42 fty42 0 0 1 0
         0 0 ftx42 fty42 0 1]*ParamOI;

     col11=colfil(1);
     fil11=colfil(2);
     col12=colfil(3);
     fil12=colfil(4);
     col21=colfil(5);
     fil21=colfil(6);
     col22=colfil(7);
     fil22=colfil(8);
     col31=colfil(9);
     fil31=colfil(10);
     col32=colfil(11);
     fil32=colfil(12);
     col41=colfil(13);
     fil41=colfil(14);
     col42=colfil(15);
     fil42=colfil(16);

%%%%% 1

if fil12==fil11
    m11=(fil12-fil11)/(col12-col11);
    n11=fil11;
    xx1=1:1:ancho;
    yy1=n11;

end
if col12==col11
    m21=(col12-col11)/(fil12-fil11);
    n21=col11;
    yy1=1:1:alto;
    xx1=n21;

else
    m1=(fil12-fil11)/(col12-col11);
    n1=fil11;

        if col11<col12 && abs(m1)<1

            xx1=1:1:ancho;
            yy1=m1*(xx1-col11)+n1;

        end
```

Figura 5-29. Función línea epipolar

```
        if col11>col12 && abs(m1)<1

            xx1=ancho:-1:1;
            yy1=m1*(xx1-col11)+n1;

        end
        if fil11<fil12 && abs(m1)>=1

            yy1=1:1:alto;
            xx1=(yy1-n1)/m1+col11;

        end
        if fil11>fil12 && abs(m1)>=1

            yy1=alto:-1:1;
            xx1=(yy1-n1)/m1+col11;

        end
end

%%%%%%% 2

if fil22==fil21
    m12=(fil22-fil21)/(col22-col21);
    n12=fil21;
    xx2=1:1:ancho;
    yy2=n12;

end
if col22==col21
    m22=(col22-col21)/(fil22-fil21);
    n22=col21;
    yy2=1:1:alto;
    xx2=n22;

else
    m2=(fil22-fil21)/(col22-col21);
    n2=fil21;

        if col21<col22 && abs(m2)<1

            xx2=1:1:ancho;
            yy2=m2*(xx2-col21)+n2;

        end
        if col21>col22 && abs(m2)<1

            xx2=ancho:-1:1;
            yy2=m2*(xx2-col21)+n2;

        end
        if fil21<fil22 && abs(m2)>=1

            yy2=1:1:alto;
            xx2=(yy2-n2)/m2+col21;
        end
```

Figura 5-30. Función línea epipolar

```
        if fil21>fil22 && abs(m2)>=1

            yy2=alto:-1:1;
            xx2=(yy2-n2)/m2+col21;
        end
end

%%%%% 3

if fil32==fil31
    m13=(fil32-fil31)/(col32-col31);
    n13=fil31;
    xx3=1:1:ancho;
    yy3=n13;

end
if col32==col31
    m23=(col32-col31)/(fil32-fil31);
    n23=col31;
    yy3=1:1:alto;
    xx3=n23;

else
    m3=(fil32-fil31)/(col32-col31);
    n3=fil31;

        if col31<col32 && abs(m3)<1

            xx3=1:1:ancho;
            yy3=m3*(xx3-col31)+n3;
        end
        if col31>col32 && abs(m3)<1

            xx3=ancho:-1:1;
            yy3=m3*(xx3-col31)+n3;
        end
        if fil31<fil32 && abs(m3)>=1

            yy3=1:1:alto;
            xx3=(yy3-n3)/m3+col31;
        end
        if fil31>fil32 && abs(m3)>=1

            yy3=alto:-1:1;
            xx3=(yy3-n3)/m3+col31;
        end
end

%%%%% 4

if fil42==fil41
    m14=(fil42-fil41)/(col42-col41);
    n14=fil41;
    xx4=1:1:ancho;
    yy4=n14;
end
```

Figura 5-31. Función línea epipolar

```
if col42==col41
    m24=(col42-col41)/(fil42-fil41);
    n24=col41;
    yy4=1:1:alto;
    xx4=n24;

else
    m4=(fil42-fil41)/(col42-col41);
    n4=fil41;

        if col41<col42 && abs(m4)<1

            xx4=1:1:ancho;
            yy4=m4*(xx4-col41)+n4;
        end
        if col41>col42 && abs(m4)<1

            xx4=ancho:-1:1;
            yy4=m4*(xx4-col41)+n4;
        end
        if fil41<fil42 && abs(m4)>=1

            yy4=1:1:alto;
            xx4=(yy4-n4)/m4+col41;
        end
        if fil41>fil42 && abs(m4)>=1

            yy4=alto:-1:1;
            xx4=(yy4-n4)/m4+col41;
        end
end
```

Figura 5-32. Función línea epipolar

5.3.3. Vertical del lugar

La siguiente función nos devuelve los parámetros para poder proyectar las verticales del lugar sobre las imágenes. Esta tarea se lleva a cabo desde el menú ver perspectiva de la imagen:

```
function [xx1 yy1 xx2 yy2 xx3 yy3 xx4 yy4] = VerticalLocus(ParamOE,
ptos_apoyo, f, factor, alto, ancho, altosensor, anchosensor)

% Reasignacion

XPA = ptos_apoyo(:,1);
YPA = ptos_apoyo(:,2);
ZPA = ptos_apoyo(:,3);

Xmin=min(XPA);
Xmax=max(XPA);
diX=Xmax-Xmin;
```

Figura 5-33. Función vertical del lugar

```
Ymin=min(YPA);
Ymax=max(YPA);
diY=Ymax-Ymin;

Zmin=min(ZPA);
Zmax=max(ZPA);
Zmed=mean(ZPA);

%% Parámetros OE y rotaciones

XO1=ParamOE(1).ima(1);
YO1=ParamOE(1).ima(2);
ZO1=ParamOE(1).ima(3);
ome1=ParamOE(1).ima(4);
phi1=ParamOE(1).ima(5);
kap1=ParamOE(1).ima(6);

XO2=ParamOE(2).ima(1);
YO2=ParamOE(2).ima(2);
ZO2=ParamOE(2).ima(3);
ome2=ParamOE(2).ima(4);
phi2=ParamOE(2).ima(5);
kap2=ParamOE(2).ima(6);

XO3=ParamOE(3).ima(1);
YO3=ParamOE(3).ima(2);
ZO3=ParamOE(3).ima(3);
ome3=ParamOE(3).ima(4);
phi3=ParamOE(3).ima(5);
kap3=ParamOE(3).ima(6);

XO4=ParamOE(4).ima(1);
YO4=ParamOE(4).ima(2);
ZO4=ParamOE(4).ima(3);
ome4=ParamOE(4).ima(4);
phi4=ParamOE(4).ima(5);
kap4=ParamOE(4).ima(6);

R1=r(ome1,phi1,kap1);
R2=r(ome2,phi2,kap2);
R3=r(ome3,phi3,kap3);
R4=r(ome4,phi4,kap4);

%%%%

k=0;
l=0;
for i=Ymin:diY/5:Ymax
    l=l+1;
    k=0;
    for j=Xmin:diX/5:Xmax

        k=k+1;

        XYZsup=[i;j;Zmax+(20*factor)];
        XYZinf=[i;j;Zmin-(20*factor)];
```

Figura 5-34. Función vertical del lugar

```
        ftxsup1 = -f*(R1(1,1)*(XYZsup(1)-XO1)+R1(2,1)*(XYZsup(2)-
YO1)+R1(3,1)*(XYZsup(3)-ZO1))/(R1(1,3)*(XYZsup(1)-
XO1)+R1(2,3)*(XYZsup(2)-YO1)+R1(3,3)*(XYZsup(3)-ZO1));
        ftysup1 = -f*(R1(1,2)*(XYZsup(1)-XO1)+R1(2,2)*(XYZsup(2)-
YO1)+R1(3,2)*(XYZsup(3)-ZO1))/(R1(1,3)*(XYZsup(1)-
XO1)+R1(2,3)*(XYZsup(2)-YO1)+R1(3,3)*(XYZsup(3)-ZO1));

        ftxinf1 = -f*(R1(1,1)*(XYZinf(1)-XO1)+R1(2,1)*(XYZinf(2)-
YO1)+R1(3,1)*(XYZinf(3)-ZO1))/(R1(1,3)*(XYZinf(1)-
XO1)+R1(2,3)*(XYZinf(2)-YO1)+R1(3,3)*(XYZinf(3)-ZO1));
        ftyinf1 = -f*(R1(1,2)*(XYZinf(1)-XO1)+R1(2,2)*(XYZinf(2)-
YO1)+R1(3,2)*(XYZinf(3)-ZO1))/(R1(1,3)*(XYZinf(1)-
XO1)+R1(2,3)*(XYZinf(2)-YO1)+R1(3,3)*(XYZinf(3)-ZO1));



        ftxsup2 = -f*(R2(1,1)*(XYZsup(1)-XO2)+R2(2,1)*(XYZsup(2)-
YO2)+R2(3,1)*(XYZsup(3)-ZO2))/(R2(1,3)*(XYZsup(1)-
XO2)+R2(2,3)*(XYZsup(2)-YO2)+R2(3,3)*(XYZsup(3)-ZO2));
        ftysup2 = -f*(R2(1,2)*(XYZsup(1)-XO2)+R2(2,2)*(XYZsup(2)-
YO2)+R2(3,2)*(XYZsup(3)-ZO2))/(R2(1,3)*(XYZsup(1)-
XO2)+R2(2,3)*(XYZsup(2)-YO2)+R2(3,3)*(XYZsup(3)-ZO2));

        ftxinf2 = -f*(R2(1,1)*(XYZinf(1)-XO2)+R2(2,1)*(XYZinf(2)-
YO2)+R2(3,1)*(XYZinf(3)-ZO2))/(R2(1,3)*(XYZinf(1)-
XO2)+R2(2,3)*(XYZinf(2)-YO2)+R2(3,3)*(XYZinf(3)-ZO2));
        ftyinf2 = -f*(R2(1,2)*(XYZinf(1)-XO2)+R2(2,2)*(XYZinf(2)-
YO2)+R2(3,2)*(XYZinf(3)-ZO2))/(R2(1,3)*(XYZinf(1)-
XO2)+R2(2,3)*(XYZinf(2)-YO2)+R2(3,3)*(XYZinf(3)-ZO2));



        ftxsup3 = -f*(R3(1,1)*(XYZsup(1)-XO3)+R3(2,1)*(XYZsup(2)-
YO3)+R3(3,1)*(XYZsup(3)-ZO3))/(R3(1,3)*(XYZsup(1)-
XO3)+R3(2,3)*(XYZsup(2)-YO3)+R3(3,3)*(XYZsup(3)-ZO3));
        ftysup3 = -f*(R3(1,2)*(XYZsup(1)-XO3)+R3(2,2)*(XYZsup(2)-
YO3)+R3(3,2)*(XYZsup(3)-ZO3))/(R3(1,3)*(XYZsup(1)-
XO3)+R3(2,3)*(XYZsup(2)-YO3)+R3(3,3)*(XYZsup(3)-ZO3));

        ftxinf3 = -f*(R3(1,1)*(XYZinf(1)-XO3)+R3(2,1)*(XYZinf(2)-
YO3)+R3(3,1)*(XYZinf(3)-ZO3))/(R3(1,3)*(XYZinf(1)-
XO3)+R3(2,3)*(XYZinf(2)-YO3)+R3(3,3)*(XYZinf(3)-ZO3));
        ftyinf3 = -f*(R3(1,2)*(XYZinf(1)-XO3)+R3(2,2)*(XYZinf(2)-
YO3)+R3(3,2)*(XYZinf(3)-ZO3))/(R3(1,3)*(XYZinf(1)-
XO3)+R3(2,3)*(XYZinf(2)-YO3)+R3(3,3)*(XYZinf(3)-ZO3));



        ftxsup4 = -f*(R4(1,1)*(XYZsup(1)-XO4)+R4(2,1)*(XYZsup(2)-
YO4)+R4(3,1)*(XYZsup(3)-ZO4))/(R4(1,3)*(XYZsup(1)-
XO4)+R4(2,3)*(XYZsup(2)-YO4)+R4(3,3)*(XYZsup(3)-ZO4));
        ftysup4 = -f*(R4(1,2)*(XYZsup(1)-XO4)+R4(2,2)*(XYZsup(2)-
YO4)+R4(3,2)*(XYZsup(3)-ZO4))/(R4(1,3)*(XYZsup(1)-
XO4)+R4(2,3)*(XYZsup(2)-YO4)+R4(3,3)*(XYZsup(3)-ZO4));

        ftxinf4 = -f*(R4(1,1)*(XYZinf(1)-XO4)+R4(2,1)*(XYZinf(2)-
YO4)+R4(3,1)*(XYZinf(3)-ZO4))/(R4(1,3)*(XYZinf(1)-
XO4)+R4(2,3)*(XYZinf(2)-YO4)+R4(3,3)*(XYZinf(3)-ZO4));
        ftyinf4 = -f*(R4(1,2)*(XYZinf(1)-XO4)+R4(2,2)*(XYZinf(2)-
YO4)+R4(3,2)*(XYZinf(3)-ZO4))/(R4(1,3)*(XYZinf(1)-
XO4)+R4(2,3)*(XYZinf(2)-YO4)+R4(3,3)*(XYZinf(3)-ZO4));
```

Figura 5-35. Función vertical del lugar

```
%% Paso de fotocoordenadas a coordenadas píxel

[ParamOI, emcOI]=CalculaOI(alto, ancho, altosensor, anchosensor, 'inverso');

 colfil=[ftxsup1 ftysup1 0 0 1 0
         0 0 ftxsup1 ftysup1 0 1
         ftxinf1 ftyinf1 0 0 1 0
         0 0 ftxinf1 ftyinf1 0 1
         ftxsup2 ftysup2 0 0 1 0
         0 0 ftxsup2 ftysup2 0 1
         ftxinf2 ftyinf2 0 0 1 0
         0 0 ftxinf2 ftyinf2 0 1
         ftxsup3 ftysup3 0 0 1 0
         0 0 ftxsup3 ftysup3 0 1
         ftxinf3 ftyinf3 0 0 1 0
         0 0 ftxinf3 ftyinf3 0 1
         ftxsup4 ftysup4 0 0 1 0
         0 0 ftxsup4 ftysup4 0 1
         ftxinf4 ftyinf4 0 0 1 0
         0 0 ftxinf4 ftyinf4 0 1]*ParamOI;

     col11=colfil(1);
     fil11=colfil(2);
     col12=colfil(3);
     fil12=colfil(4);
     col21=colfil(5);
     fil21=colfil(6);
     col22=colfil(7);
     fil22=colfil(8);
     col31=colfil(9);
     fil31=colfil(10);
     col32=colfil(11);
     fil32=colfil(12);
     col41=colfil(13);
     fil41=colfil(14);
     col42=colfil(15);
     fil42=colfil(16);

     %%%%% 1
if fil12==fil11
    m11=(fil12-fil11)/(col12-col11);
    n11=fil11;
    xx1(k,l).v=col11:1:col12;
    yy1(k,l).v=n11;

end
if col12==col11
    m21=(col12-col11)/(fil12-fil11);
    n21=col11;
    yy1(k,l).v=fil11:1:fil12;
    xx1(k,l).v=n21;
else
    m1=(fil12-fil11)/(col12-col11);
    n1=fil11;

        if col11<col12 && abs(m1)<1
            xx1(k,l).v=col11:1:col12;
            yy1(k,l).v=m1*(xx1(k,l).v-col11)+n1;
        end
```

Figura 5-36. Función vertical del lugar

```
        if col11>col12 && abs(m1)<1
            xx1(k,l).v=col11:-1:col12;
            yy1(k,l).v=m1*(xx1(k,l).v-col11)+n1;
        end
        if fil11<fil12 && abs(m1)>=1

            yy1(k,l).v=fil11:1:fil12;
            xx1(k,l).v=(yy1(k,l).v-n1)/m1+col11;
        end
        if fil11>fil12 && abs(m1)>=1

            yy1(k,l).v=fil11:-1:fil12;
            xx1(k,l).v=(yy1(k,l).v-n1)/m1+col11;
        end
end

%%%%%%% 2

if fil22==fil21
    m12=(fil22-fil21)/(col22-col21);
    n12=fil21;
    xx2(k,l).v=col21:1:col22;
    yy2(k,l).v=n12;

end
if col22==col21
    m22=(col22-col21)/(fil22-fil21);
    n22=col21;
    yy2(k,l).v=fil21:1:fil22;
    xx2(k,l).v=n22;

else
    m2=(fil22-fil21)/(col22-col21);
    n2=fil21;

        if col21<col22 && abs(m2)<1

            xx2(k,l).v=col21:1:col22;
            yy2(k,l).v=m2*(xx2(k,l).v-col21)+n2;
        end
        if col21>col22 && abs(m2)<1

            xx2(k,l).v=col21:-1:col22;
            yy2(k,l).v=m2*(xx2(k,l).v-col21)+n2;
        end
        if fil21<fil22 && abs(m2)>=1

            yy2(k,l).v=fil21:1:fil22;
            xx2(k,l).v=(yy2(k,l).v-n2)/m2+col21;
        end
        if fil21>fil22 && abs(m2)>=1

            yy2(k,l).v=fil21:-1:fil22;
            xx2(k,l).v=(yy2(k,l).v-n2)/m2+col21;
        end
end
```

Figura 5-37. Función vertical del lugar

```
%%%%% 3

if fil32==fil31
    m13=(fil32-fil31)/(col32-col31);
    n13=fil31;
    xx3(k,l).v=col31:1:col32;
    yy3(k,l).v=n13;

end
if col32==col31
    m23=(col32-col31)/(fil32-fil31);
    n23=col31;
    yy3(k,l).v=fil31:1:fil32;
    xx3(k,l).v=n23;

else
    m3=(fil32-fil31)/(col32-col31);
    n3=fil31;

        if col31<col32 && abs(m3)<1

            xx3(k,l).v=col31:1:col32;
            yy3(k,l).v=m3*(xx3(k,l).v-col31)+n3;
        end
        if col31>col32 && abs(m3)<1

            xx3(k,l).v=col31:-1:col32;
            yy3(k,l).v=m3*(xx3(k,l).v-col31)+n3;
        end
        if fil31<fil32 && abs(m3)>=1

            yy3(k,l).v=fil31:1:fil32;
            xx3(k,l).v=(yy3(k,l).v-n3)/m3+col31;
        end
        if fil31>fil32 && abs(m3)>=1

            yy3(k,l).v=fil31:-1:fil32;
            xx3(k,l).v=(yy3(k,l).v-n3)/m3+col31;
        end
end

%%%%% 4

if fil42==fil41
    m14=(fil42-fil41)/(col42-col41);
    n14=fil41;
    xx4(k,l).v=col41:1:col42;
    yy4(k,l).v=n14;

end
if col42==col41
    m24=(col42-col41)/(fil42-fil41);
    n24=col41;
    yy4(k,l).v=fil41:1:fil42;
    xx4(k,l).v=n24;
```

Figura 5-38. Función vertical del lugar

```
else
    m4=(fil42-fil41)/(col42-col41);
    n4=fil41;

        if col41<col42 && abs(m4)<1

            xx4(k,l).v=col41:1:col42;
            yy4(k,l).v=m4*(xx4(k,l).v-col41)+n4;

        end
        if col41>col42 && abs(m4)<1

            xx4(k,l).v=col41:-1:col42;
            yy4(k,l).v=m4*(xx4(k,l).v-col41)+n4;

        end
        if fil41<fil42 && abs(m4)>=1

            yy4(k,l).v=fil41:1:fil42;
            xx4(k,l).v=(yy4(k,l).v-n4)/m4+col41;

        end
        if fil41>fil42 && abs(m4)>=1

            yy4(k,l).v=fil41:-1:fil42;
            xx4(k,l).v=(yy4(k,l).v-n4)/m4+col41;

        end
end
    end
end
```

Figura 5-39. Función vertical del lugar

5.4. Funciones previas descartadas

Las funciones mostradas hasta ahora han sido todas las funciones empleadas para el programa final. No obstante, antes de conseguir implementar el programa se pensaron en otras soluciones que suponía trabajar a nivel de imagen más que a nivel de objeto.

Con el propósito de que estas queden reflejadas se exponen a continuación las funciones:

```
function [informeaux] = LocalizacionCorrespondencia(Param_OE, f, foto, recta,
id0, id, I1, I2, J1, J2, mas, handles)
LOCALIZACIÓN POR CORRESPONDENCIA
% Tamaño de las fotos
[alto ancho canales] = size(foto(id).ima);

% Reasignación de variables

XO1=Param_OE(id0).ima(1);
YO1=Param_OE(id0).ima(2);
ZO1=Param_OE(id0).ima(3);
ome1=Param_OE(id0).ima(4);
phi1=Param_OE(id0).ima(5);
kap1=Param_OE(id0).ima(6);

XO2=Param_OE(id).ima(1);
YO2=Param_OE(id).ima(2);
ZO2=Param_OE(id).ima(3);
ome2=Param_OE(id).ima(4);
phi2=Param_OE(id).ima(5);
kap2=Param_OE(id).ima(6);

informe=[];
localizacion=[];
localizacion2=[];
coordenadas=[];
infor=[];
infor.foto=[];

R1=r(ome1,phi1,kap1);
R2=r(ome2,phi2,kap2);

% Puntos terreno para las fotocoordenadas dadas

for I=I1:1:I2


       indice1=num2str(I)
       indice2=num2str(I2)
       indice3=num2str(id0)
       indice4=num2str(id)
```

Figura 5-40. Función localización por correspondencia

```
for J=J1:1:J2

        % Columna y fila de la recta vecta en su punto inferior y superior
        % en dos fotografías
        filinf1=(recta(1,id0).lim(I,J).inf(2));
        colinf1=(recta(1,id0).lim(I,J).inf(1));
        filinf2=(recta(1,id).lim(I,J).inf(2));
        colinf2=(recta(1,id).lim(I,J).inf(1));

        filsup1=(recta(1,id0).lim(I,J).sup(2));
        colsup1=(recta(1,id0).lim(I,J).sup(1));
        filsup2=(recta(1,id).lim(I,J).sup(2));
        colsup2=(recta(1,id).lim(I,J).sup(1));

        % Pendiente y ordenada sobre el origen de las rectas
        m1=recta(1,id0).m(I,J);
        n1=(recta(1,id0).n(I,J));
        m2=recta(1,id).m(I,J);
        n2=(recta(1,id).n(I,J));

        % Análisis de inclinación de la recta para el bucle
        if colinf1<colsup1 && abs(m1)<1
            opcion1=[colinf1,1,colsup1];
            opcion=opcion1;
            cambio=0;
        end
        if colinf1>colsup1 && abs(m1)<1
            opcion2=[colinf1,-1,colsup1];
            opcion=opcion2;
            cambio=0;
        end
        if filinf1<filsup1 && abs(m1)>=1
            opcion3=[filinf1,1,filsup1];
            opcion=opcion3;
            cambio=1;
        end
        if filinf1>filsup1 && abs(m1)>=1
            opcion4=[filinf1,-1,filsup1];
            opcion=opcion4;
            cambio=1;
        end

        % Contador de búsqueda iniciado a cero y valor de correlación
        % mínimo iniciado a 0.8
        cont=0;
        valor=0.8;
        while cont<=(1+mas)
            for c1=opcion(1):opcion(2):opcion(3);

                f1=(m1*c1+n1);

                if cambio == 1
                    f1=c1;
                    c1=((f1-n1)/m1);

                end
```

Figura 5-41. Función localización por correspondencia

```
                % Cálculo de fotocoordenadas de cada punto de la vertical

                [ParamOI, emc]=CalculaOI(alto, ancho, 'directo');

                FT=[c1 f1 0 0 1 0
                    0 0 c1 f1 0 1]*ParamOI;

                x1=FT(1);
                y1=FT(2);

                XYZ1=1*R1*[x1;y1;-f]+[XO1;YO1;ZO1];
                XYZ2=100*R1*[x1;y1;-f]+[XO1;YO1;ZO1];

                % Fotocoordenadas de dos puntos pertenecientes a la línea
                % epipolar
                ftx1 = -f*(R2(1,1)*(XYZ1(1)-XO2)+R2(2,1)*(XYZ1(2)-…
YO2)+R2(3,1)*(XYZ1(3)-ZO2))/(R2(1,3)*(XYZ1(1)-XO2)+R2(2,3)*(XYZ1(2)-…
YO2)+R2(3,3)*(XYZ1(3)-ZO2));
                fty1 = -f*(R2(1,2)*(XYZ1(1)-XO2)+R2(2,2)*(XYZ1(2)-…
YO2)+R2(3,2)*(XYZ1(3)-ZO2))/(R2(1,3)*(XYZ1(1)-XO2)+R2(2,3)*(XYZ1(2)-…
YO2)+R2(3,3)*(XYZ1(3)-ZO2));

                ftx2 = -f*(R2(1,1)*(XYZ2(1)-XO2)+R2(2,1)*(XYZ2(2)-…
YO2)+R2(3,1)*(XYZ2(3)-ZO2))/(R2(1,3)*(XYZ2(1)-XO2)+R2(2,3)*(XYZ2(2)-…
YO2)+R2(3,3)*(XYZ2(3)-ZO2));
                fty2 = -f*(R2(1,2)*(XYZ2(1)-XO2)+R2(2,2)*(XYZ2(2)-…
YO2)+R2(3,2)*(XYZ2(3)-ZO2))/(R2(1,3)*(XYZ2(1)-XO2)+R2(2,3)*(XYZ2(2)-…
YO2)+R2(3,3)*(XYZ2(3)-ZO2));

                [ParamOI, emc]=CalculaOI(alto, ancho, 'inverso');

                colfil=[ftx1 fty1 0 0 1 0
                    0 0 ftx1 fty1 0 1
                    ftx2 fty2 0 0 1 0
                    0 0 ftx2 fty2 0 1]*ParamOI;

                % Pendiente y ordenada en el origen de la línea epipolar
                a=(colfil(4)-colfil(2))/(colfil(3)-colfil(1));
                b=-(colfil(4)-colfil(2))/(colfil(3)-
colfil(1))*colfil(1)+colfil(2);

                % Intersección de línea epipilar y vertical del lugar
                interseccion = (inv([1 -m2;1 -a])*[n2;b]);
                f12=round(f1);
                c12=round(c1);

                % Patrón de referencia
                mascara=foto(id0).ima(f12-20:f12+20,c12-20:c12+20);

                        i=interseccion(2);
                        j=interseccion(1);
                        i2=round(i);
                        j2=round(j);
```

Figura 5-42. Función localización por correspondencia

```
                        % Máscara de correspondencia
                          mascara2=foto(id).ima(j2-20:j2+20,i2-20:i2+20);
                             % Factor de correlación
                              correlacion=corr2(mascara,mascara2);

                              % Por la casi imposibilidad de encontrar un
                              % índice de correlación que sea igual a uno
                              % esto se tomará como un falso positivo
                              if correlacion==1
                                  correlacion=0;
                              end
                              % Organización de datos en un vector
                          datos=[I J c1 f1 i j correlacion];
                          if(correlacion>valor)
                             infor.foto=[infor.foto;datos];
                             cont=cont+1;
                          end
            end

% El bucle sigue hasta encontrar la cantidad deseada de puntos para el
% informe si hace falta se reduce el valor de correlacion mínimo
            if cont<=(2+mas)
                valor=valor-0.1;
            end
        end
        informeaux(I,J)=infor;
        infor=[];
        infor.foto=[];

    end
end
```

Figura 5-43. Función localización por correspondencia

La función hace uso de verticales del lugar que anteriormente se pensó en generarlas con una función de forma independiente. Esta función adoptaba la siguiente forma:

```
function [m, n, extremo] = ParamVertical(f, Param_OE, foto, id)
CÁLCULO DE LAS VERTICALES DEL LUGAR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Función que calcula la pendiente y ordenada sobre el origen de las
% verticales del lugar así como las coordenadas pixel de sus extremos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Calculo del tamaño de la imagen, el índice id nos permite elegir imagen
[alto ancho canales] = size(foto(id).ima);
```

Figura 5-44. Función cálculo verticales del lugar

```
% Reasignación

XO=Param_OE(1);
YO=Param_OE(2);
ZO=Param_OE(3);
ome=Param_OE(4);
phi=Param_OE(5);
kap=Param_OE(6);

R=r(ome,phi,kap);
k=0;
l=0;

% Puntos terreno para las fotocoordenadas dadas
% Dados  X minimo 1041 y Y minimo 1190 los cuales se pueden pasar como
% argumento si se desea
for i=1041:1:1090
    l=l+1;
    k=0;
    for j=1141:1:1190

        k=k+1;

% Z mínimo en 998 y máximo en 1021 tambien pueden ser pasados como
% argumentos en este ejemplo
        XYZsup=[i;j;1021];
        XYZinf=[i;j;998];


% Fotocoordenadas en la imagen para las coordenadas terreno
        ftxsup = -f*(R(1,1)*(XYZsup(1)-XO)+R(2,1)*(XYZsup(2)-…
YO)+R(3,1)*(XYZsup(3)-ZO))/(R(1,3)*(XYZsup(1)-XO)+R(2,3)*(XYZsup(2)-…
YO)+R(3,3)*(XYZsup(3)-ZO));
        ftysup = -f*(R(1,2)*(XYZsup(1)-XO)+R(2,2)*(XYZsup(2)-…
YO)+R(3,2)*(XYZsup(3)-ZO))/(R(1,3)*(XYZsup(1)-XO)+R(2,3)*(XYZsup(2)-…
YO)+R(3,3)*(XYZsup(3)-ZO));


        ftxinf = -f*(R(1,1)*(XYZinf(1)-XO)+R(2,1)*(XYZinf(2)-…
YO)+R(3,1)*(XYZinf(3)-ZO))/(R(1,3)*(XYZinf(1)-XO)+R(2,3)*(XYZinf(2)-
YO)+R(3,3)*(XYZinf(3)-ZO));
        ftyinf = -f*(R(1,2)*(XYZinf(1)-XO)+R(2,2)*(XYZinf(2)-…
YO)+R(3,2)*(XYZinf(3)-ZO))/(R(1,3)*(XYZinf(1)-XO)+R(2,3)*(XYZinf(2)-…
YO)+R(3,3)*(XYZinf(3)-ZO));
% Paso de fotocoordenadas a coordenadas píxel
[ParamOI, emc]=CalculaOI(alto, ancho, 'inverso');

     colfil=[ftxsup ftysup 0 0 1 0
             0 0 ftxsup ftysup 0 1
             ftxinf ftyinf 0 0 1 0
             0 0 ftxinf ftyinf 0 1]*ParamOI;
```

Figura 5-45. Función cálculo verticales del lugar

```
extremo(k,l).sup=([colfil(1),colfil(2)]);
 extremo(k,l).inf=([colfil(3),colfil(4)]);

m(k,l)=(colfil(4)-colfil(2))/(colfil(3)-colfil(1));
n(k,l)=-(colfil(4)-colfil(2))/(colfil(3)-colfil(1))*colfil(1)+colfil(2);
    end
end

end
```

Figura 5-46. Función cálculo verticales del lugar

Siguiendo con la función localización por correspondencia se hace uso de la siguiente función posterior:

```
function [homologos] = LocalizacionHomologos4fotos(Param_OE, f, foto,
informeaux, num, I1, I2, J1, J2, handles)
LOCALIZAR HOMOLOGOS DE 4 FOTOS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Disponiendo del informe auxiliar calculado anteriormente se va a generar
% una matriz compuesta por aquellos elementos que tienen en la imagen en la
% que se tomo el patrón, las fotocoordenadas mas cercanas entre si para
% distintas parejas de imágenes
%
% Es decir, si es la fotografía 1 en la que se toma el patrón los puntos
% homólogos en las fotografías 2, 3, 4 pueden tomar distintas
% fotocoordenadas al estar en distintas posiciones pero la fotocoordenada
% en la fotografía 1 debe ser la misma para todos en el mejor de los casos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Datos auxiliares

[alto ancho canales] = size(foto(1).ima);

% Inicialización de matrices (reserva de memoria)

homologos=[];
homologos1=[];
homologos2=[];


for I=I1:1:I2
    for J=J1:1:J2
        homologos(I,J).foto=[];
        homologos1(I,J).foto=[];
        homologos2(I,J).foto=[];
    end
end
```

Figura 5-47. Función localizar homólogos 4 fotos

```
% Calculos coordenadas píxeles
for I=I1:1:I2
    for J=J1:1:J2

% Tamaño de los informes para movernos en un bucle
        [u1,v1]=size(informeaux(1).pareja(I,J).foto);
        [u2,v2]=size(informeaux(2).pareja(I,J).foto);
        [u3,v3]=size(informeaux(3).pareja(I,J).foto);

        for i=1:1:u1
            for j=1:1:u2
                for k=1:1:u3

% Calculamos distancias o separación de filas y columnas recogidas en el informe auxliar
                        d1=abs(informeaux(1).pareja(I,J).foto(i,3)-
informeaux(2).pareja(I,J).foto(j,3))+abs(informeaux(1).pareja(I,J).foto(i,3)-
informeaux(3).pareja(I,J).foto(k,3));
                        d2=abs(informeaux(1).pareja(I,J).foto(i,4)-
informeaux(2).pareja(I,J).foto(j,4))+abs(informeaux(1).pareja(I,J).foto(i,4)-
informeaux(3).pareja(I,J).foto(k,4));
                        d3=d1+d2;

                        % Nos quedamos con los valores de separación mínimo
                        if(i==1 && j==1 && k==1)
                            d=d3;
                            f1=i;
                            f2=j;
                            f3=k;
                        else
                            if d3<d
                                d=d3;
                                f1=i;
                                f2=j;
                                f3=k;
                            end
                        end
                end
            end
        end

       % Con la posición de los mínimos guardamos los mismos en una matriz
       % llamada homólogos
        homologos(I,J).foto=[[informeaux(1).pareja(I,J).foto(f1,3), …
informeaux(1).pareja(I,J).foto(f1,4), informeaux(1).pareja(I,J).foto(f1,5), …
informeaux(1).pareja(I,J).foto(f1,6), informeaux(1).pareja(I,J).foto(f1,7)];
                        [informeaux(2).pareja(I,J).foto(f2,3),
informeaux(2).pareja(I,J).foto(f2,4), informeaux(2).pareja(I,J).foto(f2,5), …
informeaux(2).pareja(I,J).foto(f2,6), informeaux(2).pareja(I,J).foto(f2,7)];
                        [informeaux(3).pareja(I,J).foto(f3,3),…
informeaux(3).pareja(I,J).foto(f3,4), informeaux(3).pareja(I,J).foto(f3,5),…
informeaux(3).pareja(I,J).foto(f3,6), informeaux(3).pareja(I,J).foto(f3,7)]];
    end
end
```

Figura 5-48. Función loclizar homólogos 4 fotos

El siguiente paso que se propuso se implementó con otra función:

```
function [homologos2] = LocalizacionHomologosTerr(Param_OE, f, foto,
informeaux, homologos, num, I1, I2, J1, J2, handles)
PASAR A TERRENO DESPUES DE LOCALIZACION DE HOMOLOGOS
% Datos auxiliares

[alto ancho canales] = size(foto(1).ima);

% Haremos referencia a las parejas en cuanto a cual se tomó como referencia
% por medio de la variable num

if num==1


XO1=Param_OE(1).ima(1);
YO1=Param_OE(1).ima(2);
ZO1=Param_OE(1).ima(3);
ome1=Param_OE(1).ima(4);
phi1=Param_OE(1).ima(5);
kap1=Param_OE(1).ima(6);

XO2=Param_OE(2).ima(1);
YO2=Param_OE(2).ima(2);
ZO2=Param_OE(2).ima(3);
ome2=Param_OE(2).ima(4);
phi2=Param_OE(2).ima(5);
kap2=Param_OE(2).ima(6);

XO3=Param_OE(3).ima(1);
YO3=Param_OE(3).ima(2);
ZO3=Param_OE(3).ima(3);
ome3=Param_OE(3).ima(4);
phi3=Param_OE(3).ima(5);
kap3=Param_OE(3).ima(6);

XO4=Param_OE(4).ima(1);
YO4=Param_OE(4).ima(2);
ZO4=Param_OE(4).ima(3);
ome4=Param_OE(4).ima(4);
phi4=Param_OE(4).ima(5);
kap4=Param_OE(4).ima(6);

end
if num==2

XO1=Param_OE(2).ima(1);
YO1=Param_OE(2).ima(2);
ZO1=Param_OE(2).ima(3);
ome1=Param_OE(2).ima(4);
phi1=Param_OE(2).ima(5);
kap1=Param_OE(2).ima(6);
```

Figura 5-49. Función localizar homólogos en terreno

```
XO2=Param_OE(1).ima(1);
YO2=Param_OE(1).ima(2);
ZO2=Param_OE(1).ima(3);
ome2=Param_OE(1).ima(4);
phi2=Param_OE(1).ima(5);
kap2=Param_OE(1).ima(6);

XO3=Param_OE(3).ima(1);
YO3=Param_OE(3).ima(2);
ZO3=Param_OE(3).ima(3);
ome3=Param_OE(3).ima(4);
phi3=Param_OE(3).ima(5);
kap3=Param_OE(3).ima(6);

XO4=Param_OE(4).ima(1);
YO4=Param_OE(4).ima(2);
ZO4=Param_OE(4).ima(3);
ome4=Param_OE(4).ima(4);
phi4=Param_OE(4).ima(5);
kap4=Param_OE(4).ima(6);

end

if num==3

XO1=Param_OE(3).ima(1);
YO1=Param_OE(3).ima(2);
ZO1=Param_OE(3).ima(3);
ome1=Param_OE(3).ima(4);
phi1=Param_OE(3).ima(5);
kap1=Param_OE(3).ima(6);

XO2=Param_OE(1).ima(1);
YO2=Param_OE(1).ima(2);
ZO2=Param_OE(1).ima(3);
ome2=Param_OE(1).ima(4);
phi2=Param_OE(1).ima(5);
kap2=Param_OE(1).ima(6);

XO3=Param_OE(2).ima(1);
YO3=Param_OE(2).ima(2);
ZO3=Param_OE(2).ima(3);
ome3=Param_OE(2).ima(4);
phi3=Param_OE(2).ima(5);
kap3=Param_OE(2).ima(6);

XO4=Param_OE(4).ima(1);
YO4=Param_OE(4).ima(2);
ZO4=Param_OE(4).ima(3);
ome4=Param_OE(4).ima(4);
phi4=Param_OE(4).ima(5);
kap4=Param_OE(4).ima(6);

end
```

Figura 5-50. Función localizar homólogos en terreno

```
if num==4

XO1=Param_OE(4).ima(1);
YO1=Param_OE(4).ima(2);
ZO1=Param_OE(4).ima(3);
ome1=Param_OE(4).ima(4);
phi1=Param_OE(4).ima(5);
kap1=Param_OE(4).ima(6);

XO2=Param_OE(1).ima(1);
YO2=Param_OE(1).ima(2);
ZO2=Param_OE(1).ima(3);
ome2=Param_OE(1).ima(4);
phi2=Param_OE(1).ima(5);
kap2=Param_OE(1).ima(6);

XO3=Param_OE(2).ima(1);
YO3=Param_OE(2).ima(2);
ZO3=Param_OE(2).ima(3);
ome3=Param_OE(2).ima(4);
phi3=Param_OE(2).ima(5);
kap3=Param_OE(2).ima(6);

XO4=Param_OE(3).ima(1);
YO4=Param_OE(3).ima(2);
ZO4=Param_OE(3).ima(3);
ome4=Param_OE(3).ima(4);
phi4=Param_OE(3).ima(5);
kap4=Param_OE(3).ima(6);

end

R1=r(ome1,phi1,kap1);
R2=r(ome2,phi2,kap2);
R3=r(ome3,phi3,kap3);
R4=r(ome4,phi4,kap4);

% Inicialización de matrices (reserva de memoria)

homologos1=[];
homologos2=[];


for I=I1:1:I2
    for J=J1:1:J2

        homologos1(I,J).foto=[];
        homologos2(I,J).foto=[];

    end

end
```

Figura 5-51. Función localizar homólogos en terreno

```
% Cálculos coordenadas terreno
for I=I1:1:I2
    for J=J1:1:J2

        for i=1:1:3
            [ParamOI, emc]=CalculaOI(alto, ancho, 'directo');

            FT=[homologos(I,J).foto(i,1) homologos(I,J).foto(i,2) 0 0 1 0
                0 0 homologos(I,J).foto(i,1) homologos(I,J).foto(i,2) 0 1
                homologos(I,J).foto(i,3) homologos(I,J).foto(i,4) 0 0 1 0
                0 0 homologos(I,J).foto(i,3) homologos(I,J).foto(i,4) 0 …
1]*ParamOI;

    x1=FT(1);
    y1=FT(2);
    x2=FT(3);
    y2=FT(4);

    % En función del valor de bucle i vamos haciendo un cálculo de
    % coordenadas terreno

    a=R1*[x1;y1;-f];
    if i==1
    c=R2*[x2;y2;-f];
    elseif i==2
    c=R3*[x2;y2;-f];
    elseif i==3
    c=R4*[x2;y2;-f];

    A=[1 0 0 -a(1) 0
       0 1 0 -a(2) 0
       0 0 1 -a(3) 0
       1 0 0 0 -c(1)
       0 1 0 0 -c(2)
       0 0 1 0 -c(3)];

    if i==1
    L=[XO1;YO1;ZO1;XO2;YO2;ZO2];
    elseif i==2
    L=[XO1;YO1;ZO1;XO3;YO3;ZO3];
    elseif i==3
    L=[XO1;YO1;ZO1;XO4;YO4;ZO4];

% Resolvemos por mínimos cuadrados
    X=A\L;
    residuos=A*X-L;
    emc=std(residuos);

    homologos1(I,J).foto=[homologos1(I,J).foto;X(1),X(2),X(3),X(4),X(5),emc];
        end

        % Acumulamos todas las coordenadas obtenidas por combinación de
        % parejas en la matriz homólogos2

        homologos2(I,J).foto=[homologos(I,J).foto,homologos1(I,J).foto];
    end
end
```

Figura 5-52. Función localizar homólogos en terreno

Por último, se encuentra la función que permite obtener coordenadas finales y estadísticos como la media y desviación típica:

```
function [CoordenadasTerreno] = Zfinal4fotos(Param_OE, f, foto, homologos, I1,
I2, J1, J2)
COORDENADAS TERRENO FINALES
% Inicialización de matrices (reserva de memoria)
CoordenadasTerreno=[];
for I=I1:1:I2
    for J=J1:1:J2
        CoordenadasTerreno(I,J).foto=[];
    end
end
% Cálculos coordenadas terreno
for I=I1:1:I2
    for J=J1:1:J2
% Multiplicamos por 10 redondeamos y dividimos por 10 para quedarnos con un
% solo decimal
            x1=round(homologos(I,J).foto(1,6)*10);
            x2=round(homologos(I,J).foto(2,6)*10);
            x3=round(homologos(I,J).foto(3,6)*10);

            x=[x1;x2;x3];
            M1=mode(x)/10;

            y1=round(homologos(I,J).foto(1,7)*10);
            y2=round(homologos(I,J).foto(2,7)*10);
            y3=round(homologos(I,J).foto(3,7)*10);

            y=[y1;y2;y3];
            M2=mode(y)/10;

            z1=round(homologos(I,J).foto(1,8)*10);
            z2=round(homologos(I,J).foto(2,8)*10);
            z3=round(homologos(I,J).foto(3,8)*10);
            z=[z1;z2;z3];

            % Moda
            M3=mode(z)/10;
            % Media
            M4=mean(z)/10;
            % Desviación tipica
            M5=std(z)/10;
            % Matriz final con coordenadas terreno dadas por la moda y
            % otros estadísticos como la media y desviación típica
            CoordenadasTerreno(I,J).foto=[[M1,M2,M3,M4,M5]];
    end
end
```

Figura 5-53. Función coordenadas terreno finales

Estas últimas funciones se descartaron ya que no daban tan buenos resultados y su ejecución era demasiado lenta. Además, lleva consigo gran complejidad por el empleo de la gran cantidad de código y funciones.

CAPÍTULO 6. RESULTADOS

6.1. Resultados con las fotos de "CHISME"

Al realizar las primeras ejecuciones con el software elaborado se observa que el resultado que ofrece se asemeja a la realidad. Veamos a continuación los resultados para las distintas rugosidades del terreno.

6.1.1. Zona lisa

Se muestra en las siguientes figuras un MDE con tamaño 250x250 dado un paso de malla de 2 mm en planimetría y incrementos sobre la vertical de 1 mm:

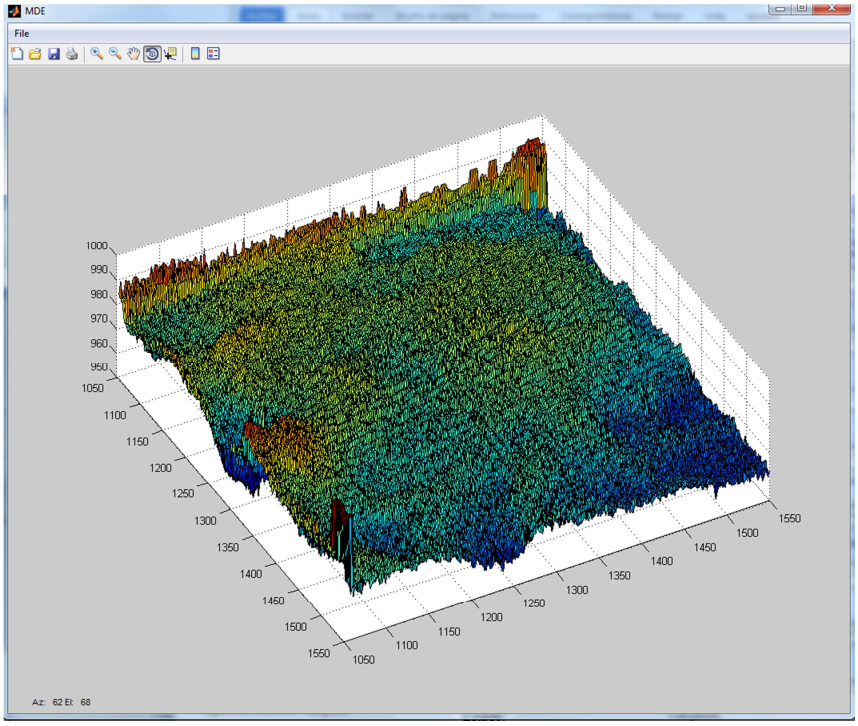


Figura 6-1. Modelo zona lisa 1

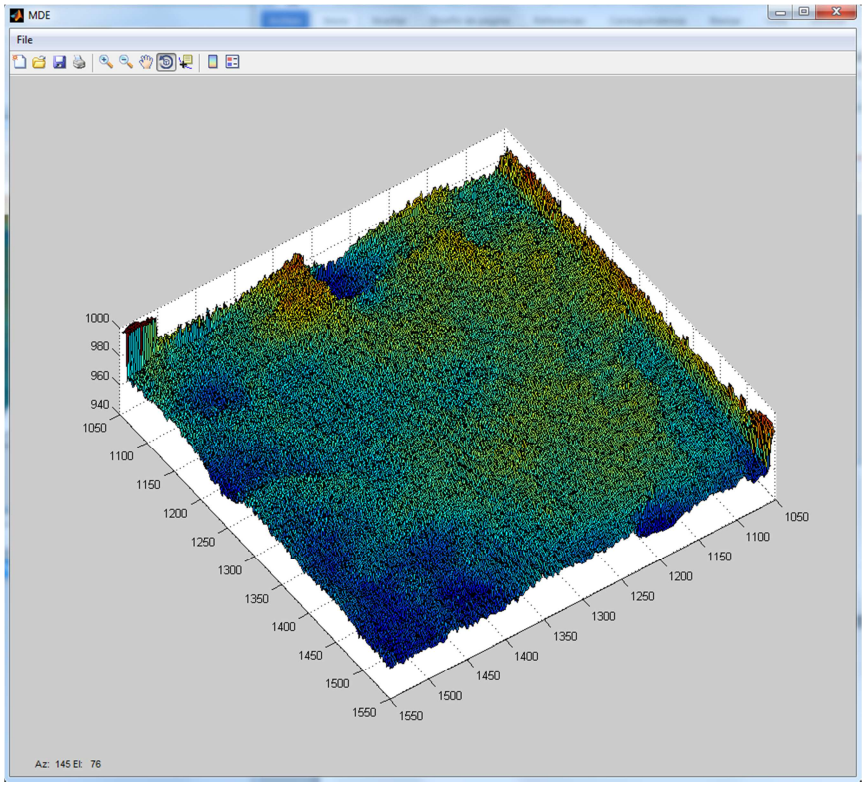


Figura 6-2. Modelo zona lisa 2

Otros ejemplos son los que observamos al renderizar y aplicar una ortofoto, para el caso anterior del MDE de tamaño 250x250:

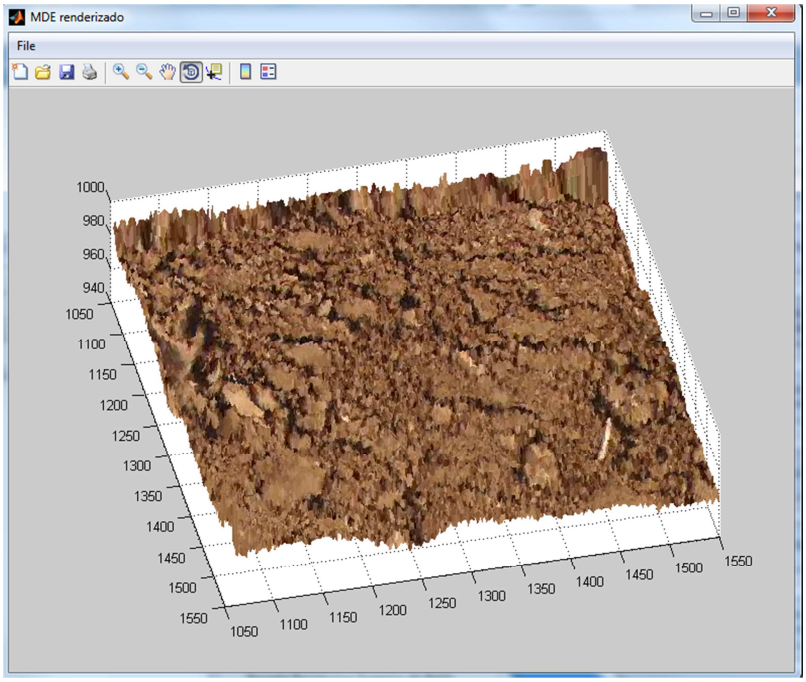


Figura 6-3. Modelo zona lisa renderizado 1

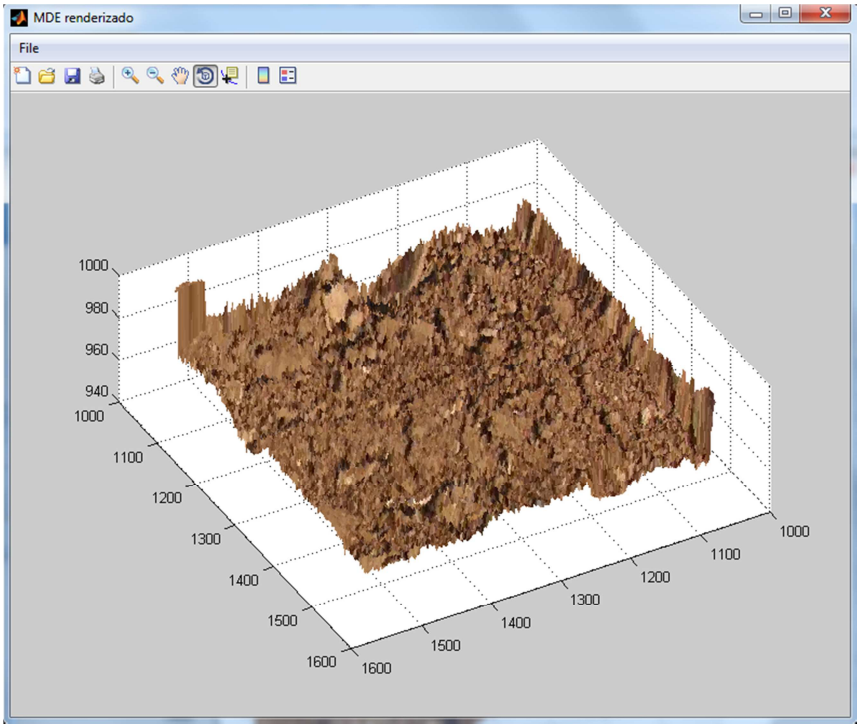


Figura 6-4. Modelo zona lisa renderizado 2

La cota mínima de este modelo es de 954 mm y la máxima de 1000 mm, sabiendo que la cota local del marco es de 1000 mm.

Las ortofotos obtenidas con cada fotografía y la ortofoto promedio han sido:

Ortofoto 1

Ortofoto 2

Ortofoto 3

Ortofoto 4

Ortofoto promedio

Figura 6-5. Orfotos zona lisa

Un ejemplo de perfil longitudinal para esta zona es:

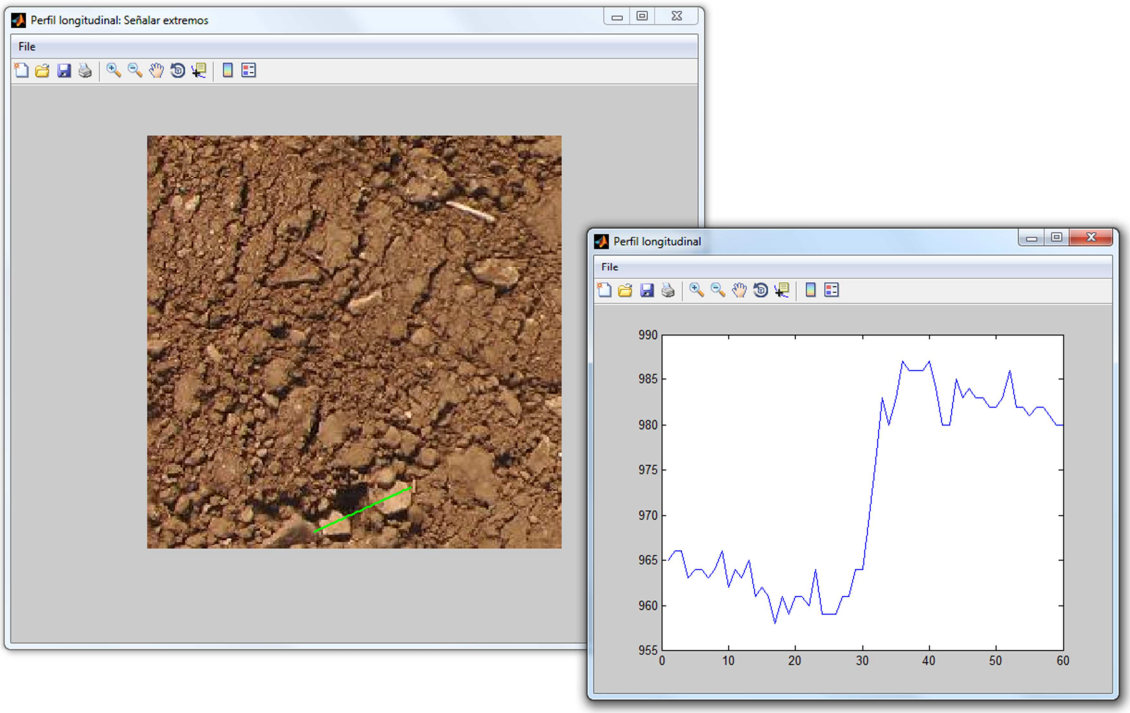


Figura 6-6. Perfil longitudinal zona lisa

6.1.2. Zona media

Para la zona media se muestra en las siguientes figuras un MDE con tamaño 100x100 dado un paso de malla de 5 mm en planimetría y incrementos sobre la vertical de 1 mm:

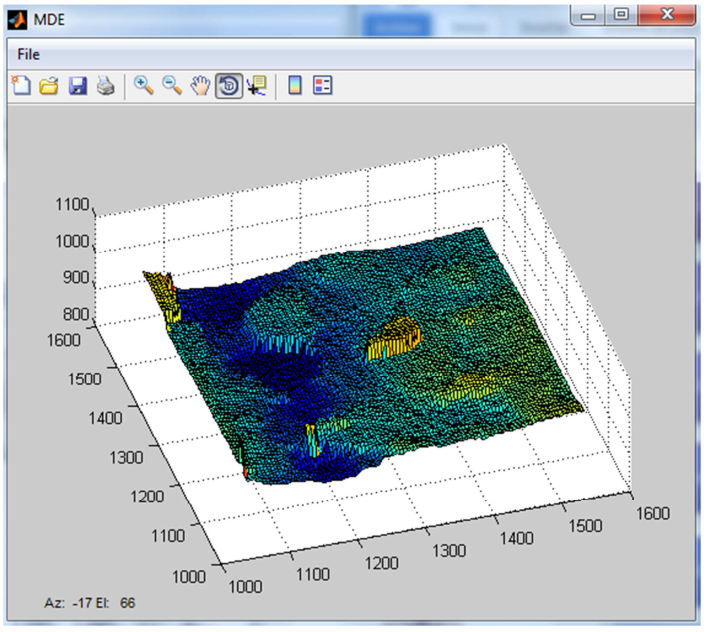


Figura 6-7. Modelo zona media 1

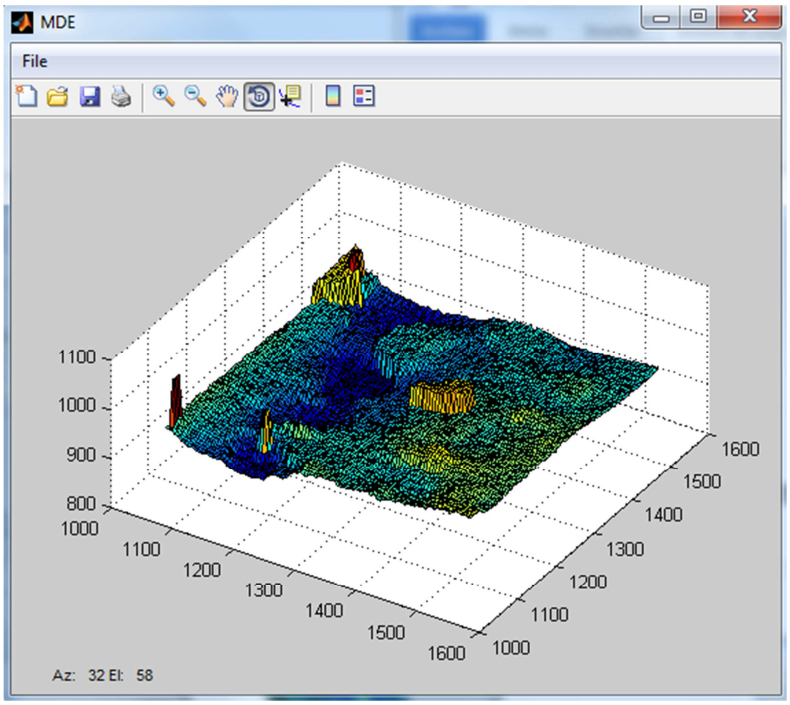


Figura 6-8. Modelo zona media 2

Otros ejemplos son los que observamos al renderizar y aplicar una ortofoto, para el caso anterior del MDE de tamaño 100x100:

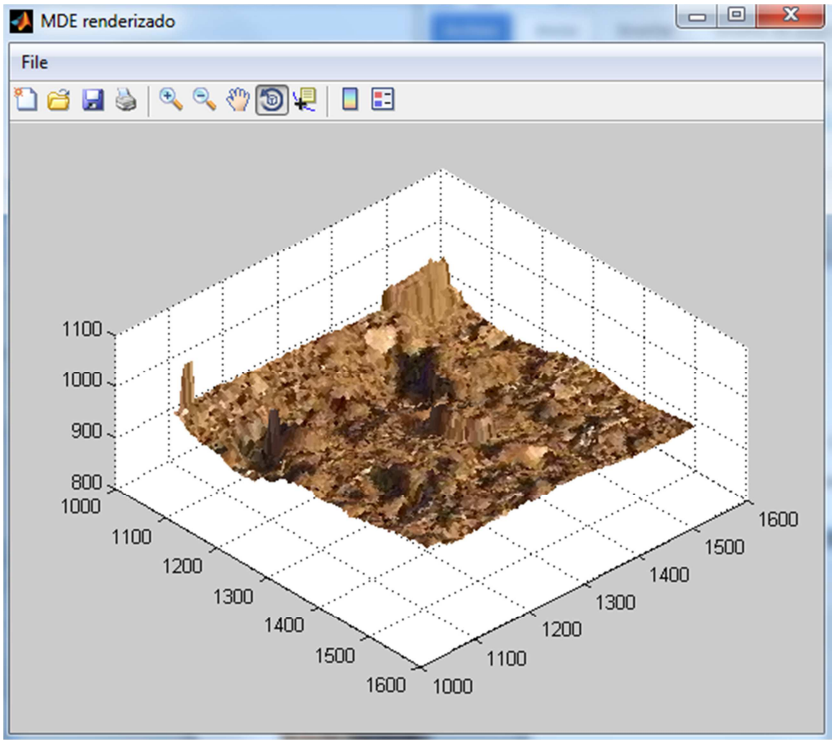


Figura 6-9. Modelo zona media renderizdo 1

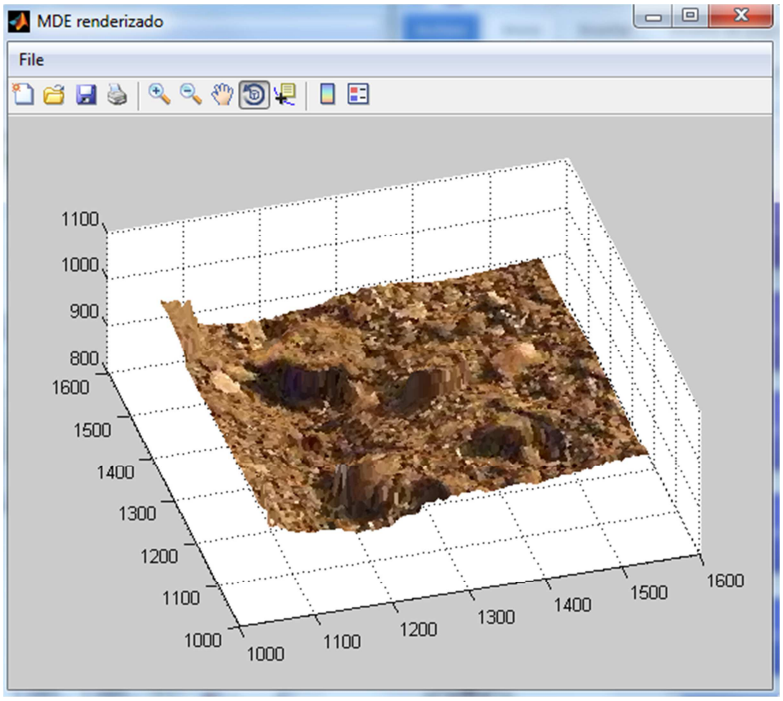


Figura 6-10. Modelo zona media renderizado 2

La cota mínima de este modelo es de 890 mm y la máxima de 1035 mm, sabiendo que la cota local del marco es de 1000 mm.

Las ortofotos obtenidas con cada fotografía y la ortofoto promedio han sido:

Ortofoto 1

Ortofoto 2

Ortofoto 3

Ortofoto 4

Ortofoto promedio

Figura 6-11. Orfotos zona media

Un ejemplo de perfil longitudinal para esta zona es:

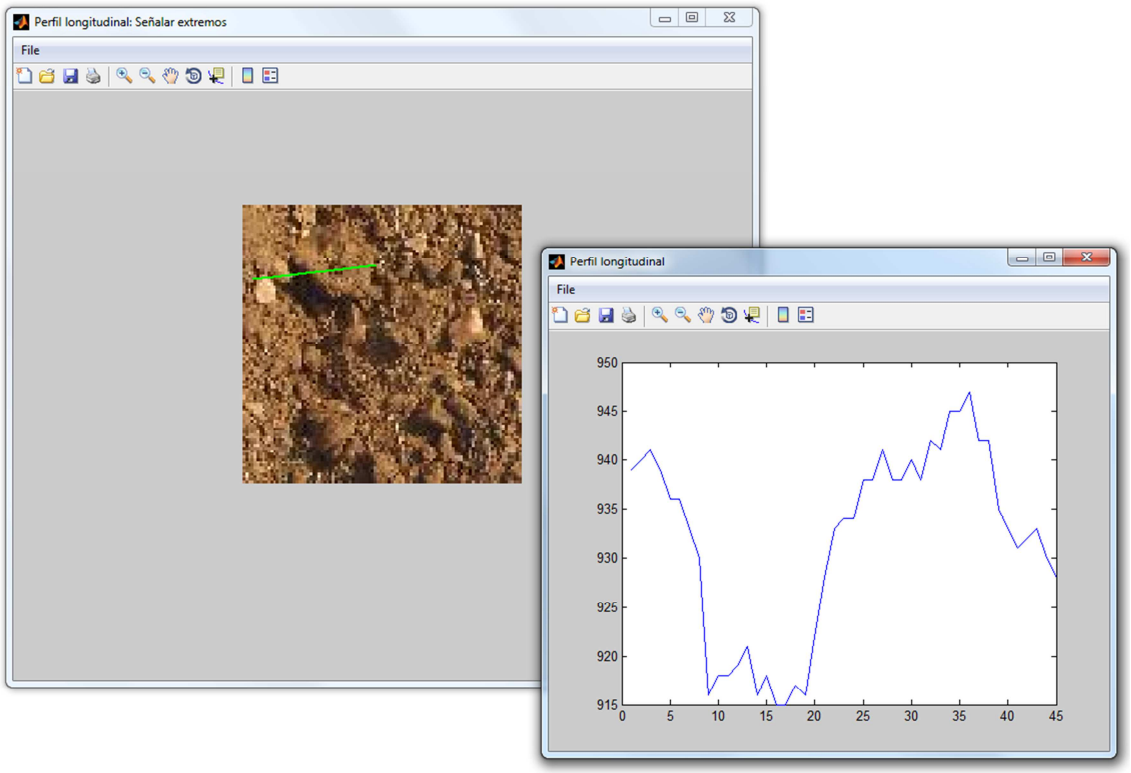


Figura 6-12. Perfil longitudonal zona media

6.1.3. Zona rugosa

Se muestra en las siguientes figuras un MDE con tamaño 100x100 dado un paso de malla de 5mm en planimetría y incrementos sobre la vertical de 1 mm:

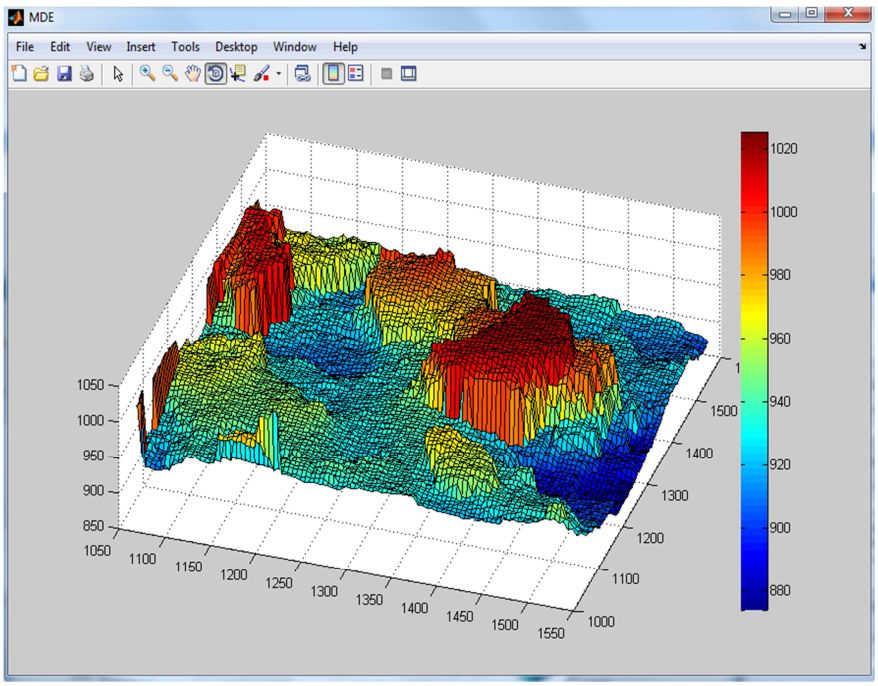


Figura 6-13. Modelo zona rugosa 1

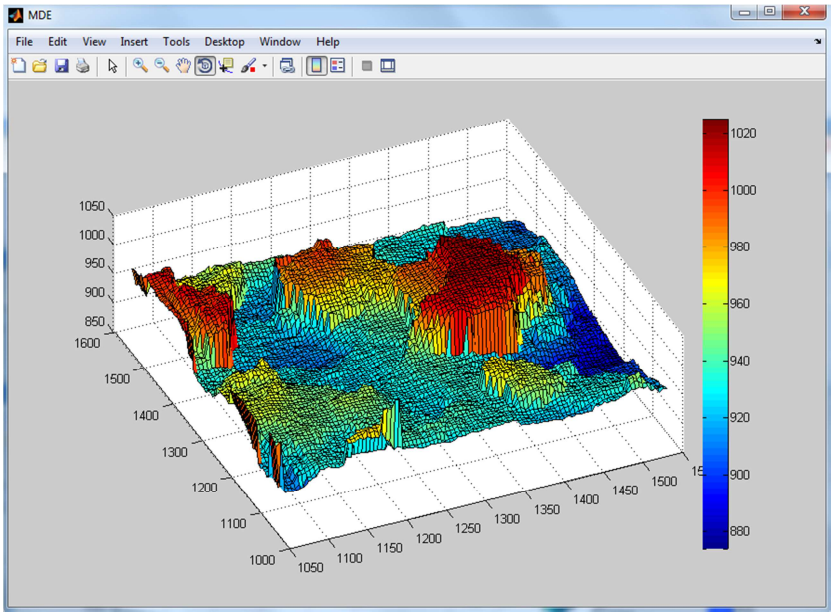


Figura 6-14. Modelo zona rugosa 2

Cota mínima 874 mm y cota máxima 1025 mm con cota del marco 1000 mm.

Otros ejemplos son los que observamos al renderizar y aplicar una ortofoto, para el caso anterior del MDE de tamaño 100x100:

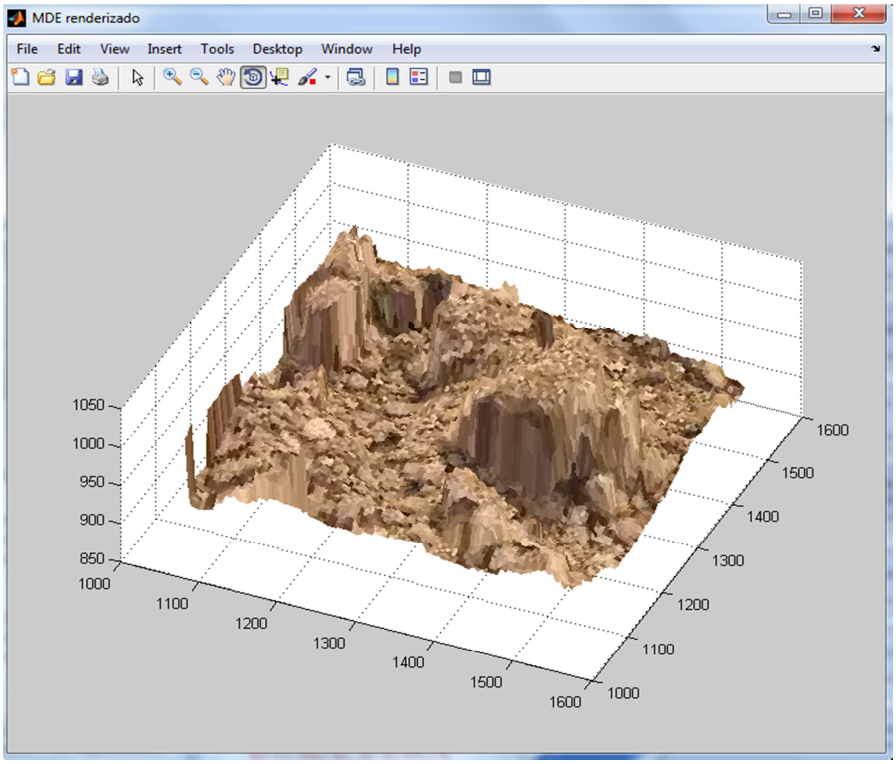


Figura 6-15. Modelo zona rugosa renderizado

Con un perfil longitudinal:

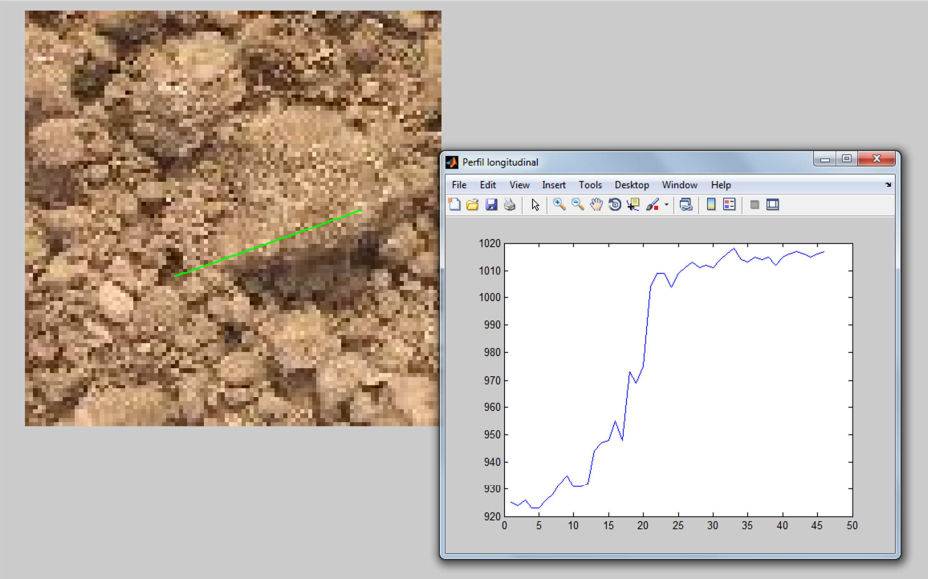


Figura 6-16. Perfil longitudinal zona rugosa

Para otro MDE del mismo lugar con tamaño 40x40 tenemos:

Obsérvese en este último caso como se ve a simple vista cada celda del MDE.

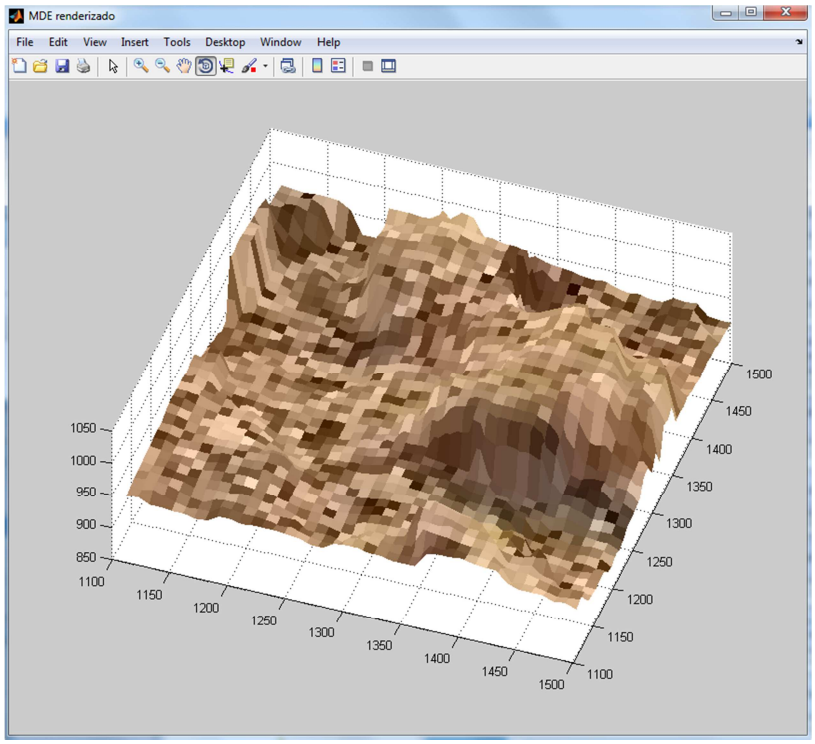
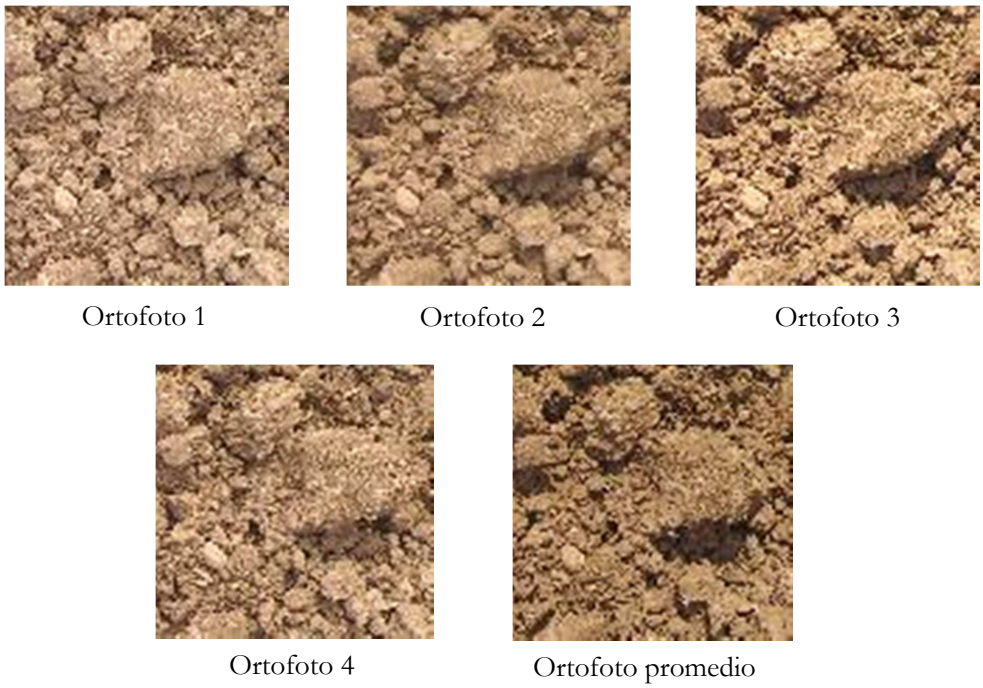


Figura 6-17. Modelo zona rugosa renderizado menor resolución

Las ortofotos obtenidas con cada fotografía y la ortofoto promedio han sido:



Ortofoto 1 Ortofoto 2 Ortofoto 3

Ortofoto 4 Ortofoto promedio

Figura 6-18. Orfotos zona rugosa

6.2. Comparativa con los resultados ofrecidos por Orthoengine PCI Geomatica

Para comparar resultados con otro software se ha realizado lo mismo con Orthoengine de PCI Geomatica. Los pasos a dar en este programa son los siguientes:

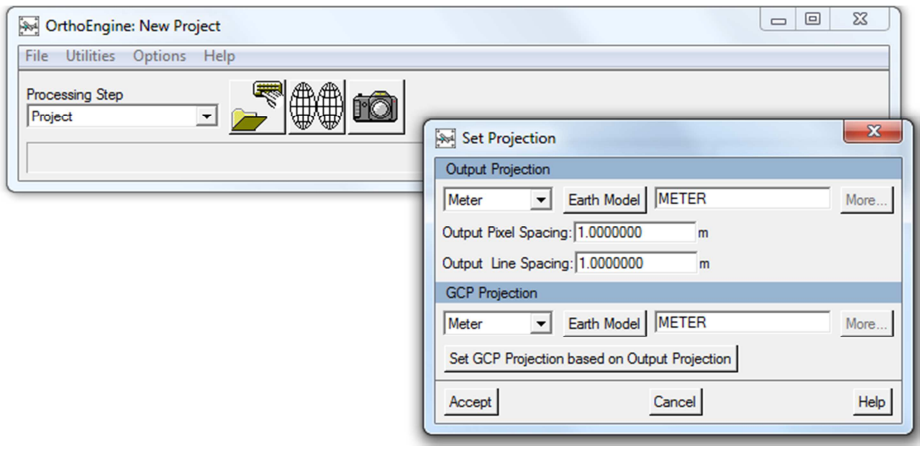


Figura 6-19. Ventana inicio Orthoengine

Las unidades se dejan en metros considerando en nuestro caso que son milímetros y tomamos que cada pixel se de al milímetro.

La información de la cámara que introducimos será la misma que en Matlab despreciando correcciones por distorsión y del punto principal.

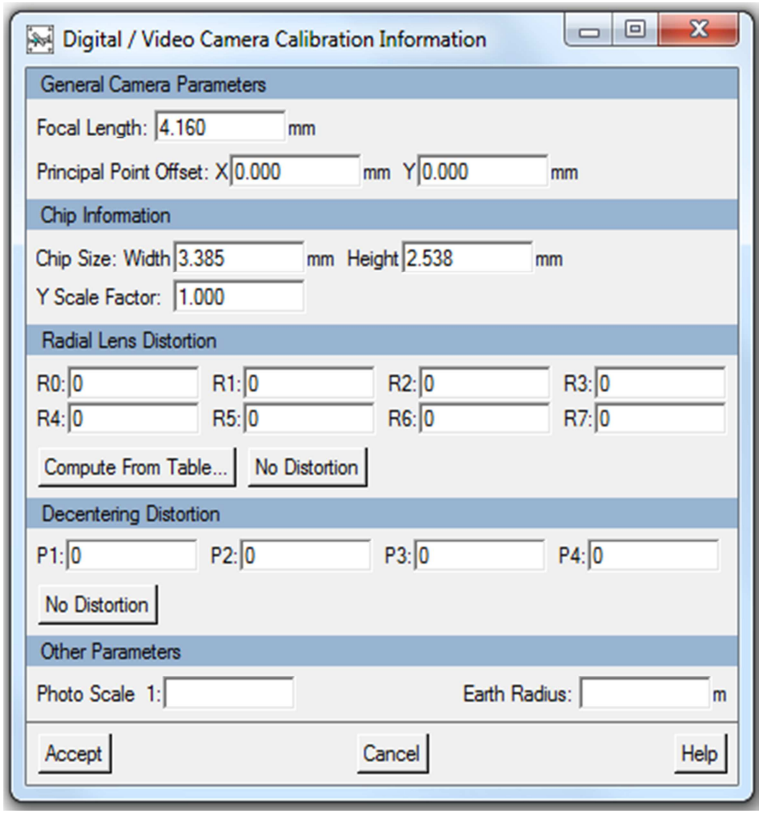


Figura 6-20. Ventana cámara Orthoengine

Realizamos la orientación externa marcando puntos en la imagen e introduciendo las coordenadas terreno de los mismos. Esta tarea se lleva a cabo para cada foto.

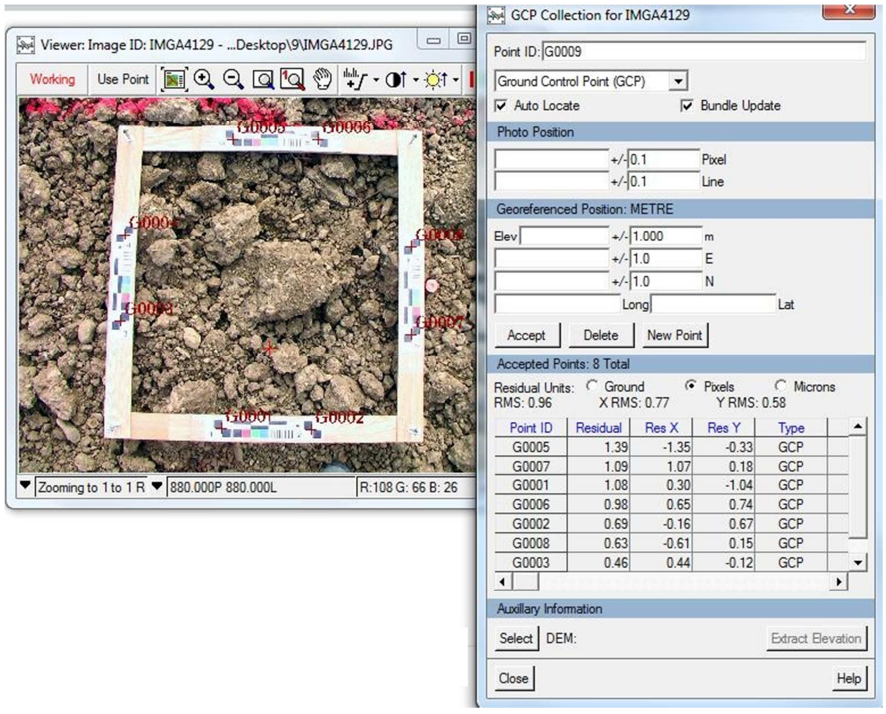


Figura 6-21. Ventana captura de puntos

Creamos parejas de imágenes epipolares como vemos:

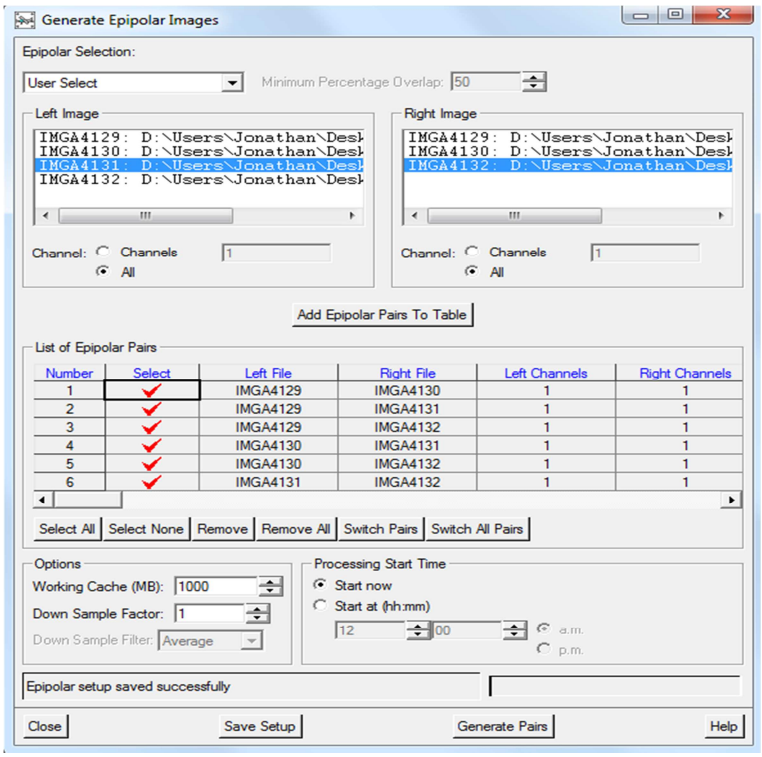


Figura 6-22. Ventana creación de imágenes epipolares Orthoengine

Finalmente ejecutamos la creación para la resolución final de 1 y de 5 pixels.

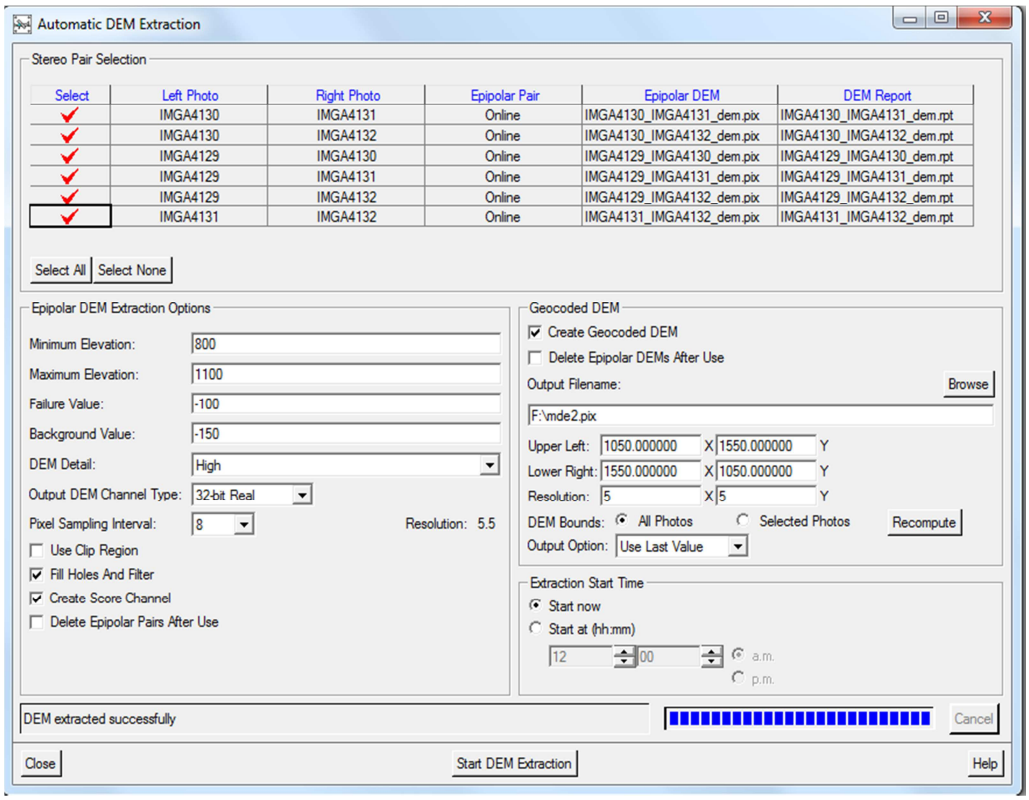


Figura 6-23. Ventana creación MDE

Se observan las diferencias en detalle de ambos. Pero ahora para comparar estos debemos exportar el de resolución 5 a un tipo de imagen .tiff y abrirlo con Matlab.

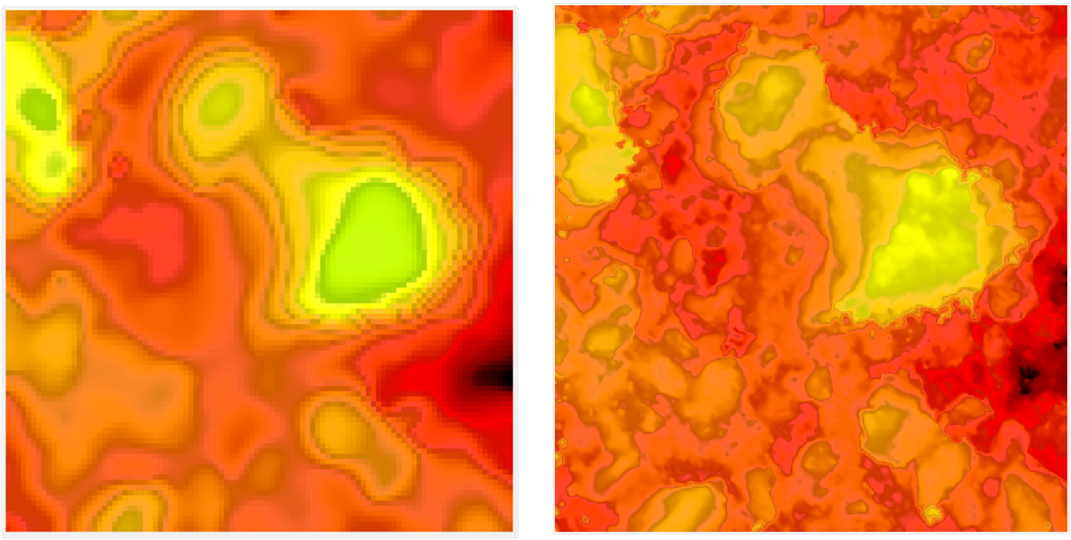


Figura 6-24. MDE resultado con Orthoengine de resolución 5 píxel en la izquierda y de 1 píxel a la derecha

Podemos ver el grado de suavizado que se obtiene en esta representación obtenida lo que varía mucho de nuestro modelo.

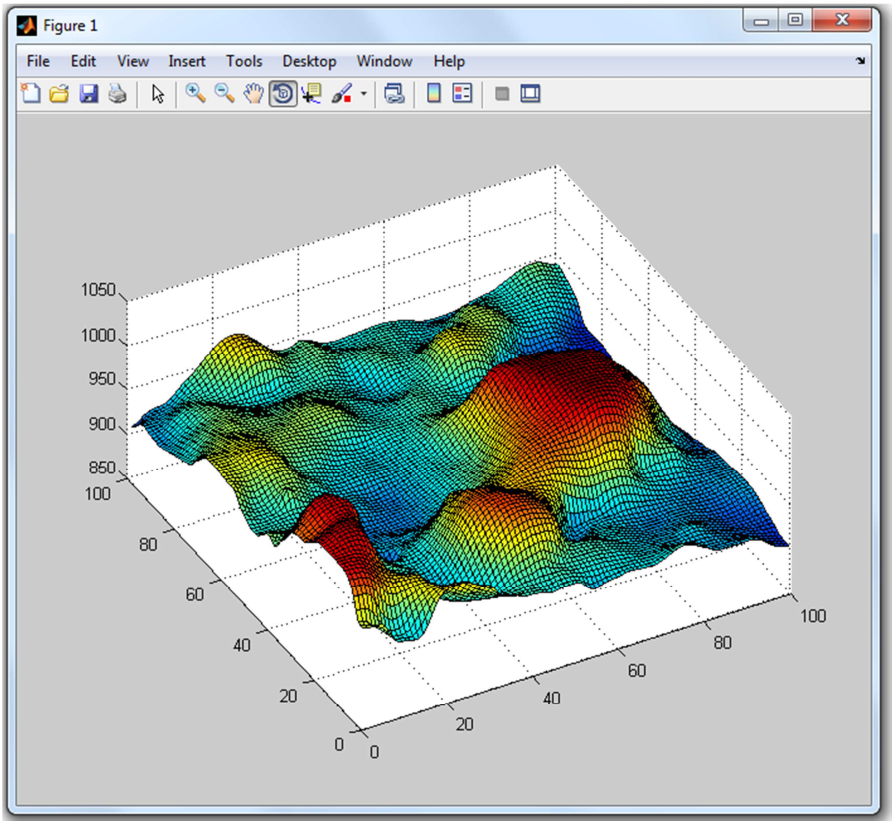


Figura 6-25. MDE resultado con Orthoengine sobre Matlab

Al comparar el modelo de PCI con el programado en Matlab vamos a utilizar la resta de matrices, posteriormente obtendremos el valor máximo positivo y negativo de estas diferencias y la media de diferencias obteniéndose los siguientes resultados:

- Diferencia máxima positiva = 88.2163 mm
- Diferencia máxima negativa = -79.0419 mm
- Media de las diferencias = -3.4423 mm

Como de esta forma en la media intervienen valores positivos y negativos procedemos a obtener una matriz de diferencias positiva prescindiendo del signo:

- Diferencia máxima = 88.2163 mm
- Diferencia mínima = 0.0012 mm
- Media de diferencia prescindiendo de signo = 8.0947 mm

Con PCI se han obtenido valores decimales que podemos despreciar.

En las diferencias máximas llegamos a los 88 milímetros que pueden ser debidos al suavizado pero como media tenemos un error de 8 milímetros que puede ser un valor significativo si tenemos en cuenta que el programa opera de forma distinta pero que devuelve un MDE de aspecto similar.

Visualizando la matriz de diferencias comprobamos mayores diferencias en las zonas de mayor suavizado:

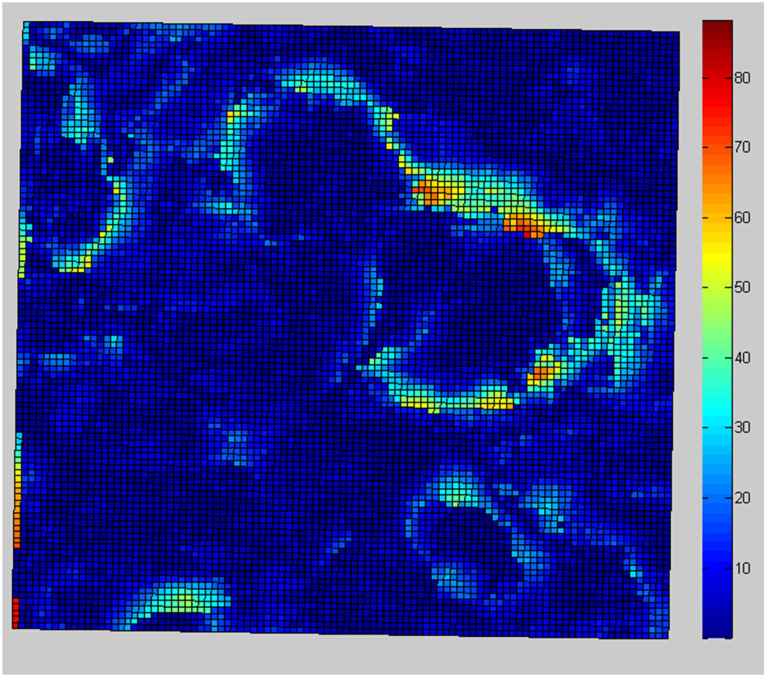


Figura 6-26. Matriz de diferencias

Otro ejemplo que incluimos para la comparativa es el de la zona media:

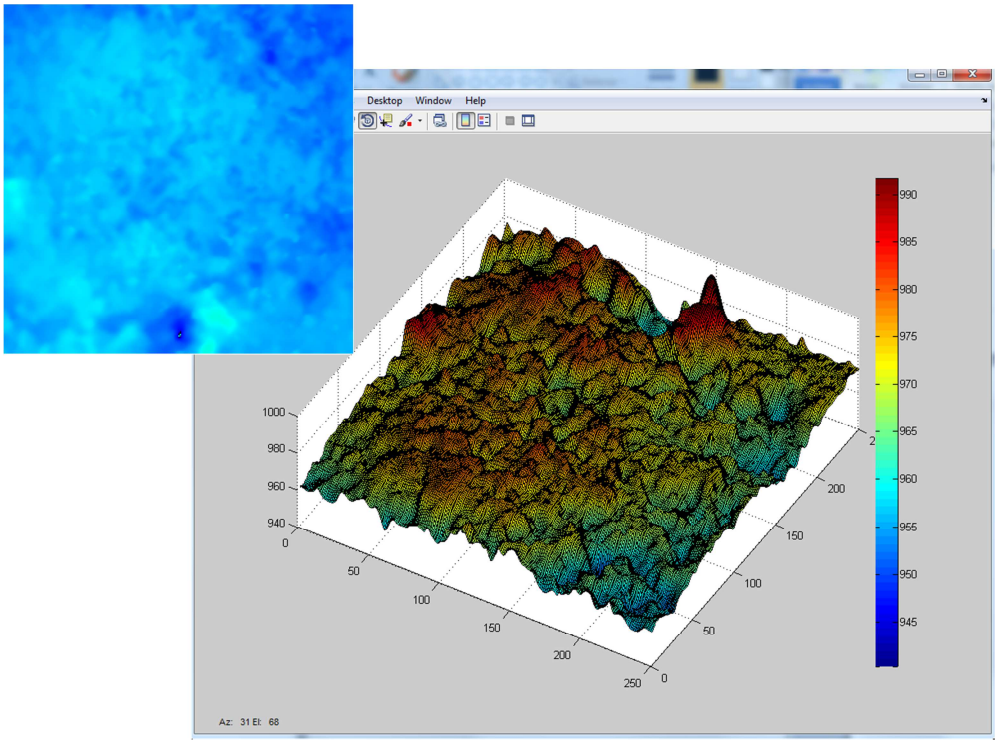


Figura 6-27. MDE resultado con Focus Geomatica (izquierda parte superior) y Orthoengine sobre Matlab (derecha parte inferior)

Procediendo de la misma forma al anterior tenemos:

- Diferencia máxima = 30.7862 mm
- Diferencia mínima = 0.0001 mm
- Media de diferencia prescindiendo de signo = 2.9091 mm

Aquí tenemos un error medio de 2.9091 mm mucho inferior al otro caso ya que tenemos una zona lisa con un suavizado que afecta menos.

6.3. Prueba con "CHISME" de objetos regulares

Para comprobar la veracidad de los reultados obtenidos, de forma que no solo nos basamos en lo que obtenemos con otro software, el cual, también puede tener sus fallos, se ha decidido emplear el dispositivo CHISME para medir por medio de fotogrametría, dimensiones de objetos regulares de los cuales conocemos sus dimensiones por haberse medido con flexómetro y cinta métrica.

Las fotos tomadas son las cuatro siguientes:

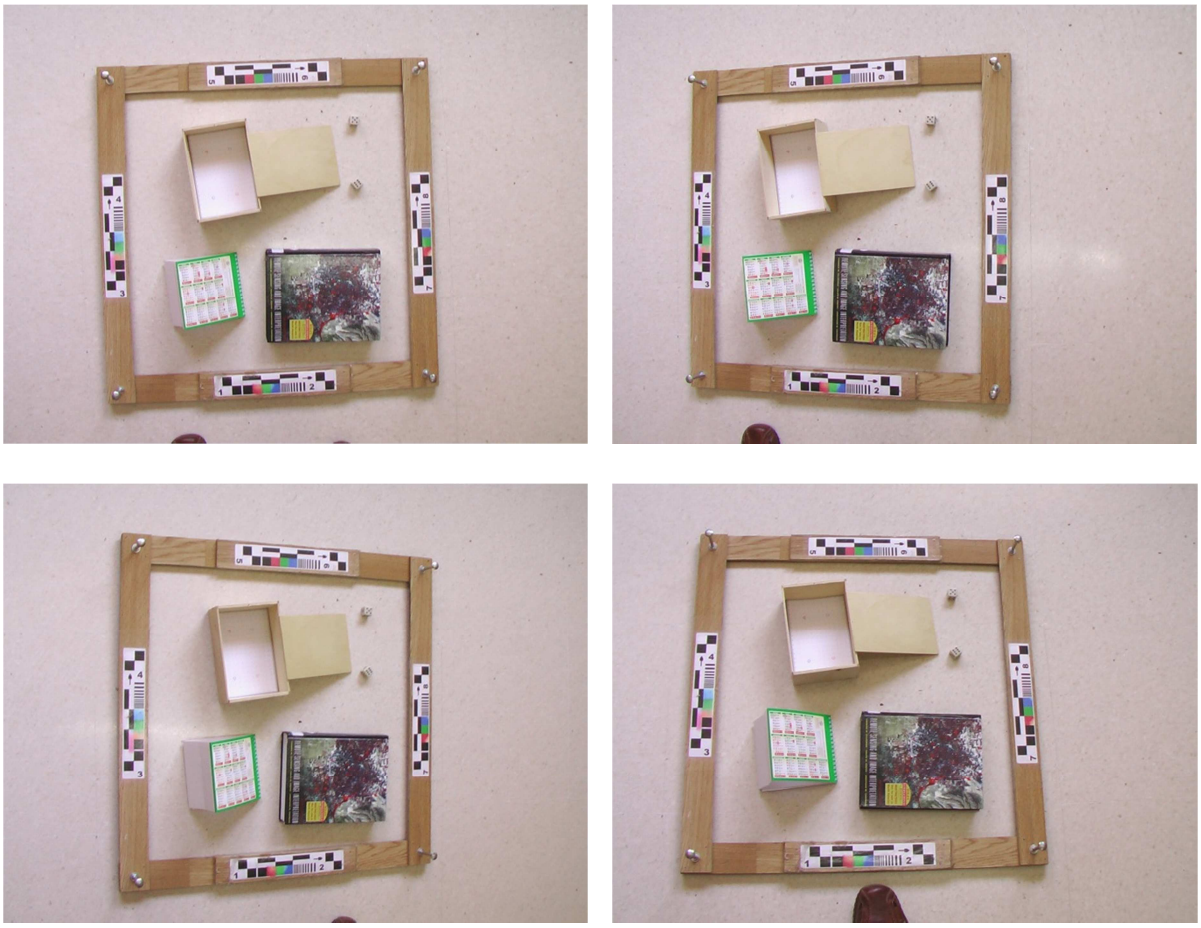


Figura 6-28. Imágenes de objetos regulares con "CHISME"

En las fotos se observan una caja de madera, dos dados, un libro y un calendario de mesa de la universidad.

Primero vamos a observar el MDE que se obtiene al renderizar el mismo, así como sus ortofotos:

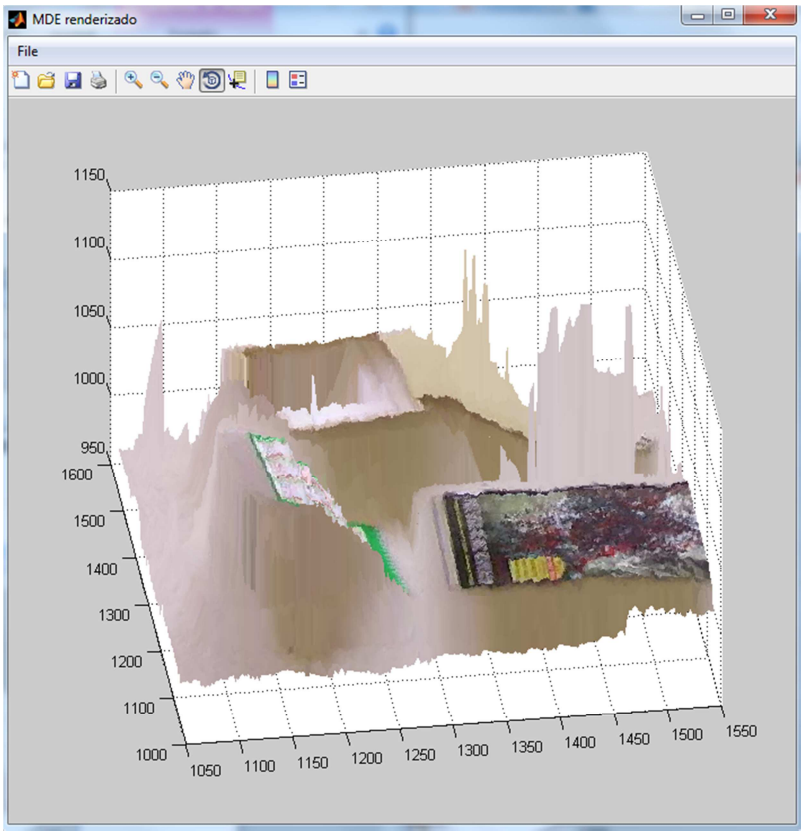


Figura 6-29. MDE objeto regulares 1

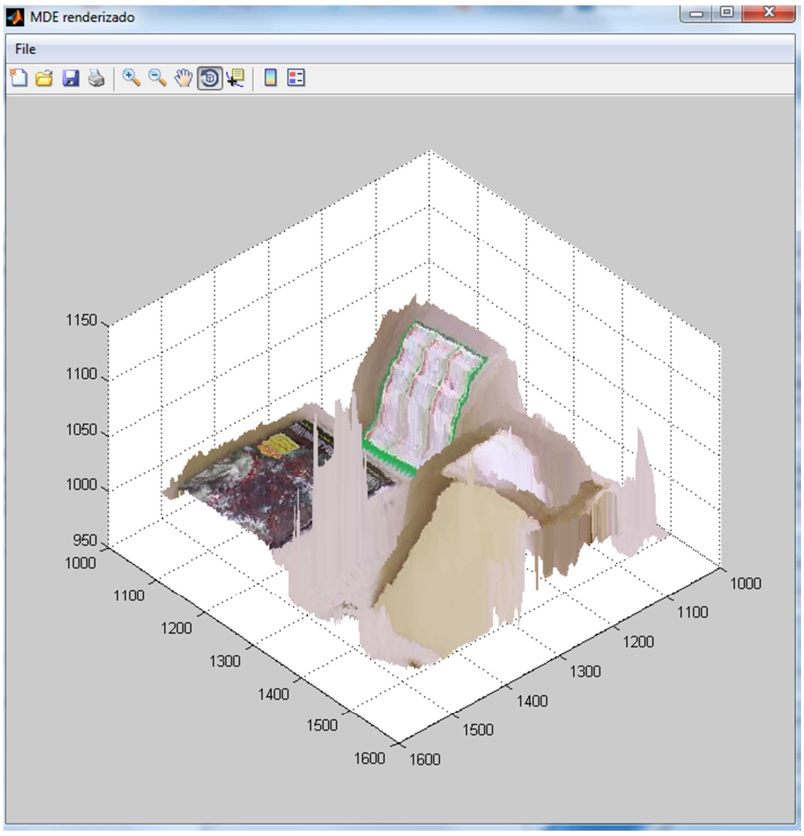


Figura 6-30. MDE objetos regulares 2

Ortofoto 1 Ortofoto 2 Ortofoto 3

Ortofoto 4 Ortofoto promedio

Figura 6-31. Orfotos de objetos regulares

Las dimensiones de los objetos son las siguientes:

- Caja de madera: 13,5x19,5x7cm
- Libro: 19,5x24,2x3,5 cm
- Dados: 2x2x2 cm
- Calendario: 14x15,5 cm en planta con cota máxima de 10 cm

Para proceder a las medidas con el MDE obtenido haremos uso de la función del perfil longitudinal.

El MDE de salida es de tamaño 250x250 con paso de malla de 2 mm en planimetría y 1 mm en altimetría.

Como tenemos 2 mm en planimetría, tendremos que multiplicar por este valor la medida en planta.

Empecemos con el caso del calendario y veamos cual es la medida aproximada que obtenemos por este procedimiento:

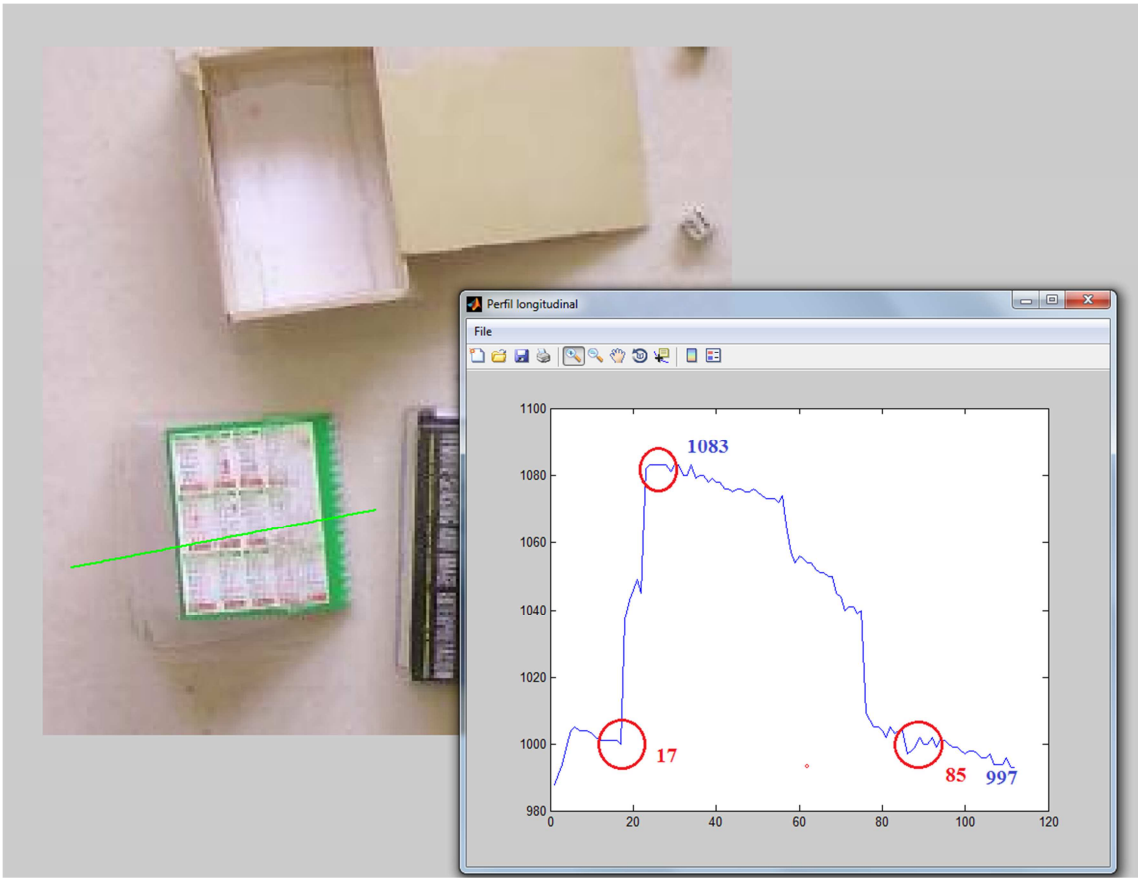


Figura 6-32. Perfil longitudinal del calendario

Con el perfil longitudinal observamos diferencia en longitud en planimetría (rojo) de 85-17=68 mm que x2 es 136 mm o lo que es lo mismo 13,6 cm. En diferencia de cota (azul) tenemos 1083-997=86 mm, 8,6 cm.

Discrepancias de:

- Planimetría 14-13,6 cm =0,4 cm = 4 mm
- Altimetría 10-8,6 cm= 1,4 cm

En el caso del libro:

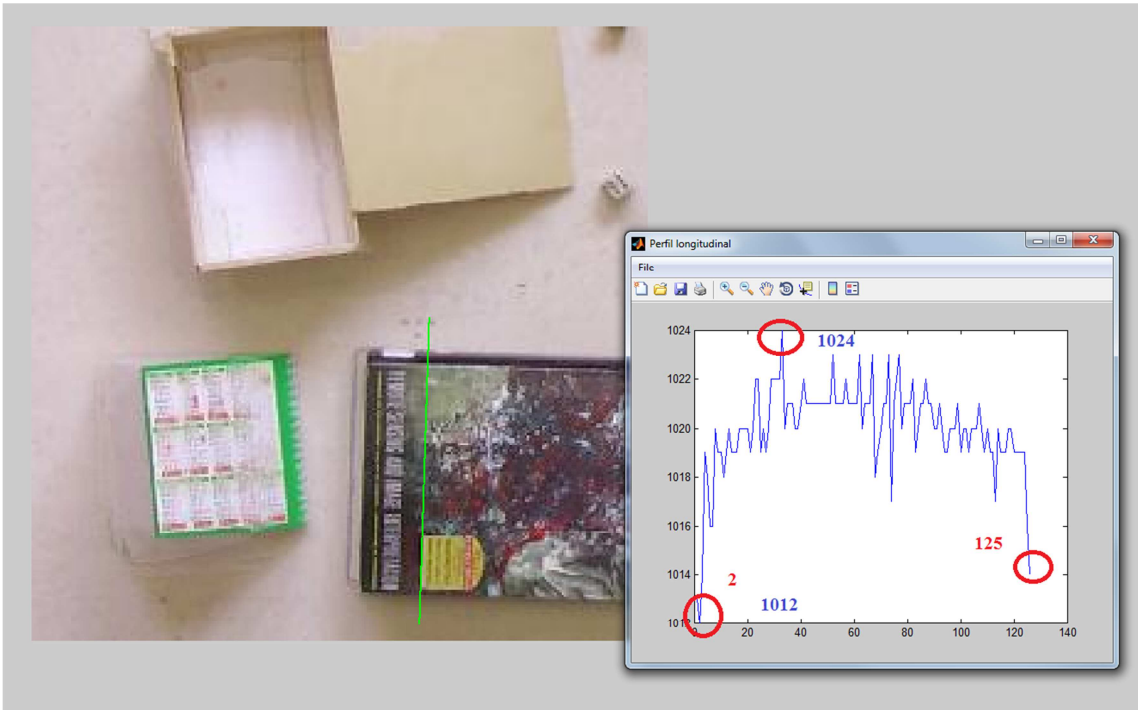


Figura 6-33. Perfil longitudinal del libro

Con el perfil longitudinal observamos diferencia en longitud en planimetría (rojo) de 125-2=123 mm que x2 es 246 mm o lo que es lo mismo 24,6 cm. En diferencia de cota (azul) tenemos 1024-1012=12 mm, 1,2 cm.

Discrepancias de:

- Planimetría 19,5-24,6 cm =-5,1 cm
- Altimetría 3,5-1,2 cm= 2,3 cm

En la caja tenemos:

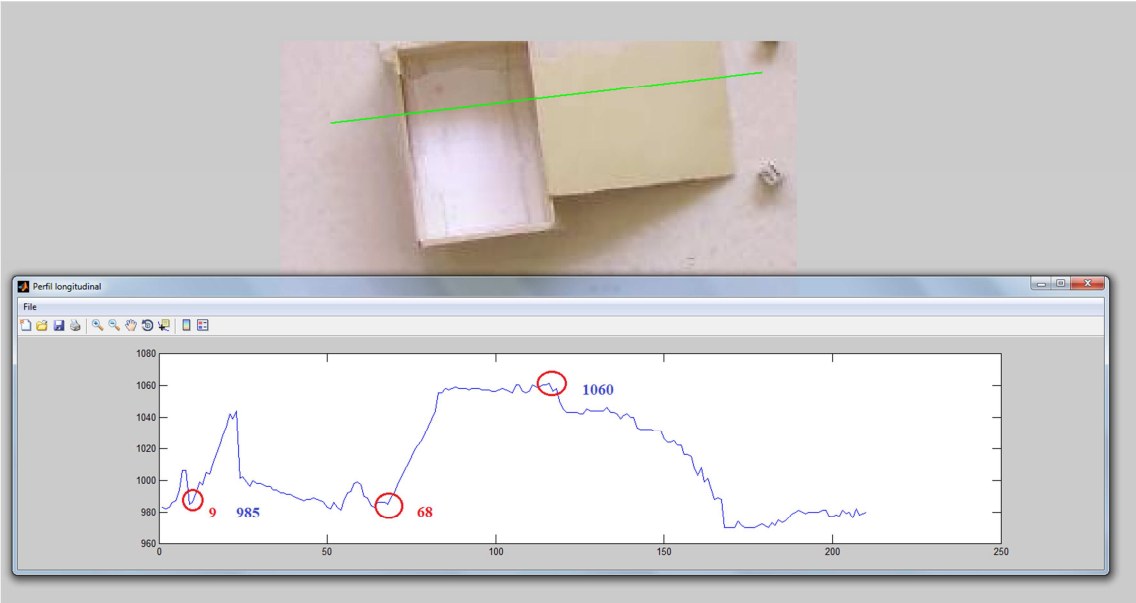


Figura 6-34. Perfil longitudinal de la caja

Con el perfil longitudinal observamos diferencia en longitud en planimetría (rojo) de 68-9=59 mm que x2 es 118 mm o lo que es lo mismo 11,8 cm. En diferencia de cota (azul) tenemos 1060-995=65 mm, 6,5 cm.

Discrepancias de:

- Planimetría 13,5-11,8 cm = 1,7 cm
- Altimetría 7-6,5 cm= 2,3 cm

Para los dados:

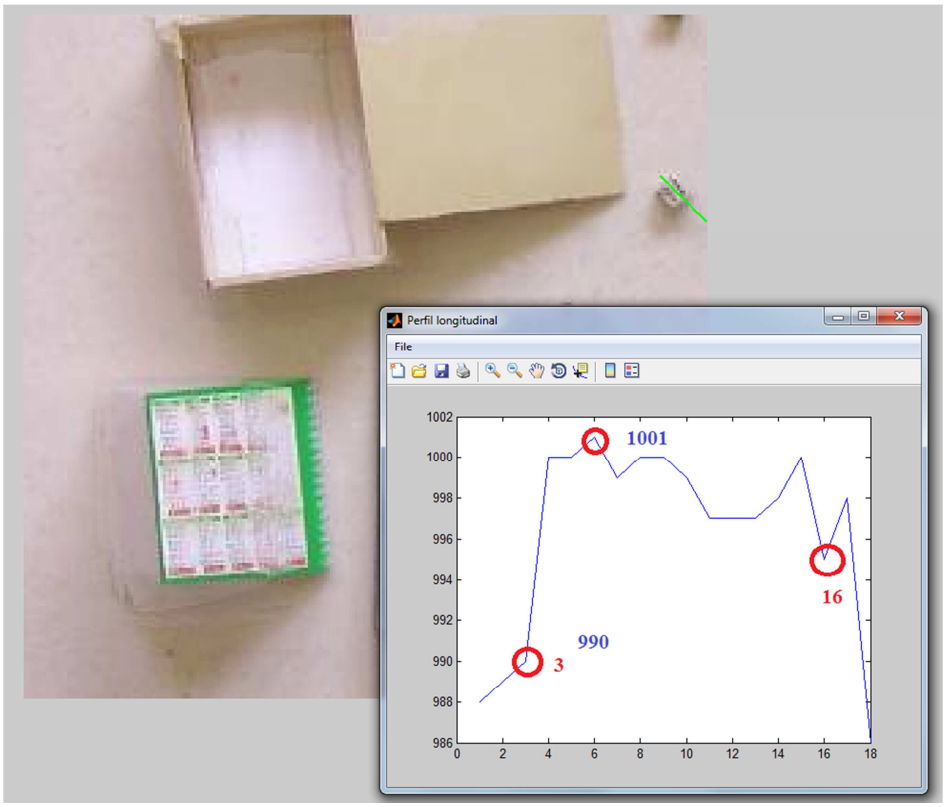


Figura 6-35. Perfil longitudinal del dado

Con el perfil longitudinal observamos diferencia en longitud en planimetría (rojo) de 16-3=13 mm que x2 es 26 mm o lo que es lo mismo 2,6 cm. En diferencia de cota (azul) tenemos 1001-990=11 mm, 1,1 cm.

Discrepancias de:

- Planimetría 2-2,6 cm = -0.6 cm
- Altimetría 2-1,1 cm= 0,9 cm

La primera impresión que obtenemos no es buena ya que las discrepancias son bastante altas. No obstante hay que tener en cuenta que el suelo donde están apoyados los objetos tiene una cota inferior a la del marco ya que el marco tiene un espesor de casi 2 cm.

A parte de esto, tenemos en cuenta que el suelo es una superficie que en las imágenes se ve con una textura muy homogénea lo que conlleva problemas de correspondencia basada en áreas.

Por tanto, si el suelo no es capaz de modelarlo correctamente no se están añadiendo los 2 cm por debajo de la cota 1000 del marco.

Aquí, es donde empezamos a encontrar algunas de las limitaciones de este software puesto que intentábamos realizar la prueba de que obtenemos resultados fiables pero no se daban las mejores condiciones para ello.

Vamos a ver cual es el resultado que obtendríamos con Orthoengine de PCI Geomatica:

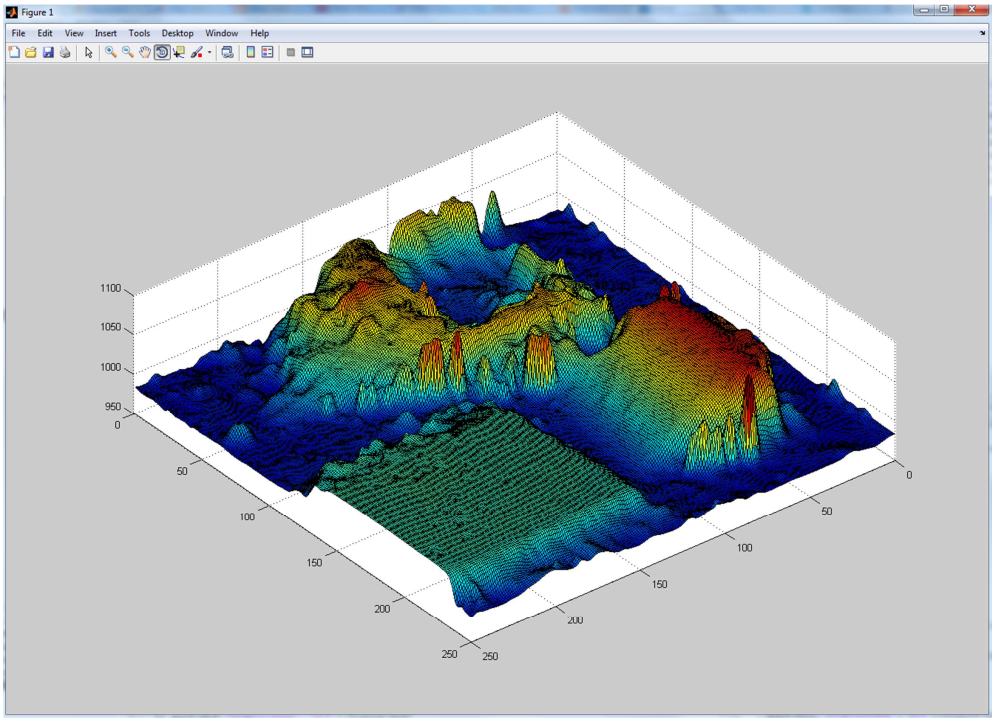


Figura 6-36. MDE objetos regulares obtenido con Orthoengine

La matriz de diferencias es la siguiente, entre el obtenido con nuestro software y el de PCI Geomatica:

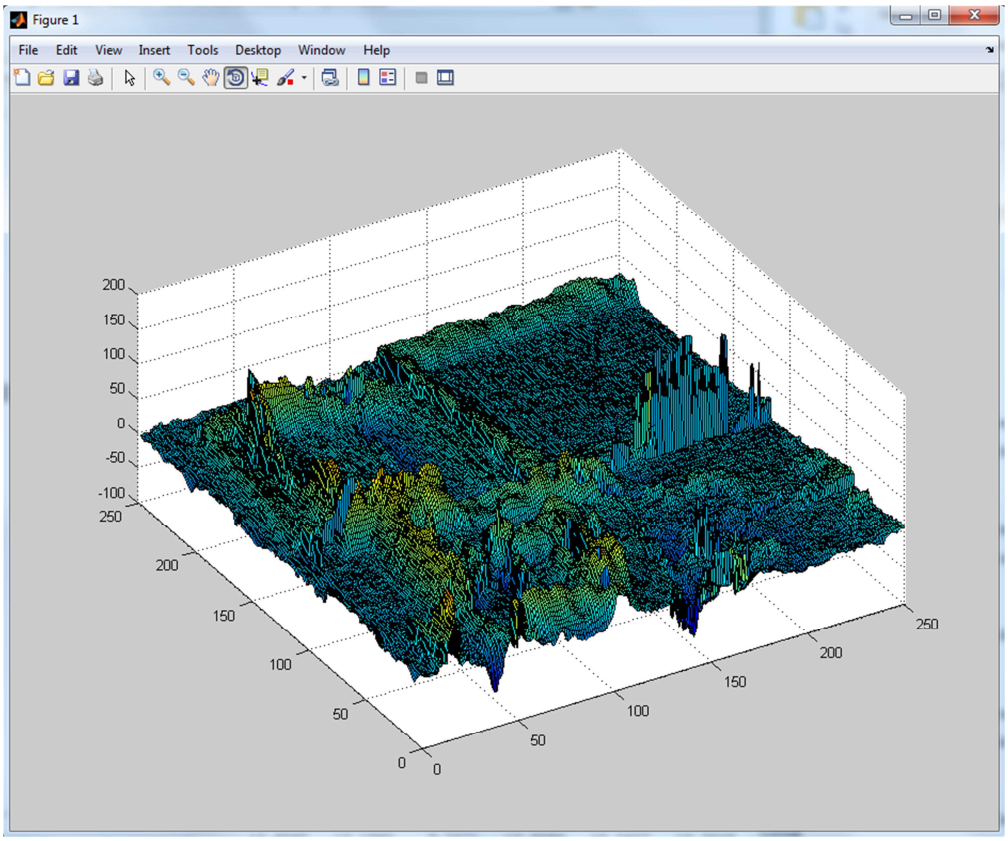


Figura 6-37. Matriz de diferencias

Al igual que sucedía con las tomas del terreno se observa gran suavizado en los modelos obtenidos.

Tampoco se han obtenido muy buenos resultados y las diferencias son las siguientes:

- Diferencia máxima = 156,2853 mm
- Diferencia mínima 0,0003 mm
- Media de diferencias = 11,7192 mm

Concluimos esta comparativa diciendo que las mayores diferencias se encuentran en el suavizado pero por lo demás los resultados toman un aspecto semejante.

CAPÍTULO 7. CONCLUSIONES

La primera impresión que se obtiene del proyecto es buena, si bien nos encontramos con algunas limitaciones.

Unas de las limitaciones principales que encontramos con este software al igual que sucede con otros comerciales es el que nos encontramos al elaborar un MDE de una zona que presenta homogeneidad de textura con lo que es difícil encontrar puntos homólogos y por tanto no seremos capaces de obtener un modelo realista.

Para solucionar problemas de este tipo se aconseja emplear correspondencias que no solo estén basadas en áreas sino también en características o bordes y esquinas.

Otra limitación que encontramos en el software es el tiempo que necesita para elaborar un MDE de tamaño 100x100, pues este está entorno a las 2 horas.

Matlab es un lenguaje de programación y un entorno que trabaja de forma sencilla con matrices e imágenes, pero no es un lenguaje tan rápido como lo sería el lenguaje estructurado C. Sería conveniente analizar otras alternativas de lenguaje y compilación del software que permitan una mayor velocidad y eficiencia.

En la comparativa realizada con el software Orthoengine de PCI Geomatica encontramos que hay diferencias en el funcionamiento de ambos y que este conlleva un gran suavizado de los resultados que produce discrepancias con los obtenidos por el software elaborado.

Finalmente, se piensa que el software cumple con los objetivos que se buscaban y puede servir de ayuda a la hora de analizar la rugosidad del terreno.

ANEXO: Calibración de la cámara con PhotoModeler Scanner

La calibración de la cámara se realiza para unos determinados ajustes de distancia focal, resolución y otros ajustes mediante una operación sencilla y automática dentro del programa. Desde el menú del programa se pude imprimir una rejilla de puntos, que se sitúa en una superficie horizontal.

La forma más habitual de hacer una calibración en Fotogrametría es utilizar ajuste de haces, tomando los parámetros intrínsecos de la cámara como desconocido y todas las partes de la tierra como conocidas. Photomodeler tiene una simple manera de hacer eso usando la rejilla de calibración de cámaras que es fotografiada desde 8 puntos de vista bajo una incidencia de más o menos 45º.

Véase a continuación (figura 8-1) como es la rejilla que utiliza Photomodeler a la que harán varias fotos:

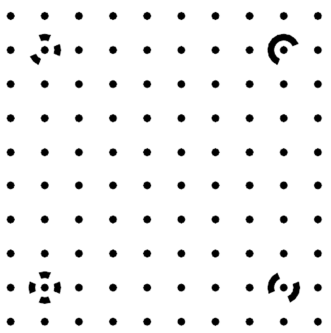


Figura 8-1. Imagen de la rejilla utilizada por Photomodeler Scanner

Para un mejor seguimiento de como se realiza la calibración con dicho programa, se incluye a continuación una serie de imágenes del programa descriptivas.

Al empezar el programa nos muestra algunas funciones, entre ellas se encuentra la que se denominada "Camera Calibration Project" como podemos observar

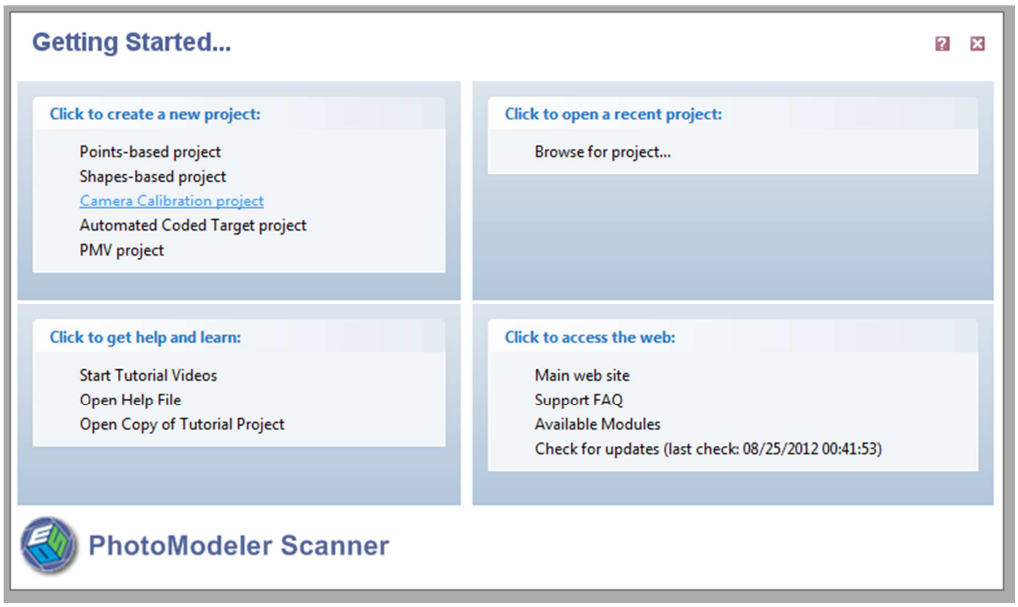


Figura 8-2. Ventana inicio

Al seleccionar esta opción pasamos a la siguiente ventana:

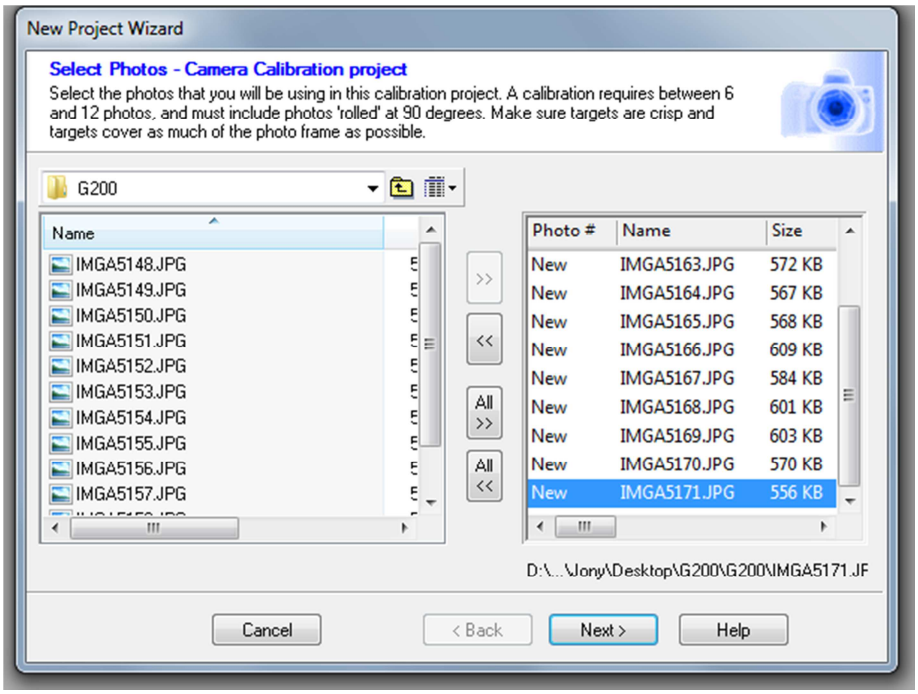


Figura 8-3. Selección de imágenes

Esta ventana nos pide que seleccionemos entre 6 y 12 fotos correspondientes a las de la rejilla de calibración.

Procediendo con el botón "Next" llegamos a otra ventana:

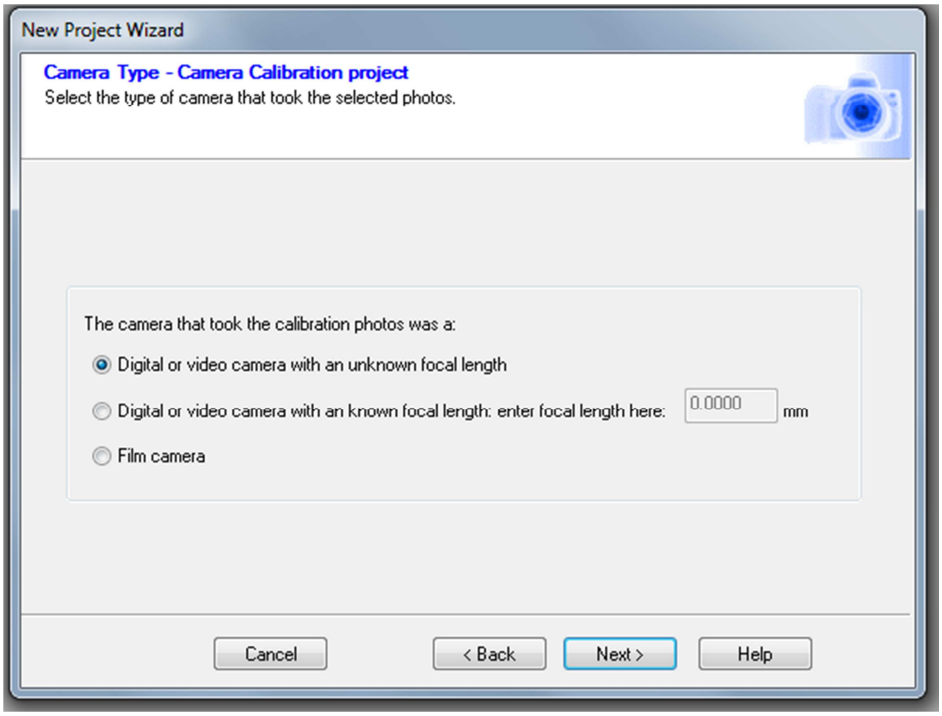


Figura 8-4. Tipo de cámara y focal conocida o desconocida

En ella se nos pregunta por el tipo de cámara y si conocemos el valor de la focal, aunque este sea aproximado, que se introduzca.

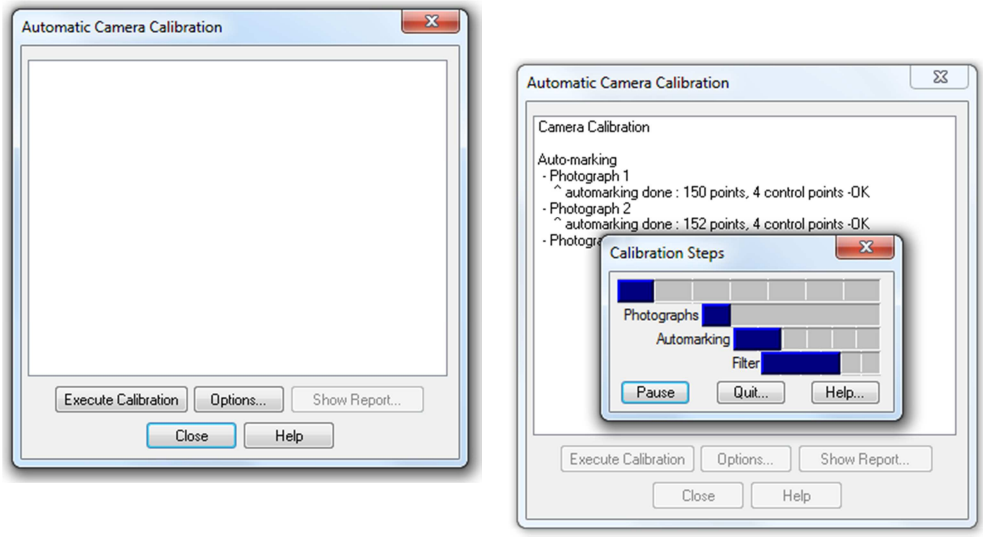


Figura 8-5. Ventana de ejecución

De nuevo con el botón "Next" llegamos a otra ventana que nos dice si queremos ejecutar la calibración, (execute calibration) y tras seleccionar el botón pasamos de la ventana superior izquierda a la de la derecha indicándonos los pasos que el programa va haciendo.

El programa tarda un tiempo en hacer los cálculos y finalizar con la tarea de calibración, finalmente nos muestra los resultados y nos permite imprimirlos a un fichero.

En el caso nuestro el informe es el que se recoge a continuación:

```
Precisions / Standard Deviations
      Camera Calibration Standard Deviations
         Camera1: Unnamed Default Camera
            Focal Length
               Value: 4.161390 mm
               Deviation: Focal: 0.001 mm
            Xp - principal point x
               Value: 1.734840 mm
               Deviation: Xp: 3.3e-004 mm
            Yp - principal point y
               Value: 1.210864 mm
               Deviation: Yp: 9.5e-004 mm
            Fw - format width
               Value: 3.384756 mm
               Deviation: Fw: 6.0e-005 mm
            Fh - format height
               Value: 2.538000 mm
            K1 - radial distortion 1
               Value: 1.505e-002
               Deviation: K1: 6.6e-005
            K2 - radial distortion 2
               Value: -5.948e-004
               Deviation: K2: 2.0e-005
            K3 - radial distortion 3
               Value: 0.000e+000
            P1 - decentering distortion 1
               Value: 4.646e-004
               Deviation: P1: 4.5e-006
            P2 - decentering distortion 2
               Value: -2.110e-004
               Deviation: P2: 1.2e-005
```

Figura 8-6. Informe de resultados

De aquí el valor que recogeremos será el de la focal de 4,16 mm redondeado con dos decimales y no introduciremos correcciones de distorsiones ni del punto principal.

ANEXO: Lenguaje de Programación: Matlab

El lenguaje de programación utilizado es Matlab. MATLAB (abreviatura de MATrix LABoratory, "laboratorio de matrices") es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows y Mac OS X.

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (toolboxes); y las de Simulink con los paquetes de bloques (blocksets).

Es un software muy usado en universidades y centros de investigación y desarrollo. En los últimos años ha aumentado el número de prestaciones, como la de programar directamente procesadores digitales de señal o crear código VHDL.

En este anexo se recogen algunas de las funciones más usadas en el proyecto con una breve explicación de las mismas.

Empezaremos comentando que para MATLAB el carácter tanto por ciento (%) indica comienzo de comentario. Los comentarios nos sirven para guiarnos en lo que hace el programa, ya que son las notas que se nos quieren hacer constar para otras personas que analicen el programa. Los comentarios como es obvio no se ejecutan.

MATLAB es fundamentalmente un programa para cálculo matricial. Inicialmente se utilizará MATLAB como programa interactivo, en el que se irán definiendo las matrices, los vectores y las expresiones que los combinan y obteniendo los resultados sobre la marcha. Si estos resultados son asignados a otras variables podrán ser utilizados posteriormente en otras expresiones. En este sentido MATLAB sería como una potente calculadora matricial.

Para definir una matriz no hace falta declararlas o establecer de antemano su tamaño (de hecho, se puede definir un tamaño y cambiarlo posteriormente). MATLAB determina el número de filas y de columnas en función del número de elementos que se proporcionan (o se utilizan).

Las matrices se definen o introducen por filas; los elementos de una misma fila están separados por blancos o comas, mientras que las filas están separadas por pulsaciones intro o por caracteres punto y coma (;). Por ejemplo, el siguiente comando define una matriz A de dimensión (3×3):

```
>> A=[1 2 3; 4 5 6; 7 8 9]
```

La respuesta del programa es la siguiente:

```
A =

     1     2     3
     4     5     6
     7     8     9
```

A partir de este momento la matriz A está disponible para hacer cualquier tipo de operación con ella (además de valores numéricos, en la definición de una matriz o vector se pueden utilizar expresiones y funciones matemáticas). Por ejemplo, una sencilla operación con A es hallar su matriz traspuesta. En MATLAB el apóstrofo (') es el símbolo de transposición matricial. Para calcular A' (traspuesta de A) basta teclear lo siguiente (se añade a continuación la respuesta del programa):

```
>> A'

ans =

     1     4     7
     2     5     8
     3     6     9
```

Como el resultado de la operación no ha sido asignado a ninguna otra matriz, MATLAB utiliza un nombre de variable por defecto (ans, de answer), que contiene el resultado de la última operación. La variable ans puede ser utilizada como operando en la siguiente expresión que se introduzca.

También podría haberse asignado el resultado a otra matriz llamada B:

```
>> B=A'

B =

     1     4     7
     2     5     8
     3     6     9
```

Ahora ya están definidas las matrices A y B, y es posible seguir operando con ellas. Por ejemplo, se puede hacer el producto B*A (deberá resultar una matriz simétrica):

```
>> B*A

ans =

    66    78    90
    78    93   108
    90   108   126
```

En MATLAB se accede a los elementos de un vector poniendo el índice entre paréntesis (por ejemplo x(3) ó x(i)). Los elementos de las matrices se acceden poniendo los dos índices entre paréntesis, separados por una coma (por ejemplo A(1,2) ó A(i,j)). Las matrices se almacenan por columnas (aunque se introduzcan por filas, como se ha dicho antes), y teniendo en cuenta esto puede accederse a cualquier elemento de una matriz con un sólo subíndice. Por ejemplo, si A es una matriz (3×3) se obtiene el mismo valor escribiendo A(1,2) que escribiendo A(4).

Invertir una matriz es casi tan fácil como trasponerla. A continuación se va a definir una nueva matriz A -no singular- en la forma:

```
>> A=[1 4 -3; 2 1 5; -2 5 3]

A =

     1     4    -3
     2     1     5
    -2     5     3
```

Ahora se va a calcular la inversa de A y el resultado se asignará a B. Para ello basta hacer uso de la función inv( ) (la precisión o número de cifras con que se muestra el resultado se puede cambiar con el menú File/Preferences/General):

```
>>B=inv(A)

B =

    0.1803    0.2213   -0.1885
    0.1311    0.0246    0.0902
   -0.0984    0.1066    0.0574
```

Para comprobar que este resultado es correcto basta pre-multiplicar A por B;

```
>> B*A

ans =

    1.0000    0.0000    0.0000
    0.0000    1.0000    0.0000
    0.0000    0.0000    1.0000
```

MATLAB puede operar con matrices por medio de operadores y por medio de funciones. Se han visto ya los operadores suma (+), producto (*) y traspuesta ('), así como la función invertir inv( ).

Los **operadores matriciales** de MATLAB son los siguientes:

+ adición o suma
– sustracción o resta
* multiplicación
' traspuesta
^ potenciación
\ división-izquierda
/ división-derecha
.* producto elemento a elemento
./ y .\ división elemento a elemento
.^ elevar a una potencia elemento a elemento

Estos operadores se aplican también a las variables o valores escalares, aunque con algunas diferencias. Todos estos operadores son coherentes con las correspondientes operaciones matriciales: no se puede por ejemplo sumar matrices que no sean del mismo tamaño. Si los operadores no se usan de modo correcto se obtiene un mensaje de error.

MATLAB utiliza el operador de división / para dividir por un escalar todos los elementos de una matriz o un vector. Esto no constituye ninguna sorpresa. Sin embargo, el uso que se describe a continuación sí requiere más atención.

MATLAB utiliza los operadores de división para la resolución de sistemas de ecuaciones lineales.

Por su gran importancia, estos operadores requieren una explicación detenida. Considérese el siguiente sistema de ecuaciones lineales,

$$\mathbf{A \cdot x = b}$$

donde x y b son vectores columna, y A una matriz cuadrada invertible. La resolución de este sistema de ecuaciones se puede escribir en las 2 formas siguientes (¡Atención a la 2ª forma, basada en la barra invertida (\), que puede resultar un poco extraña!):

x = inv(A)*b

x = A\b

Así, pues, el operador división-izquierda por una matriz (barra invertida \) equivale a pre-multiplicar por la inversa de esa matriz. En realidad, este operador es más general y más inteligente de lo que aparece en el ejemplo anterior: el operador división-izquierda es aplicable aunque la matriz no tenga inversa e incluso no sea cuadrada, en cuyo caso la solución que se obtiene (por lo general) es la que proporciona el método de los mínimos cuadrados. Cuando la matriz es triangular o simétrica aprovecha esta circunstancia para reducir el número de operaciones aritméticas. En algunos casos se obtiene una solución con no más de r elementos distintos de cero, siendo r el rango de la matriz.

Esto puede estar basado en que la matriz se reduce a forma de escalón y se resuelve el sistema dando valor cero a las variables libres o independientes. Por ejemplo,

```
>> A=[1 2], b=[2]
A =
     1     2
b =
     2
>> x=A\b
x =
     0
     1
```

que es la solución obtenida dando valor cero a la variable independiente x(1). Por otra parte, en el caso de un sistema de ecuaciones redundante (o sobre-determinado) el resultado de MATLAB es el punto más "cercano" -en el sentido de mínima norma del error- a las ecuaciones dadas (aunque no cumpla exactamente ninguna de ellas). Véase el siguiente ejemplo de tres ecuaciones formadas por una recta que no pasa por el origen y los dos ejes de coordenadas:

```
>> A=[1 2; 1 0; 0 1], b=[2 0 0]'

A =

     1     2
     1     0
     0     1

b =

     2
     0
     0

>> x=A\b, resto=A*x-b

x =

      0.3333
      0.6667

resto =

     -0.3333
      0.3333
      0.6667
```

Si la matriz es singular o está muy mal escalada, el operador \ da un aviso (warning), pero proporciona una solución.

Ya han aparecido algunos ejemplos de variables y expresiones matriciales. Ahora se va a tratar de generalizar un poco lo visto hasta ahora.

Una variable es un nombre que se da a una entidad numérica, que puede ser una matriz, un vector o un escalar. El valor de esa variable, e incluso el tipo de entidad numérica que representa, puede cambiar a lo largo de una sesión de MATLAB o a lo largo de la ejecución de un programa. La forma más normal de cambiar el valor de una variable es colocándola a la izquierda del operador de asignación (=).

Una expresión de MATLAB puede tener las dos formas siguientes: primero, asignando su resultado a una variable,

variable = expresión

y segundo evaluando simplemente el resultado del siguiente modo,

expresión

en cuyo caso el resultado se asigna automáticamente a una variable interna de MATLAB llamada ans (de answer) que almacena el último resultado obtenido. Se considera por defecto que una expresión termina cuando se pulsa intro. Si se desea que una expresión continúe en la línea siguiente, hay que introducir tres puntos (...) antes de pulsar intro. También se pueden incluir varias expresiones en una misma línea separándolas por comas (,) o puntos y comas (;).

Si una expresión termina en punto y coma (;) su resultado se calcula, pero no se escribe en pantalla. Esta posibilidad es muy interesante, tanto para evitar la escritura de resultados intermedios, como para evitar la impresión de grandes cantidades de números cuando se trabaja con matrices de gran tamaño.

A semejanza de C, MATLAB distingue entre mayúsculas y minúsculas en los nombres de variables. Los nombres de variables deben empezar siempre por una letra y pueden constar de hasta 63 letras y números.

MATLAB ofrece también la posibilidad de crear una matriz a partir de matrices previas ya definidas, por varios posibles caminos:

– recibiendo alguna de sus propiedades (como por ejemplo el tamaño),

– por composición de varias submatrices más pequeñas,

– modificándola de alguna forma.

A continuación se describen algunas de las funciones que crean una nueva matriz a partir de otra o de otras, comenzando por dos funciones auxiliares:

[m,n]=size(A) devuelve el número de filas y de columnas de la matriz A. Si la matriz es cuadrada basta recoger el primer valor de retorno

n=length(x) calcula el número de elementos de un vector x

Para MATLAB una matriz definida sin ningún elemento entre los corchetes es una matriz que existe, pero que está vacía, o lo que es lo mismo que tiene dimensión cero.

El lenguaje de programación de MATLAB dispone de los siguientes **operadores relacionales**:

< menor que

> mayor que

<= menor o igual que

>= mayor o igual que

== igual que

~= distinto que

Los **operadores lógicos** de MATLAB son los siguientes:

& and (función equivalente: and(A,B)). Se evalúan siempre ambos operandos, y el resultado es true sólo si ambos son true.

&& and breve: si el primer operando es false ya no se evalúa el segundo, pues el resultado final ya no puede ser más que false.

| or (función equivalente: or(A,B)). Se evalúan siempre ambos operandos, y el resultado es false sólo si ambos son false.

|| or breve: si el primer operando es true ya no se evalúa el segundo, pues el resultado final no puede ser más que true.

~ negación lógica (función equivalente: not(A))

xor(A,B) realiza un "or exclusivo", es decir, devuelve 0 en el caso en que ambos sean 1 ó ambos sean 0.

Los operadores lógicos se combinan con los relacionales para poder comprobar el cumplimiento de condiciones múltiples.

Funciones matemáticas elementales que operan de modo escalar: Estas funciones, que comprenden las funciones matemáticas trascendentales y otras funciones básicas, cuando se aplican a una matriz actúan sobre cada elemento de la matriz como si se tratase de un escalar. Por tanto, se aplican de la misma forma a escalares, vectores y matrices. Algunas de las funciones de este grupo son las siguientes:

sin(x) seno

cos(x) coseno

tan(x) tangente

asin(x) arco seno

acos(x) arco coseno

atan(x) arco tangente (devuelve un ángulo entre $-\pi/2$ y $+\pi/2$)

atan2(x) arco tangente (devuelve un ángulo entre $-\pi$ y $+\pi$); se le pasan 2 argumentos, proporcionales al seno y al coseno

log(x) logaritmo natural

log10(x) logaritmo decimal

exp(x) función exponencial

sqrt(x) raíz cuadrada

rem(x,y) resto de la división (2 argumentos que no tienen que ser enteros)

mod(x,y) similar a rem

round(x) redondeo hacia el entero más próximo

fix(x) redondea hacia el entero más próximo a 0

abs(x) valores absolutos

Funciones que actúan sobre vectores: Las siguientes funciones sólo actúan sobre vectores (no sobre matrices, ni sobre escalares):

[xm,im]=max(x) máximo elemento de un vector. Devuelve el valor máximo xm y la posición que ocupa im

min(x) mínimo elemento de un vector. Devuelve el valor mínimo y la posición que ocupa

sum(x) suma de los elementos de un vector

cumsum(x) devuelve el vector suma acumulativa de los elementos de un vector (cada elemento del resultado es una suma de elementos del original)

mean(x) valor medio de los elementos de un vector

std(x) desviación típica

prod(x) producto de los elementos de un vector

cumprod(x) devuelve el vector producto acumulativo de los elementos de un vector

[y,i]=sort(x) ordenación de menor a mayor de los elementos de un vector x. Devuelve el vector ordenado y, y un vector i con las posiciones iniciales en x de los elementos en el vector ordenado y.

Hipermatrices (arrays de más de dos dimensiones): MATLAB permite trabajar con hipermatrices, es decir con matrices de más de dos dimensiones. (figura 9-1)

Una posible aplicación es almacenar con un único nombre distintas matrices del mismo tamaño (resulta una hipermatriz de 3 dimensiones). Los elementos de una hipermatriz pueden ser números, caracteres, estructuras, y vectores o matrices de celdas.

El tercer subíndice representa la tercera dimensión: la "profundidad" de la hipermatriz.

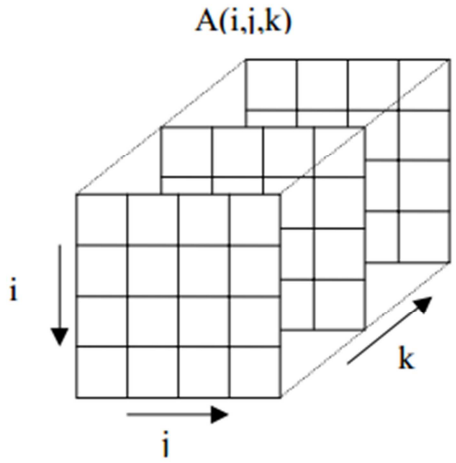


Figura 9-1. Hipermatriz de tres dimensiones

Las funciones para trabajar con estas hipermatrices están en el sub-directorio toolbox\matlab\datatypes. Las funciones que operan con matrices de más de dos dimensiones son análogas a las funciones vistas previamente, aunque con algunas diferencias. Por ejemplo, las siguientes sentencias generan, en dos pasos, una matriz de 2×3×2:

```
>> AA(:, :, 1)=[1 2 3; 4 5 6]   % matriz inicial

AA =

     1     2     3
     4     5     6

>> AA(:, :, 2)=[2 3 4; 5 6 7]   % se añade una segunda matriz

AA(:, :, 1) =

     1     2     3
     4     5     6

AA(:, :, 2) =

     2     3     4
     5     6     7
```

Estructuras: Una estructura (struct) es una agrupación de datos de tipo diferente bajo un mismo nombre. Estos datos se llaman miembros (members) o campos (fields). Una estructura es un nuevo tipo de dato, del que luego se pueden crear muchas variables (objetos o instances). Por ejemplo, la estructura alumno puede contener los campos nombre (una cadena de caracteres) y carnet (un número).

En MATLAB la estructura alumno se crea creando un objeto de dicha estructura. A diferencia de otros lenguajes de programación, no hace falta definir previamente el modelo o patrón de la estructura. Una posible forma de hacerlo es crear uno a uno los distintos campos, como en el ejemplo siguiente:

```
>> alu.nombre='Miguel'
alu =  nombre: 'Miguel'
>> alu.carnet=75482
alu =
        nombre: 'Miguel'
        carnet: 75482
>> alu
alu =
        nombre: 'Miguel'
        carnet: 75482
```

Se accede a los miembros o campos de una estructura por medio del operador punto (.), que une el nombre de la estructura y el nombre del campo (por ejemplo: alu.nombre).

También puede crearse la estructura por medio de la función struct(), como por ejemplo,

```
>> al = struct('nombre', 'Ignacio', 'carnet', 76589)
al =
        nombre: 'Ignacio'
        carnet: 76589
```

Los nombres de los campos se pasan a la función struct() entre apóstrofos ('), seguidos del valor que se les quiere dar. Este valor puede ser la cadena vacía ('') o la matriz vacía ([]).

Pueden crearse vectores y matrices (e hipermatrices) de estructuras. Por ejemplo, la sentencia,

```
>> alum(10) = struct('nombre', 'Ignacio', 'carnet', 76589)
```

crea un vector de 10 elementos cada uno de los cuales es una estructura tipo alumno. Sólo el elemento 10 del vector es inicializado con los argumentos de la función struct(); el resto de los campos se inicializan con una cadena vacía o una matriz vacía. Para dar valor a los campos de los elementos restantes se puede utilizar un bucle for con sentencias del tipo:

```
>> alum(i).nombre='Noelia', alum(i).carnet=77524;
```

MATLAB permite añadir un nuevo campo a una estructura en cualquier momento. La siguiente sentencia añade el campo edad a todos los elementos del vector alum, aunque sólo se da valor al campo del elemento 5,

```
>> alum(5).edad=18;
```

Para ver el campo edad en los 10 elementos del vector puede teclearse el comando:

```
>> alum.edad
```

Bifurcaciones y bucles: MATLAB posee un lenguaje de programación que –como cualquier otro lenguaje– dispone de sentencias para realizar bifurcaciones y bucles. Las bifurcaciones permiten realizar una u otra operación según se cumpla o no una determinada condición. Los bucles permiten repetir las mismas o análogas operaciones sobre datos distintos.

El cuerpo de estas sentencias viene determinado por un **end** al final. Algunas de estas sentencias iterativas son las siguientes:

Sentencia if: En su forma más simple, la sentencia **if** se escribe en la forma siguiente:

```
if condicion
    sentencias
end
```

Existe también la bifurcación múltiple, en la que pueden concatenarse tantas condiciones como se desee, y que tiene la forma:

```
if condicion1
    bloque1
elseif condicion2
    bloque2
elseif condicion3
    bloque3
else  % opción por defecto para cuando no se cumplan las condiciones 1,2,3
    bloque4
end
```

donde la opción por defecto **else** puede ser omitida: si no está presente no se hace nada en caso de que no se cumpla ninguna de las condiciones que se han chequeado.

Sentencia switch: La sentencia **switch** realiza una función análoga a un conjunto de **if...elseif** concatenados. Su forma general es la siguiente:

```
switch switch_expresion
    case case_expr1,
        bloque1
    case {case_expr2, case_expr3, case_expr4,...}
        bloque2
    ...
    otherwise,    % opción por defecto
        bloque3
end
```

Al principio se evalúa la **switch_expresion**, cuyo resultado debe ser un número escalar o una cadena de caracteres. Este resultado se compara con las **case_expr**, y se ejecuta el bloque de sentencias que corresponda con ese resultado. Si ninguno es igual a **switch_expresion** se ejecutan las sentencias correspondientes a **otherwise.**

Sentencia for: La sentencia **for** repite un conjunto de sentencias un número predeterminado de veces. La siguiente construcción ejecuta sentencias con valores de i de 1 a n, variando de uno en uno.

```
for i=1:n
    sentencias
end
```

o bien,

```
for i=vectorValores
    sentencias
end
```

donde vectorValores es un vector con los distintos valores que tomará la variable i.

En el siguiente ejemplo se presenta el caso más general para la variable del bucle (valor_inicial: incremento: valor_final); el bucle se ejecuta por primera vez con i=n, y luego i se va reduciendo de 0.2 en 0.2 hasta que llega a ser menor que 1, en cuyo caso el bucle se termina:

```
for i=n:-0.2:1
    sentencias
end
```

En el siguiente ejemplo se presenta una estructura correspondiente a dos bucles anidados. La variable j es la que varía más rápidamente (por cada valor de i, j toma todos sus posibles valores):

```
for i=1:m
    for j=1:n
        sentencias
    end
end
```

Cuando se introducen interactivamente en la línea de comandos, los bucles for se ejecutan sólo después de introducir la sentencia end que los completa.

Sentencia while: La estructura del bucle **while** es muy similar a la de C/C++/Java. Su sintaxis es la siguiente:

```
while condicion
    sentencias
end
```

donde condicion puede ser una expresión vectorial o matricial. Las sentencias se siguen ejecutando mientras haya elementos distintos de cero en condicion, es decir, mientras haya algún o algunos elementos true. El bucle se termina cuando todos los elementos de condicion son false (es decir, cero).

Se verá a continuación una forma sencilla de **leer variables desde teclado y escribir mensajes en la pantalla del PC**. Más adelante se considerarán otros modos más generales –y complejos– de hacerlo.

La función **input** permite imprimir un mensaje en la línea de comandos de MATLAB y recuperar como valor de retorno un valor numérico o el resultado de una expresión tecleada por el usuario. Después de imprimir el mensaje, el programa espera que el usuario teclee el valor numérico o la expresión. Cualquier expresión válida de MATLAB es aceptada por este comando. El usuario puede teclear simplemente un vector o una matriz. En cualquier caso, la expresión introducida es evaluada con los valores actuales de las variables de MATLAB y el resultado se devuelve como valor de retorno. Véase un ejemplo de uso de esta función:

```
>> n = input('Teclee el número de ecuaciones')
```

Otra posible forma de esta función es la siguiente (obsérvese el parámetro 's'):

```
>> nombre = input('¿Cómo te llamas?','s')
```

En este caso el texto tecleado como respuesta se lee y se devuelve sin evaluar, con lo que se almacena en la cadena nombre. Así pues, en este caso, si se teclea una fórmula, se almacena como texto sin evaluarse.

La función **disp** permite imprimir en pantalla un mensaje de texto o el valor de una matriz, pero sin imprimir su nombre. En realidad, **disp** siempre imprime vectores y/o matrices: las cadenas de caracteres son un caso particular de vectores. Considérense los siguientes ejemplos de cómo se utiliza:

```
>> disp('El programa ha terminado')
```

Ficheros *.m: Los ficheros con extensión (.m) son ficheros de texto sin formato (ficheros ASCII) que constituyen el centro de la programación en MATLAB. Ya se han utilizado en varias ocasiones. Estos ficheros se crean y modifican con un editor de textos cualquiera. En el caso de MATLAB ejecutado en un PC bajo Windows, lo mejor es utilizar su propio editor de textos, que es también Debugger.

Existen dos tipos de ficheros *.m, los ficheros de comandos (llamados scripts en inglés) y las funciones. Los primeros contienen simplemente un conjunto de comandos que se ejecutan sucesivamente cuando se teclea el nombre del fichero en la línea de comandos de MATLAB o se incluye dicho nombre en otro fichero *.m. Un fichero de comandos puede llamar a otros ficheros de comandos. Si un fichero de comandos se llama desde de la línea de comandos de MATLAB, las variables que crea pertenecen al espacio de trabajo base de MATLAB, y permanecen en él cuando se termina la ejecución de dicho fichero.

Las funciones permiten definir funciones enteramente análogas a las de MATLAB, con su nombre, sus argumentos y sus valores de retorno. Los ficheros *.m que definen funciones permiten extender las posibilidades de MATLAB; de hecho existen bibliotecas de ficheros *.m que se venden (toolkits) o se distribuyen gratuitamente (a través de Internet). Las funciones definidas en ficheros *.m se caracterizan porque la primera línea (que no sea un comentario) comienza por la palabra function, seguida por los valores de retorno (entre corchetes [ ] y separados por comas, si hay más de uno), el signo igual (=) y el nombre de la función, seguido de los argumentos (entre paréntesis y separados por comas).

Recuérdese que un fichero *.m puede llamar a otros ficheros *.m, e incluso puede llamarse a sí mismo de forma recursiva. Los ficheros de comandos se pueden llamar también desde funciones, en cuyo caso las variables que se crean pertenecen al espacio de trabajo de la función. El espacio de trabajo de una función es independiente del espacio de trabajo base y del espacio de trabajo de las demás funciones. Esto implica por ejemplo que no puede haber colisiones entre nombres de varia bles: aunque varias funciones tengan una variable llamada A, en realidad se trata de variables completamente distintas (a no ser que A haya sido declarada como variable global).

Definición de funciones: La primera línea de un fichero llamado name.m que define una función tiene la forma:

```
function [lista de valores de retorno] = name(lista de argumentos)
```

donde name es el nombre de la función. Entre corchetes y separados por comas van los valores de retorno (siempre que haya más de uno), y entre paréntesis también separados por comas los argumentos. Puede haber funciones sin valor de retorno y también sin argumentos. Recuérdese que los argumentos son los datos de la función y los valores de retorno sus resultados. Si no hay valores de retorno se omiten los corchetes y el signo igual (=); si sólo hay un valor de retorno no hace falta poner corchetes. Tampoco hace falta poner paréntesis si no hay argumentos.

Gráficos bidimensionales: La función **plot** es la función clave de todos los gráficos 2-D en MATLAB. Ya se ha dicho que el elemento básico de los gráficos bidimensionales es el vector. Se utilizan también cadenas de 1, 2 ó 3 caracteres para indicar colores y tipos de línea. La función plot(), en sus diversas variantes, no hace otra cosa que dibujar vectores. Un ejemplo muy sencillo de esta función, en el que se le pasa un único vector como argumento, es el siguiente:

```
>> x=[1 3 2 4 5 3]
x =
     1   3   2   4   5   3
>> plot(x)
```

El resultado de este comando es que se abre una ventana mostrando el gráfico de la figura 9-2. Por defecto, los distintos puntos del gráfico se unen con una línea continua. También por defecto, el color que se utiliza para la primera línea es el azul.

Una segunda forma de utilizar la función plot() es con dos vectores como argumentos. En este caso los elementos del segundo vector se representan en ordenadas frente a los valores del primero, que se representan en abscisas.

```
>> x=[1 6 5 2 1]; y=[1 0 4 3 1];
>> plot(x,y)
```

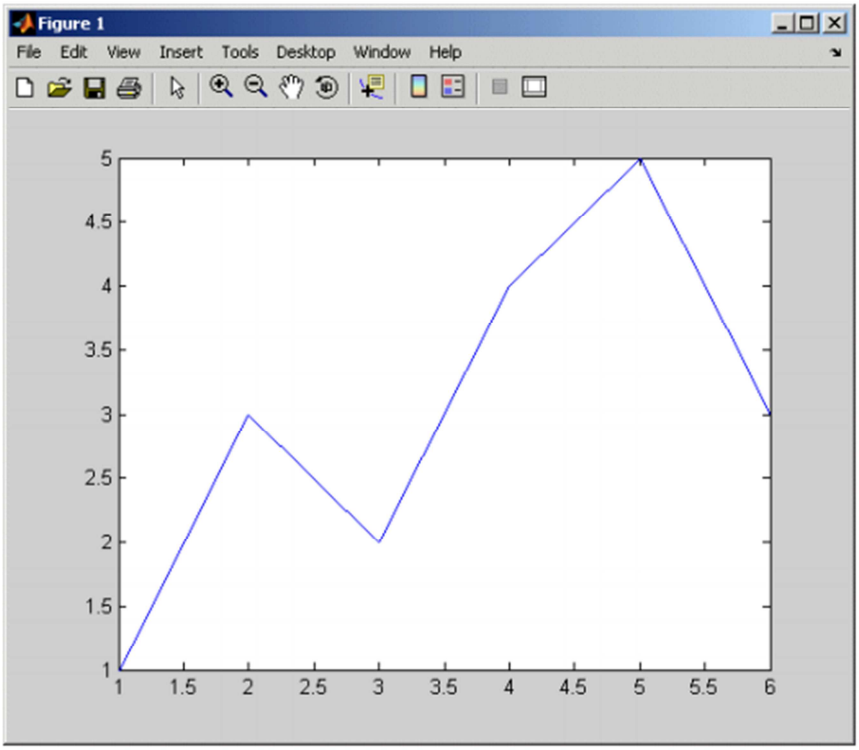


Figura 9-2. Gráfico del vector x=[1 3 2 4 5 3]

En la sección anterior se ha visto cómo la tarea fundamental de la función plot() era dibujar los valores de un vector en ordenadas, frente a los valores de otro vector en abscisas. En el caso general esto exige que se pasen como argumentos un par de vectores. En realidad, el conjunto básico de argumentos de esta función es una tripleta formada por dos vectores y una cadena de 1, 2 ó 3 caracteres que indica el color y el tipo de línea o de marker. En la tabla siguiente se pueden observar las distintas posibilidades.

Símbolo	Color	Símbolo	Marcadores (markers)
y	yellow	.	puntos
m	magenta	o	círculos
c	cyan	x	marcas en x
r	red	+	marcas en +
g	green	*	marcas en *
b	blue	s	marcas cuadradas (square)
w	white	d	marcas en diamante (diamond)
k	black	^	triángulo apuntando arriba
		v	triángulo apuntando abajo
Símbolo	**Estilo de línea**	>	triángulo apuntando a la dcha
-	líneas continuas	<	triángulo apuntando a la izda
:	líneas a puntos	p	estrella de 5 puntas
-.	líneas a barra-punto	h	estrella se seis puntas
--	líneas a trazos		

Figura 9-3. Colores, markers y estilos de línea

Existe la posibilidad de añadir líneas a un gráfico ya existente, sin destruirlo o sin abrir una nueva ventana. Se utilizan para ello los comandos **hold on** y **hold off**. El primero de ellos hace que los gráficos sucesivos respeten los que ya se han dibujado en la figura (es posible que haya que modificar la escala de los ejes); el comando **hold off** deshace el efecto de **hold on**. El siguiente ejemplo muestra cómo se añaden las gráficas de x2 y x3 a la gráfica de x previamente creada (cada una con un tipo de línea diferente):

```
>> plot(x)
>> hold on
>> plot(x2,'--')
>> plot(x3,'-.')
>> hold off
```

Si se llama a la función **figure** sin argumentos, se crea una nueva ventana gráfica con el número consecutivo que le corresponda. El valor de retorno es dicho número.

Por otra parte, el comando **figure(n)** hace que la ventana n pase a ser la ventana o figura activa. Si dicha ventana no existe, se crea una nueva ventana con el número consecutivo que le corresponda (que se puede obtener como valor de retorno del comando). La función **close** cierra la figura activa, mientras que **close(n)** cierra la ventana o figura número n.

Entrada de puntos con el ratón: Se realiza mediante la función **ginput**, que permite introducir las coordenadas del punto sobre el que está el cursor, al clicar (o al pulsar una tecla). Algunas formas de utilizar esta función son las siguientes:

[x,y] = ginput lee un número indefinido de puntos –cada vez que se clica o se pulsa una tecla cualquiera– hasta que se termina pulsando la tecla intro

[x,y] = ginput(n) lee las coordenadas de n puntos

Gráficos tridimensionales: dibujo de mallados: funciones meshgrid, mesh y surf:

Ahora se verá con detalle cómo se puede dibujar una función de dos variables (z=f(x,y)) sobre un dominio rectangular. Se verá que también se pueden dibujar los elementos de una matriz como función de los dos índices.

Sean x e y dos vectores que contienen las coordenadas en una y otra dirección de la retícula (grid) sobre la que se va a dibujar la función. Hay que crear dos matrices X (cuyas filas son copias de x) e Y (cuyas columnas son copias de y). Estas matrices se crean con la función **meshgrid**. Estas matrices representan respectivamente las coordenadas x e y de todos los puntos de la retícula.

La matriz de valores Z se calcula a partir de las matrices de coordenadas X e Y. Finalmente hay que dibujar esta matriz Z con la función **mesh**, cuyos elementos son función elemento a elemento de los elementos de X e Y.

La función **mesh** dibuja en perspectiva una función en base a una retícula de líneas de colores, rodeando cuadriláteros del color de fondo, con eliminación de líneas ocultas. Esta función dibuja en perspectiva una función en base a una retícula de líneas de colores, rodeando cuadriláteros del color de fondo, con eliminación de líneas ocultas.

Con la función **surf** en vez de líneas aparece una superficie faceteada, también con eliminación de líneas ocultas. El color de las facetas depende también del valor de la función.

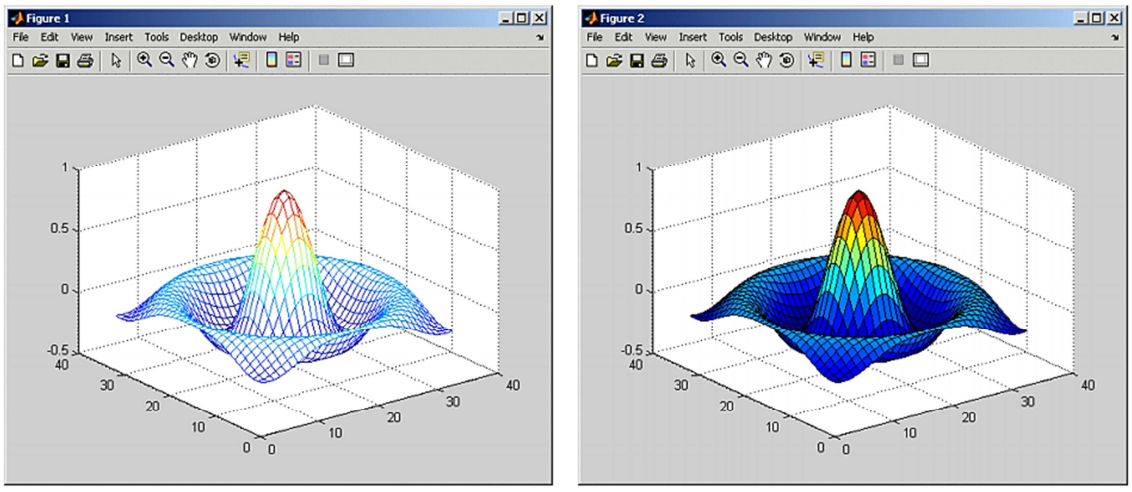


Figura 9-4. Diferecias entre mesh y surf

Una forma distinta de representar funciones tridimensionales es por medio de isolíneas o curvas de nivel empleando el comando **contour**.

ANEXO: Interfaz Gráfica de Usuario (GUI)

Una interfaz gráfica es el vínculo entre el usuario y un programa computacional, constituida generalmente por un conjunto de comandos o menús, instrumentos y métodos por medio de los cuales el usuario se comunica con el programa durante las operaciones que se desean realizar, facilitando la entrada y salida de datos e información.

Constituye una de las partes más importantes de cualquier programa puesto que determina qué tan factible y preciso será el desempeño del programa ante los comandos que el usuario pretenda ejecutar. Aunque un programa sea muy poderoso, si se manipula por medio de una interfaz pobremente elaborada, tendrá poco valor para un usuario inexperto. Es por esto que las interfaces gráficas tienen una gran importancia para usuarios inexpertos o avanzados de cualquier programa ya que facilitan su uso.

Una interfaz gráfica consta de botones, menús, ventanas, etc, que permiten utilizar de una manera muy simple y en ocasiones casi intuitiva programas realizados en ambientes como Windows y Linux. Las interfaces gráficas también se conocen como interfaces de usuario.

El nombre en inglés de las interfaces gráficas es Graphical User Interfase y se denominan GUI, por lo que nosotros también nos referiremos a ellas de la misma manera.

Existen diferentes lenguajes de programación que permiten crear GUIs tales como Visual C, Visual Basic, TK y MATLAB por mencionar algunos. Todos ellos permiten usar diferentes controles y tienen distintas maneras de programarlos.

MATLAB nos permite realizar GUIs de una manera sencilla usando una herramienta llamada GUIDE (GUI Development Environment). En este capítulo presentaremos una introducción muy completa a las técnicas en MATLAB para crear interfaces gráficas.

GUIDE (Graphical User Interfase Development Environment) es un juego de herramientas que se extiende por completo en el soporte de MATLAB, diseñadas para crear GUIs (Graphical User Interfaces) fácil y rápidamente, prestando ayuda en el diseño y presentación de los controles de la interfaz, reduciendo la labor al grado de seleccionar, tirar, arrastrar y personalizar propiedades.

Una vez que los controles están en posición se editan las funciones de llamada (Callback) de cada uno de ellos, escribiendo el código de MATLAB que se ejecutará cuando el control sea utilizado. Siempre será difícil diseñar GUIs, pero no debería ser difícil implementarlas.

GUIDE está diseñado para hacer menos tedioso el proceso de aplicación de la interfaz gráfica y obviamente para trabajar como herramienta de trazado de GUIs. Entre sus poderosos componentes está el editor de propiedades (property editor), éste se encuentra disponible en cualquier momento que se esté lidiando con los controles de MATLAB. El editor de propiedades por separado se puede concebir como una herramienta de trazado, y asistente de codificación (revisión de nombres y valores de propiedades). Cuando se fusiona con el panel de control, el editor de menú, y herramienta de alineación, resulta una combinación que brinda un inigualable control de los gráficos en MATLAB.

El beneficio que proporciona el uso de GUIs es evidente, ya que permiten al usuario ejecutar cómodamente código desarrollado en MATLAB sin necesidad de cumplir la incómoda sintaxis funcional necesaria cuando se trabaja desde la línea de órdenes. A diferencia de la ejecución de funciones o scripts de MATLAB, la ejecución de GUIs no predetermina el flujo de ejecución del código. Es el usuario, a través de su interacción con el GUI, el que determina el orden en que se ejecutan las diferentes órdenes y funciones desarrolladas. Otra diferencia importante es que la ejecución no termina cuando finaliza la ejecución del script o función, sino que el GUI permanece abierto, permitiendo al usuario invocar la ejecución de ese u otro código desarrollado.

El desarrollo de GUIs se realiza en dos etapas:

- Diseño de los componentes (controles, menús y ejes) que formarán el GUI.
- Codificación de la respuesta de cada uno de los componentes ante la interacción del usuario.

A la herramienta GUIDE se accede de varias maneras, la primera de ellas es tecleando guide en la ventana de comando.

>> guide

Otra manera de entrar en GUIDE es través de la opción File, haciendo clic en New y por último eligiendo la opción GUI, (como se muestra en la figura).

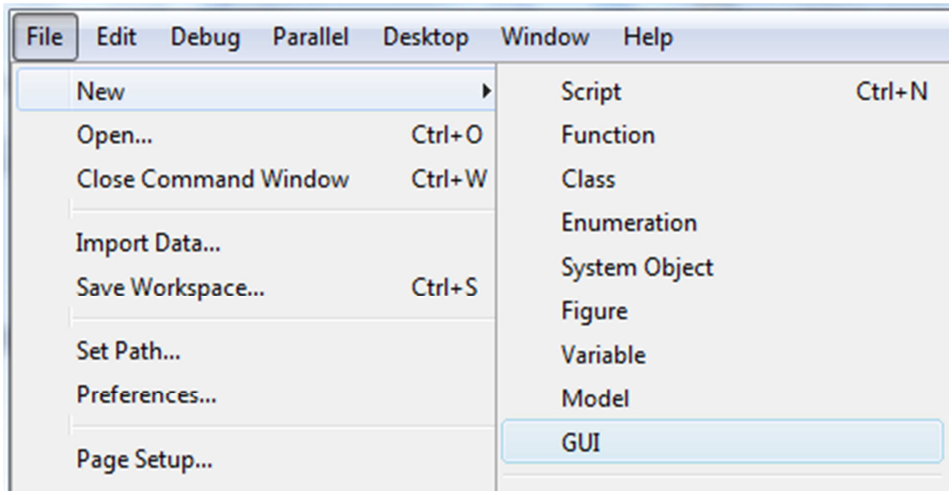


Figura 10-1. Menú nuevo GUI

A continuación, se presenta el siguiente cuadro de diálogo, correspondiente con la Ventana de inicio de GUI:

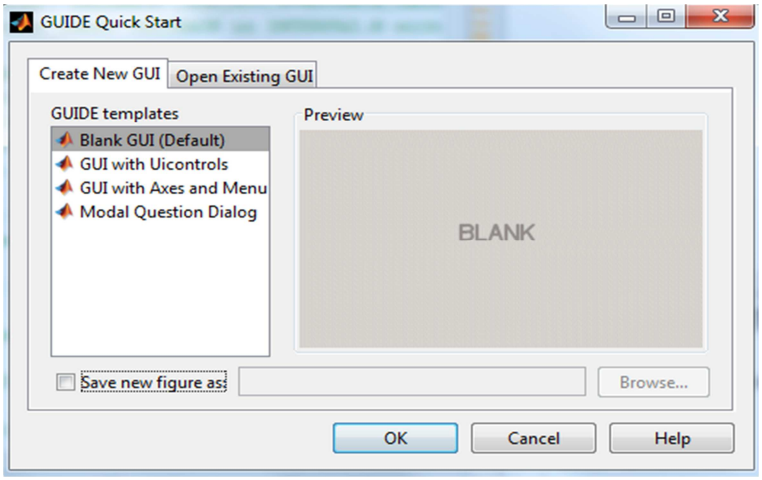


Figura 10-2. Panel GUIDE rápido

Donde se presentan las siguientes opciones:

a) Blank GUI (Default)

La opción de interfaz gráfica de usuario en blanco (viene predeterminada), nos presenta un formulario nuevo, en el cual podemos diseñar nuestro programa.

b) GUI with Uicontrols

Esta opción presenta un ejemplo en el cual se calcula la masa, dada la densidad y el volumen, en alguno de los dos sistemas de unidades. Podemos ejecutar este ejemplo y obtener resultados.

c) GUI with Axes and Menu

Esta opción es otro ejemplo el cual contiene el menú File con las opciones Open, Print y Close. En el formulario tiene un Popup menu, un push button y un objeto Axes, podemos ejecutar el programa eligiendo alguna de las seis opciones que se encuentran en el menú despegable y haciendo click en el botón de comando.

d) Modal Question Dialog

Con esta opción se muestra en la pantalla un cuadro de diálogo común, el cual consta de una pequeña imagen, una etiqueta y dos botones Yes y No, dependiendo del botón que se presione, el GUI retorna el texto seleccionado (la cadena de caracteres 'Yes' o 'No').

Si elegimos la primera opción, Blank GUI, tenemos:

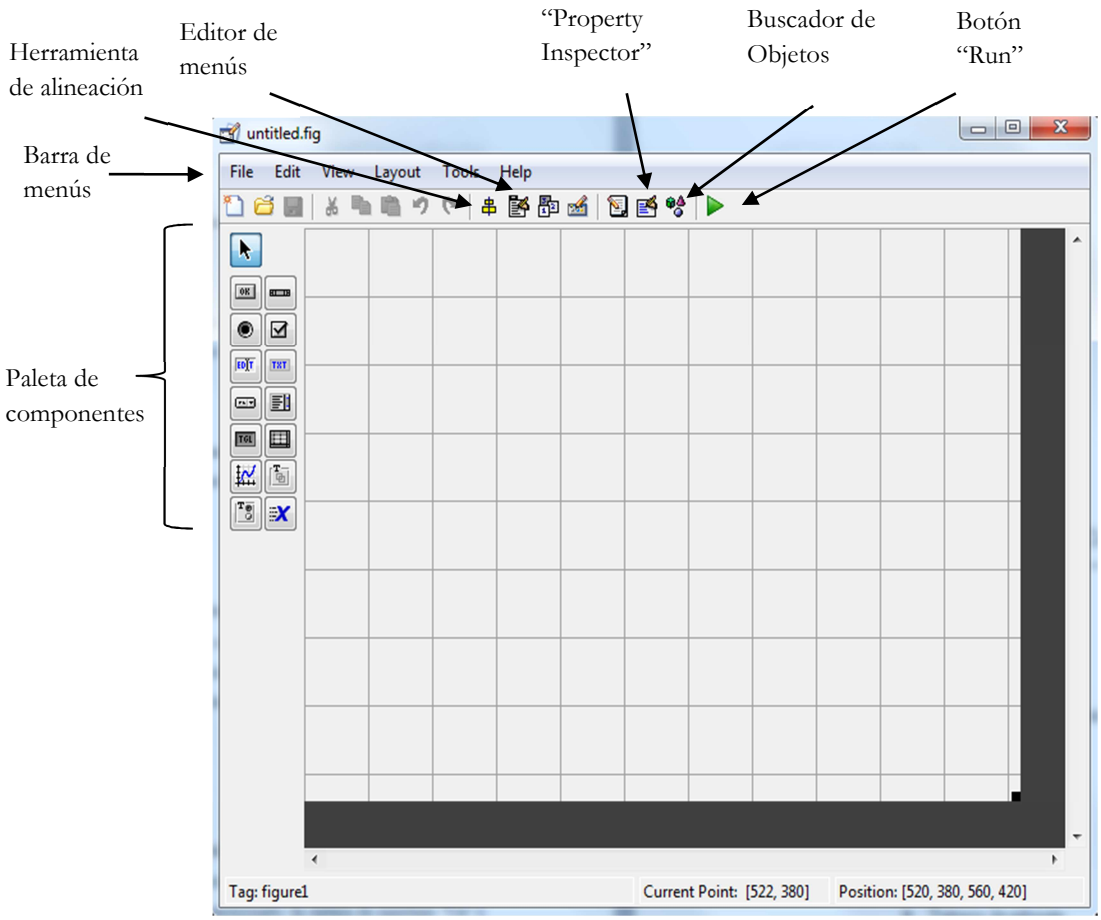


Figura 10-3. Aspecto GUI

Los componentes principales de GUIDE son:

- Barra de Menús: Aquí se encuentran las funciones elementales de Edición de GUIs.
- Paleta de Componentes (component Palette): Aquí se encuentran los uicontrols, estos componentes permiten seleccionar los controles (objetos) que son los que se muestran en la figura.

- La Barra de Herramientas: En ella se encuentran los siguientes botones
  - Botón de ejecución (Run button): Al presionarse crea la figura de la interfaz diseñada en el Layout Área.

- Alineación de Componentes (Alignment tool): esta opción permite alinear los componentes que se encuentra en el área de trabajo (Layout Área) de manera personalizada.

- Inspector de Propiedades (Property Inspector): con esta opción se asignan y modifican las propiedades de cada objeto en forma personalizada.

- Navegador de Objetos (Object Browser): Muestra todos los objetos que se encuentran en la figura (en forma de árbol) y a través del Object Browser se pueden seleccionar los objetos.

- Editor de Menús (Menú Editor): El redactor de Menú crea menús de ventana y menús de contexto.

Con una GUI, el flujo de cómputo está controlado por las acciones en la interfaz. Mientras que en un script el flujo de comandos está predeterminado, el flujo de operaciones con una GUI no lo está. Los comandos para crear una interfaz con el usuario se escriben en un script, la interfaz invoca que se ejecute el script, mientras la interfaz del usuario permanece en la pantalla aunque no se haya completado la ejecución del script.

El programa MATLAB ofrece una serie de funciones de dibujo y visualización de datos, como es el caso de imshow, imagesc o plot, por ejemplo. Cuando llamamos a una función gráfica, MATLAB crea el gráfico requerido usando objetos como ventanas, ejes de coordenadas, líneas, texto, etc.

Cada objeto está asociado a un identificador (handle) único desde el momento de su creación. A través de este identificador podremos modificar las características (llamadas propiedades del objeto) de un objeto gráfico. Naturalmente, también podremos establecer las propiedades de un objeto en el momento de su creación (cambiarlas con respecto a los valores por omisión). El identificador del objeto raíz, la pantalla, es siempre cero. El identificador de las distintas ventanas (figure) es un entero que aparecerá en la barra de título de la ventana. Los identificadores de los demás objetos gráficos son números reales. Cualquier identificador se puede obtener como valor de retorno de una función y almacenarse en una variable.

Puede haber varias ventanas abiertas, pero sólo una de ellas es la ventana activa en cada momento. De la misma forma, una ventana puede contener varios ejes de coordenadas, pero sólo unos son los ejes activos. El objeto activo será el último objeto creado o sobre el que se haya hecho clic con el ratón.

Podemos obtener los identificadores de la ventana, los ejes y el objeto activos con las órdenes:

- gcf: devuelve un entero, el identificador de la ventana activa.
- gca: devuelve el identificador de los ejes activos.
- gco: devuelve el identificador del objeto activo.

La principal utilidad que tiene conocer los identificadores de los objetos gráficos es que a través de ellos podremos modificar las propiedades de los objetos o incluso borrarlos:

- set (id): muestra en pantalla todas las propiedades del objeto al que corresponde el identificador id.
- get (id): produce un listado de las propiedades y de sus valores.
- set (id, 'propiedad', 'valor'): establece un nuevo valor para la propiedad del objeto con identificador id. Se pueden establecer varias propiedades en la misma llamada a set incluyendo una lista de parejas 'propiedad', 'valor' en la llamada.
- get (id, 'propiedad'): obtiene el valor de la propiedad especificada.
- delete (id): borra el objeto cuyo identificador es id y todos sus hijos.

Los controles de la interfaz con el usuario en MATLAB se especifican con la orden uicontrol. Estos controles tienen mucho en común con los menús de la interfaz con el usuario, pero los primeros tienen muchos estilos. La sintaxis de uicontrol es:

```
k = uicontrol ('Style', 'especificación de estilo', …
        'String', 'cadena para exhibir', …
        'Value', [valor], …
        'BackgroundColor', [r,g,b], …
        'Max' [valor], …
        'Min' [valor], …
        'Position', [izq, base, ancho, alto], …
        'CallBack', 'cadena de invocacion')
```

donde 'especificación de estilo' es una de las siguientes cadenas:

popup
push
radio
checkbox
slider
edit (texto editable)
text (texto estático)
frame

Las propiedades de iucontrol son similares a las de uimenu. Destacamos:

a) 'Value', valor: especifica el valor por omisión de ajuste. En el caso de interruptores de encendido/apagado, valor es 0 o 1. En el caso de un control deslizante (slider), puede ser cualquier valor entre el mínimo y el máximo.

b) 'Min', Valor: establece el valor mínimo. Su significado difiere dependiendo del estilo.

c) 'Max', Valor: establece el valor máximo. Su significado difiere dependiendo del estilo.

Hay muchas más propiedades que pueden incluirse en los comandos del uicontrol, tal como sucede con las propiedades del uimenu, aunque al programar conviene minimizar el número de propiedades a fin de simplificar el script.

Mencionaremos las partes más importantes de GUIDE:

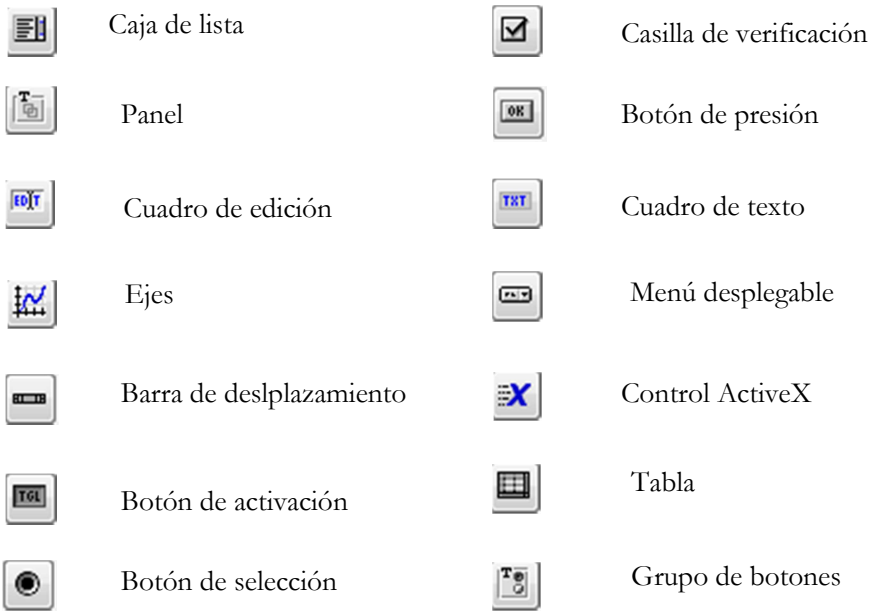


Figura 10-4. Paleta de componentes

Texto estático (cuadro de teto): Un static text puede exhibir símbolos, mensajes o incluso valores numéricos de una GUI, y puede colocarse en un lugar deseado. El texto estático no tiene cadenas de invocación. A continuación mostramos un ejemplo de texto estático.

```
k1 = uicontrol (´Style´, ´text´, …
        ´String´, ´cadena para exhibir´, …
        ´Position´, [20, 50, 140, 30])
```

El contenido de un texto exhibido puede modificarse si es necesario. Esto se hace con el comando set. Por ejemplo, si se ejecuta el comando que sigue desde la ventana de comandos mientras está vigente el ejemplo anterior de orden uicontrol:

```
set (k1, ´string´, ´Ahora aparece un texto modificado.´)
```

Menú desplegable: Los pop-up menús difieren de los menús de interfaz con el usuario en que pueden aparecer en cualquier punto de la ventana de la figura, mientras que los menús de interfaz con el usuario solo se localizan en la parte superior.

Botón de presión: Los Push button generan una acción cuando hacemos click con el puntero del ratón sobre ellos. Cuando se da click en un push button, aparece presionado; cuando se suelta el botón del ratón, el botón aparece levantado; y su rutina de llamada se ejecuta.

Casilla de verificación: Las casillas de verificación están diseñadas para realizar operaciones de encendido/apagado. Las posiciones de encendido/apagado se registran en Value que puede examinarse con get(handle, ´value´). Los comandos axis on y axis off se escriben en la cadena de invocación.

Botón de selección: Cuando solo se usa un botón de radio, no existe diferencia funcional alguna con respecto a una casilla de verificación. Por otro lado, los botones de radio en grupo son mutuamente exclusivos (es decir, si un botón esta encendido, todos los demás botones se apagan), mientras que las casillas de verificación son independientes entre sí. Sin embargo, esta característica exclusiva de los botones de radio sólo puede implementarse mediante la programación del usuario en la cadena de invocación.

Barra de desplazamiento: Los sliders aceptan datos de entrada numéricos con un rango específico. Los usuarios mueven la barra dejando presionado el botón del mouse y arrastrándola, haciendo click en la flecha. La posición de la barra indica un valor numérico.

Texto editable (cuadro de edición): El dispositivo de texto editable permite al usuario teclear una cadena de entrada. Se pueden escribir varios valores numéricos en forma de vector o matriz como cadena mediante el mismo dispositivo; esta cadena se convertirá posteriormente en valores numéricos con el comando str2num.

Un ejemplo de uicontrol para texto editable es:

```
ed1 = uicontrol (gcf, ´Style´, ´edit´, …
        ´Position´, [10, 260, 110, 20], …
        ´CallBack´, inp_txt = get (ed1, ´´string´´)´)
```

Las palabras clave en el comando anterior son ´Style´, ´edit´ y get que capturan el texto introducido.

Panel y grupo de botones: El estilo panel o grupo de botones puede servir para agrupar dispositivos como lo botones de radio o las casillas de verificación.

Botón de activación: El toggle button genera una acción que indica un estado binario (on u off). Cuando se hace click en un toggle button, aparece presionado y permanece así hasta que se suelta el botón del mouse, y en ese momento ejecuta la llamada. Un click posterior del mouse regresa al toggle button a su estado original y vuelve a ejecutar la rutina de llamada.

Cajas de lista: El componente List Box muestra una lista de artículos y permite a usuarios seleccionar unos o más artículos.

El inspector de propiedades ("Property inspector") está compuesto de las siguientes propiedades o atributos, tal y como se muestra en la figura.

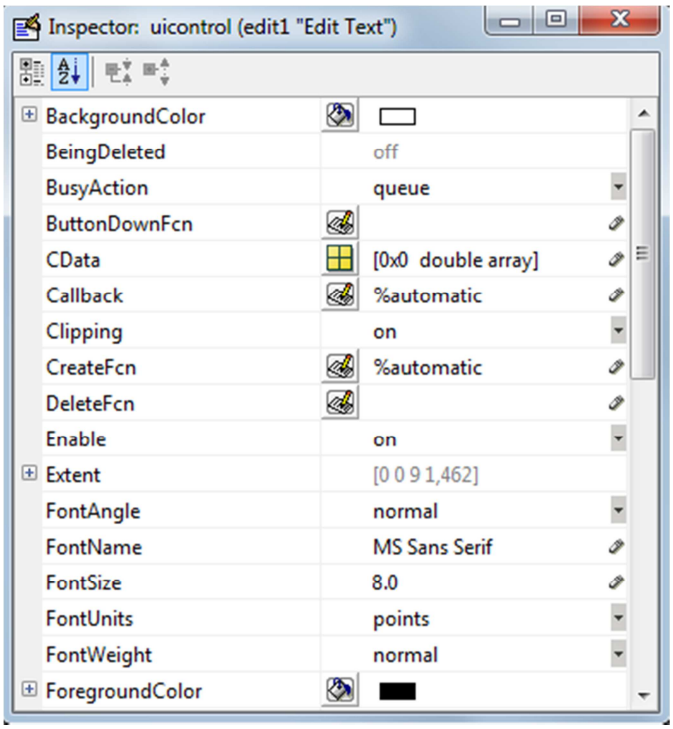


Figura 10-5. Inspector de propiedades

Cada uno de los controles tiene propiedades particulares, algunos de los cuales se analizan a continuación:

BackgroundColor: El color usado para rellenar el rectángulo de unicontrol. Especifica un color usando un vector de tres elementos RGB (rojo, verde y azul) o uno de los nombres ya predefinidos en Matlab. El color por "defecto" es determinado por la configuración del sistema.

BusyAction: Interrupción de la rutina de llamada (callback). Si una llamada es ejecutada y el usuario activa un evento en un objeto para el cual una llamada está definida, esa llamada trata de interrumpir la primera llamada. La primera llamada puede ser interrumpida solamente por uno de los siguientes comandos: drawnow, figure, getframe, pause o waitfor; si la llamada no contiene ninguno de estos comandos no puede ser interrumpida.

Si la propiedad Interruptible del objeto que está ejecutando la llamada está desactivada (off), la llamada no puede ser interrumpida (excepto por algunas llamadas).

La propiedad BusyAction del objeto cuya llamada está esperando para ejecutarse determina lo que le pasa a la llamada:

- Si el valor es queue, la llamada es agregada al evento queue y se ejecuta después de que la primera llamada termina de ejecutarse.
- Si el valor es Cancel, el evento es descartado y la llamada no se ejecuta.

Nota: Si la llamada interrumpida es una llamada de DeleteFcn o CreateFcn o una de una figura de CloseRequest o ResizeFcn, se interrumpe y ejecuta sin importar el valor de la propiedad Interrumpible del objeto.

ButtonDownFcn: Una rutina de llamada que se ejecuta cuando se presiona un botón del mouse mientras el cursor está en un uicontrol. Cuando la propiedad enable del unicontrol está desactivada, el ButtonDownFcn se ejecuta al hacer click en el unicontrol. Esto es útil para implementar acciones, para modificar interactivamente las propiedades de control del objeto, como el tamaño y la posición.

Esta rutina se define como una cadena (string) que es una expresión válida en Matlab o el nombre de un archivo M (M-file). La expresión se ejecuta en el espacio de trabajo de Matlab.

La propiedad de llamada define la rutina de llamada que se ejecuta cuando se hace clic en el botón.

Callback: Controla la acción. Una rutina que se ejecuta cuando se activa un objeto de la clase uicontrol. Define esta rutina como una cadena. La expresión se ejecuta en el espacio de trabajo de Matlab.

Para ejecutar la rutina para un control de texto editable, escribe el texto deseado y después sigue uno de los siguientes pasos:

- Mueve la selección del objeto (da clic en cualquier otra parte)
- Para un texto editable de una sola línea, presiona Return
- Para una caja de texto (text box), presiona Ctrl-Return.

Esta rutina definida para los componentes frame y ststic text no se ejecuta porque ninguna acción está asociada con estos objetos.

Cdata: Imagen de color verdadero mostrada en un control. Una matriz tridimensional de valores RGB que definen una imagen de color verdadero que es mostrada ya sea en un push button o un toggle button. Cada valor debe tener un rango entre cero y uno.

CreateFcn: Rutina de llamada ejecutada cuando se crea un objeto. Esta propiedad define una rutina de llamada que es ejecutada cuando Matlab crea un objeto de la clase uicontrol. Se debe definir esta propiedad como un valor por defecto para los uicontrols.

DeleteFcn: Una rutina de llamada que se ejecuta cuando se borra un objeto uicontrol. Matlab ejecuta la rutina antes de destruir las propiedades del objeto, así sus valores están disponibles para la rutina de llamada.

Enable: Activa o desactiva el uicontrol. Esta propiedad controla cómo los uicontrols responden a un clic del mouse, incluyendo qué rutina de llamada se ejecuta.

- on - El uicontrol es operacional
- inactive - no es operacional pero se ve como si estuviera activado
- off - No es operacional y su etiqueta se vuelve gris

Cuando se hace clic con el botón izquierdo del mouse sobre un objeto uicontrol que tiene su propiedad enable activada (en on), Matlab ejecuta las siguientes acciones:

- Asigna la propiedad de la figura SelectionType
- Ejecuta la rutina de llamada del control.
- No asigna la propiedad de la figura CurrentPoint y tampoco ejecuta ni la propiedad de control ButtonDownFcn ni la rutina de llamada de la figura WindowButtonDownFcn

Cuando se hace clic con el botón izquierdo del mouse sobre un objeto uicontrol que tiene su propiedad enable inactiva, o cuando se hace clic derecho en uno en el que Enable tiene cualquier valor, Matlab ejecuta estas acciones en este orden:

- Asigna la propiedad de la figura SelectionType.
- Asigna la propiedad de la figura CurrentPoint.
- Ejecuta la rutina de llamada WindowButtonDownFcn de la figura.

- En un click derecho, si el uicontrol esta asociado con un context menu, registra el context menu.
- Ejecuta la llamada ButtonDownFcn del control.
- Ejecuta la llamada del elemento seleccionado del context menu.
- No ejecuta la rutina de llamada del control.
- Poniendo esta propiedad inactiva te capacita para implementar arrastre o cambio de tamaño de objetos usando la rutina de llamada ButtonDownFcn.

Extent: Tamaño de un carácter cadena uicontrol. Un vector de cuatro elementos que define el tamaño y la posición de un carácter de tipo cadena usado para etiquetar el uicontrol.

Tiene la forma:

[0, 0, width, height]

Los dos primeros elementos siempre son cero. width (ancho) y height (alto) son las dimenciones del rectángulo. Todas las medidas son unidades especificadas por la propiedad Units.

Ya que la propiedad Extent está definida en las mismas unidades que el uicontrol mismo, se puede usar esta propiedad para determinar el tamaño adecuado del ancho del uicontrol con respecto a su etiqueta. Haciendo lo siguiente:

- Definiendo la propiedad String y seleccionando la fuente usando las propiedades relevantes.
- Tomando el valor de la propiedad Extend
- Definiendo width y height de la propiedad Position (posición) propiamente a ser de alguna manera más grandes que width y height de Extend.

FontAngle: Inclinación de un carácter. Poniendo esta propiedad en Italic (italica) u oblique (oblicua) selecciona una versión inclinada de la fuente, cuando está disponible en el sistema.

FontName: El nombre de la fuente que mostrará la cadena. Para mostrar e imprimir correctamente, debe ser un tipo de fuente que el sistema soporte.

Para usar un ancho ajustado que se vea bien en cualquier exterior, se debe poner al FontName la cadena FixedWidth (esta cadena es sensible a las mayúsculas):

Set (uicontrol_handle, 'FontName', 'FixedWidth')

FontSize: Tamaño de la fuente. Un número que especifica el tamaño de la fuente que va a ser mostrado en la cadena, en unidades determinadas por la propiedad FontUnits. El tamaño por defecto es dependiente del sistema.

FontUnits: Unidades del tamaño de la fuente. Esta propiedad determina las unidades usadas por la propiedad FontSize. Las unidades normalizadas interpretan el FontSize como una fracción de la altura del uicontrol. Cuando se cambia el tamaño del uicontrol, Matlab modifica la pantalla FontSize. pixels (pixeles), inches (pulgadas), centimeters (centimetros) y points (puntos) son unidades absolutas (1 punto = 1/72 pulgada).

FontWeight: Peso de un carácter. Poniendo esta propiedad en Bold hace que Matlab use una versión "negrita" de la fuente, cuando está disponible en el sistema.

ForegroundColor: Color de texto. Esta propiedad determina el color del texto definido por la propiedad String. Especifica un color usando un vector de tres elementos RGB o un nombre predefinido en Matlab.

ANEXO:
Planificación de desarrollo

En la planificación de desarrollo del software distinguimos 3 partes fundamentales:

- Revisión bibliográfica
- Trabajos de campo
- Trabajos de gabinete

Como sabemos en este proyecto el trabajo de campo solo consiste en la captura de fotografías y el tiempo empleado es muy corto a no ser que queramos realizar una toma de fotografías aéreas en las que necesitamos utilizar un vehículo aéreo no tripulado (UAV) o realizar un vuelo implicando su proyecto de vuelo.

Una parte importante del proyecto es la que denominamos revisión bibliográfica, pues son muchos los trabajos similares que se han realizado en estas últimas décadas y aun es un tema en desarrollo.

En la revisión bibliográfica recurrimos a manuales de fotogrametría de distintas universidades, a distintas publicaciones de artículos que encontramos en internet y material diverso procedente de otros proyectos finales de carrera, trabajos fin de master y tesis doctorales.

El proyecto que se ha realizado ha sido llevado a cabo por un estudiante de Ingeniería Técnica en Topografía en su proyecto final de carrera.

Sin embargo, comprobamos que en la web se encuentra información desarrollada no solamente por los profesionales de la Topografía sino también de las Telecomunicaciones.

Al buscar información nos encontramos que esta es abundante y a la vez compleja, siendo esta en algunos casos difícil de emplear para lo que se pretende y la categoría de Ingeniería Técnica en Topogría que se pretende conseguir.

A la vez que se busca información se idean nuevas formas de llevar a cabo el proyecto con el fin de que las tareas sean más rápidas y el resultado siga siendo bueno.

La revisión bibliográfica esta presente en toda la fase del desarrollo del proyecto desde que se empieza hasta que se termina.

Centrandonos ahora en el trabajo de gabinete, este consta de la tarea propiamente dicha de programar e implementar los algoritmos deseados así como realizar distinto software fotogramétrico para comparar y realizar las calibraciones deseadas.

Para llevar a cabo la planificación de tareas se pretende desarrollar diagramas de GANNT y de PERT siendo necesario una subclasificación de las tareas anteriores:

- Revisión bibliográfica (durante todo el proyecto)
- Trabajo de campo
  - Toma de fotografías con el fin de calibrar sobre una plantilla modelo
  - Toma de fotografías para utilizar en el proyecto con el uso de "CHISME"
- Trabajo de gabinete
  - Calibración de la cámara utilizada con Photomodeler
  - Uso de Orthoengine PCI Geomatica para obtener un primer resultado
  - Implementación de la orientación interna
  - Implementación de captura de puntos en pantalla
  - Implementación de la orientación externa
  - Implementación del algoritmo de creación de MDE
    - Pruebas
    - Implementación
    - Ejecución
    - Depuración y mejoras
  - Implementación de algoritmo para generar ortofoto
  - Creación de una interfaz grafica
  - Creación de herramientas auxiliares para la interfaz (líneas epipolares y verticales) y perfil longitudinal
  - Comprobación de resultados y depuración de programas

Con el conjunto de tareas se muestra a continuación los diagramas mencionados realizados con el software Microsoft Project:

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**PROYECTO**	**89 días**	**mar 20/03/12**	**vie 20/07/12**	
2	**REVISIÓN BIBLIOGRÁFICA**	**89 días**	**mar 20/03/12**	**vie 20/07/12**	
3	Revisión bibliográfica	89 días	mar 20/03/12	vie 20/07/12	
4	**TRABAJO DE CAMPO**	**3 días**	**lun 26/03/12**	**mié 28/03/12**	
5	**Trabajo de campo**	**3 días**	**lun 26/03/12**	**mié 28/03/12**	
6	Toma de fotografías con el fin de calibrar sobre una plantilla modelo	1 día	lun 26/03/12	lun 26/03/12	
7	Toma de fotografías para utilizar en el proyecto con el uso de "CHISME"	2 días	mar 27/03/12	mié 28/03/12	
8	**TRABAJO DE GABINETE**	**83 días**	**jue 29/03/12**	**lun 23/07/12**	**7**
9	**Trabajo de gabinete**	**83 días**	**jue 29/03/12**	**lun 23/07/12**	
10	Calibración de la cámara utilizada con Photomodeler	2 días	jue 29/03/12	vie 30/03/12	7
11	Uso de Orthoengine PCI Geomatica para obtener un primer resultado	3 días	lun 02/04/12	mié 04/04/12	10
12	Implementación de la Orientación Interna	1 día	jue 05/04/12	jue 05/04/12	11
13	Implementación de captura de puntos en pantalla	1 día	vie 06/04/12	vie 06/04/12	12
14	Implementación de la Orientación Externa	3 días	lun 09/04/12	mié 11/04/12	13
15	**Implementación del algoritmo de creación de MDE**	**40 días**	**jue 12/04/12**	**mié 06/06/12**	**14**
16	Pruebas	5 días	jue 12/04/12	mié 18/04/12	14
17	Implementación	8 días	jue 19/04/12	lun 30/04/12	16
18	Ejecución	25 días	mar 01/05/12	lun 04/06/12	17
19	Depuración y mejoras	2 días	mar 05/06/12	mié 06/06/12	18
20	Implementación de algoritmo para generar ortofoto	6 días	jue 07/06/12	jue 14/06/12	19
21	Creación de una interfaz gráfica	15 días	vie 15/06/12	jue 05/07/12	20
22	Creación de herramientas auxiliares para la interfaz (lineas epipolares y verticales) y función de perfil longitudinal	5 días	vie 06/07/12	jue 12/07/12	21
23	Comprobación de resultados y depuración de programas	6 días	vie 13/07/12	vie 20/07/12	22

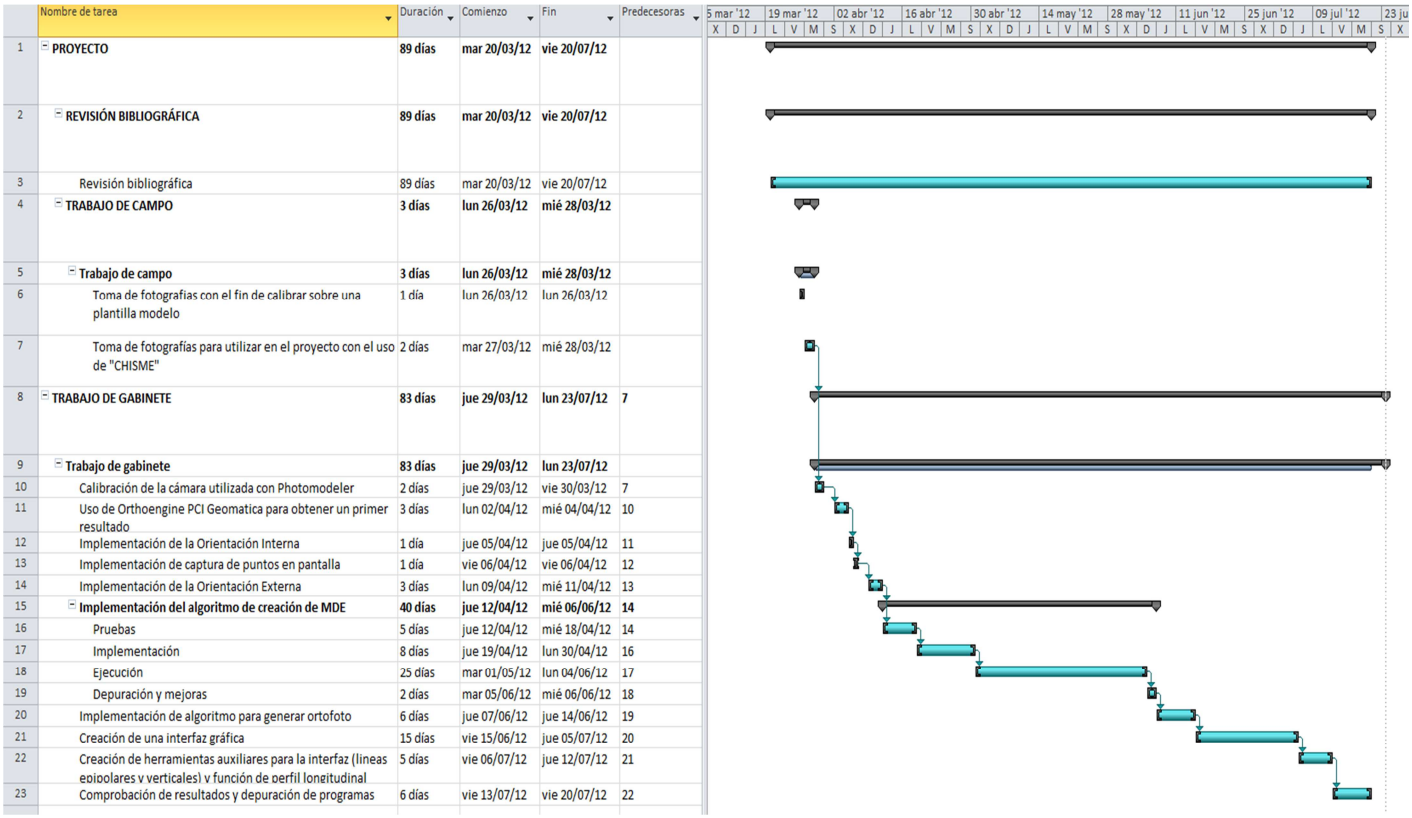


Figura 11-1. Diagrama de GANTT

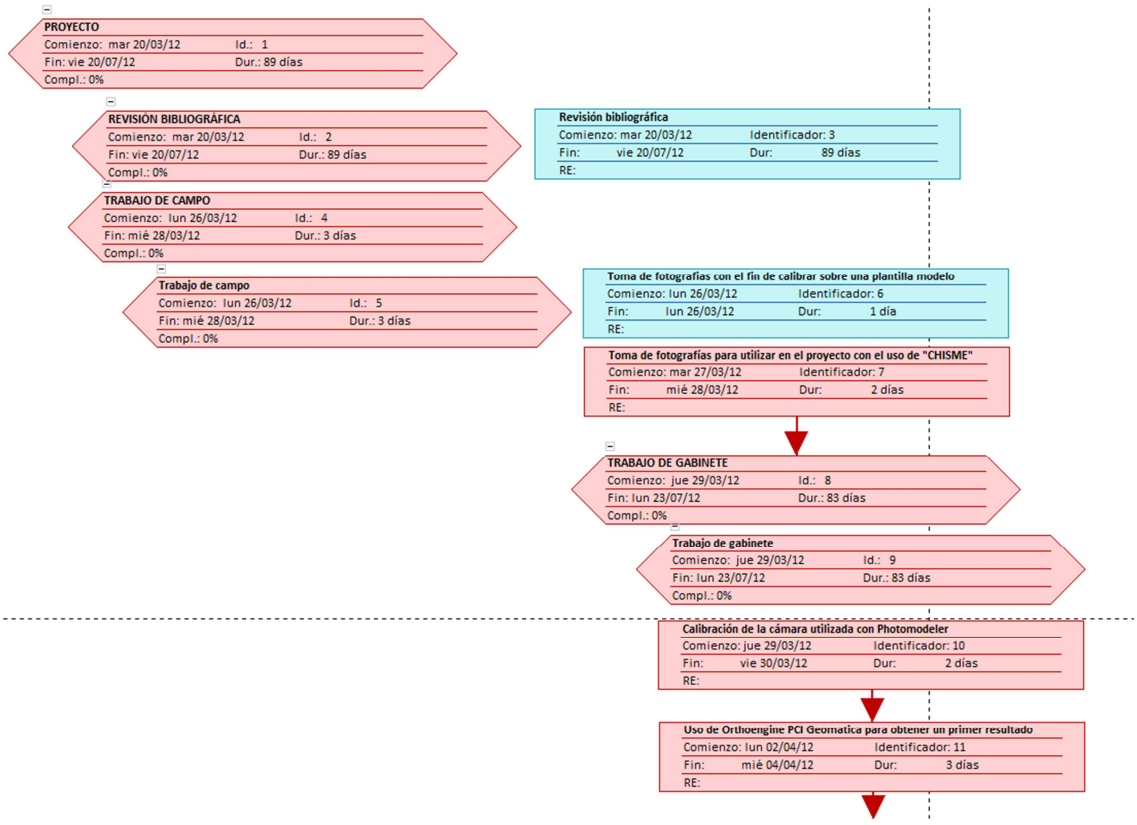


Figura 11-2. Diagrama de PERT parte 1

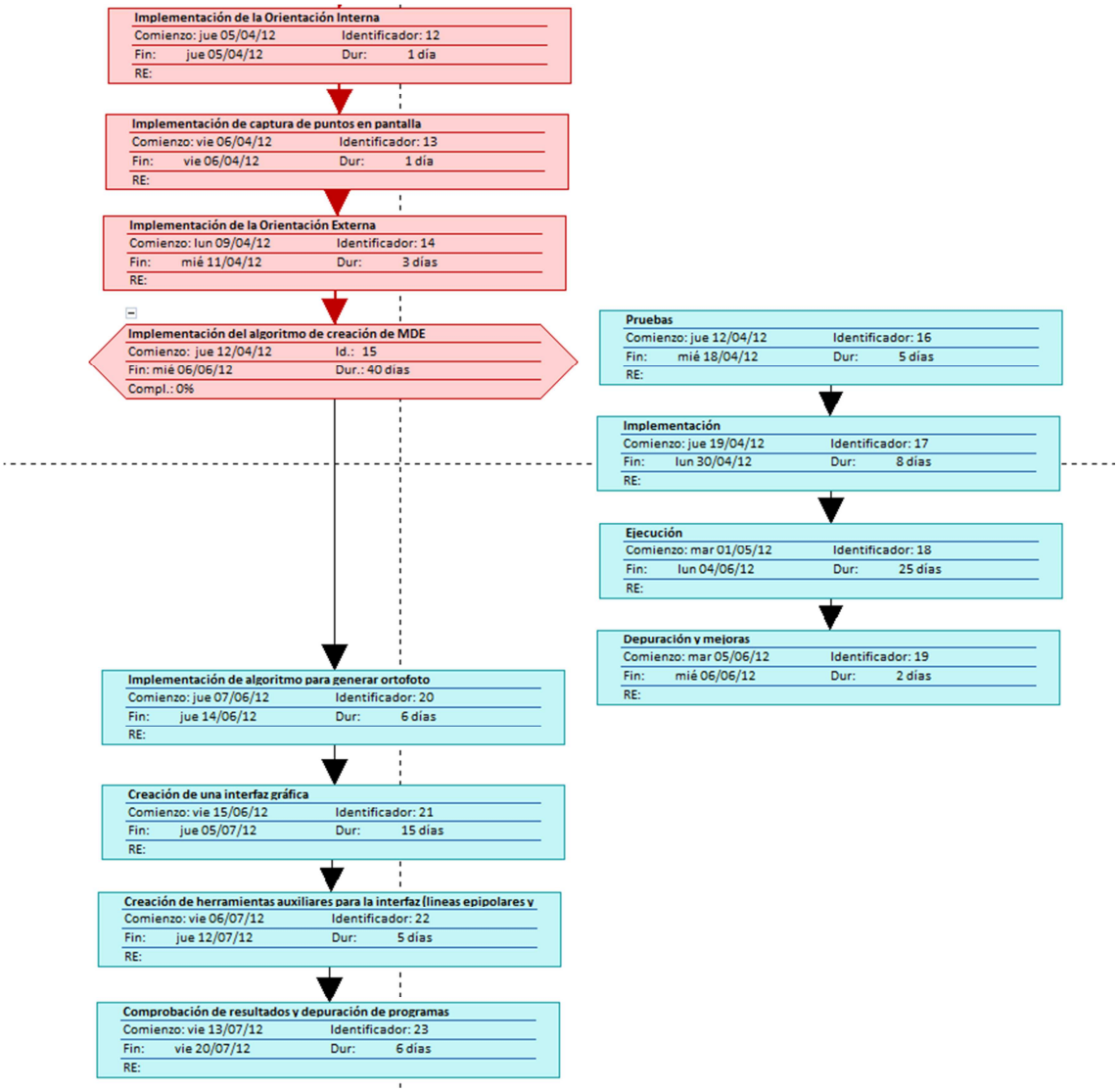


Figura 11-3. Diagrama de PERT parte 2

ANEXO: Justificación de precios y Presuspuesto

Antes de elaborar el cuadro de precios vamos a determinar el número de días útiles trabajados:

Días del año 2012	366
Días que no se trabajan	
Fiestas	14
Domingos	52
Sábados	52
Total	11

366	
-118	
248	Total días hábiles
-15	6% de pérdidas sobre 248
233	Total
-20	Vacaciones
213	Total días útiles

	Titulado Medio
Retribuciones brutas anuales (14 pagas) del convenio	22.157,87€
Cotizaciones empresa 38%	8.419,99€
Total coste anual	30.577,86€
Coste diario 213 días	143,55€ ≈ 144€

El siguiente paso es la justificación de precios correspondientes a gastos de maquinaria:

Los días de uso en la justificación son 3 para trabajos de campo y 86 para los de gabinete

Gastos de maquinaria y otros	Equipo de topografía	Valor residual 33%	Diferencia	Amortización 20% anual	Días usados	Coste diario (€/día)
Manuales y bibliografía	50,00 €	15,00 €	35,00 €	7,00 €	89	0,08 €
Cámara Panasonic GS-200	300,00 €	90,00 €	210,00 €	42,00 €	3	14,00 €
Marco "CHISME"	100,00 €	30,00 €	70,00 €	14,00 €	3	4,67 €
Equipo Informatico	1.000,00 €	300,00 €	700,00 €	140,00 €	89	1,57 €
Software Photomodeler	3.000,00 €	900,00 €	2.100,00 €	420,00 €	2	210,00 €
Software Orthoengine PCI Geomatis	3.000,00 €	900,00 €	2.100,00 €	420,00 €	3	140,00 €
Software Matlab	6.000,00 €	1.800,00 €	4.200,00 €	840,00 €	78	10,77 €
Total	13.450,00 €	4.035,00 €	9.415,00 €	1.883,00 €		381,09 €

Cuadro de precios número 1

Nº Actividad	Código	Descripción unidad de obra	Precio	
1	1	**REVISIÓN BIBLIOGRÁFICA**		
01 01	1.1	Revisión bibliográfica	14,60 €/revisión	CATORCE CON SESENTA EUROS
2	2	**TRABAJO DE CAMPO**		
02 01	2.1	Toma de fotografías con el fin de calibrar sobre un plantilla modelo (10 fotos)	15,96 €/foto	QUINCE CON NOVENTA Y SEIS EUROS
02 02	2.2	Toma de fotografías para utilizar en el proyecto con el uso de "CHISME" (16 fotos)	21,11 €/foto	VEINTIUNO CON ONCE EUROS
3	3	**TRABAJO DE GABINETE**		
03 01	3.1	Calibración de la cámara utilizada con Photomodeler (10 fotos)	71,11 €/foto	SETENTA Y UNO CON ONCE EUROS
03 02	3.2	Uso de Orthoengine PCI Geomatica para obtener un primer resultado (16 fotos)	53,54 €/foto	CINCUENTA Y TRES CON CINCUENTA Y CUATRO EUROS
03 03	3.3	Implementación de la orientación interna (1 función)	156,34 €/función	CIENTOCIENCUENTA Y SEIS CON TREINTA Y CUATRO EUROS
03 04	3.4	Implementación de captura de puntos en pantalla (1 función)	156,34 €/función	CIENTOCIENCUENTA Y SEIS CON TREINTA Y CUATRO EUROS
03 05	3.5	Implementación de la orientación externa (1 función)	469,02 €/función	CUATROCIENTOS SESENTA Y NUEVE CON CERO DOS EUROS
03 06	3.6	Implementación del algoritmo de creación del MDE (pruebas, implementación, ejecución, depuración y mejoras, 1 función)	6.253,60 €/función	SEISMIL DOSCIENTAS CINCUENTA Y TRES CON SESENTA EUROS
03 07	3.7	Implementación de algoritmo para generar ortofoto (1 función)	938,04 €/función	NUEVECIENTAS TREINTA Y OCHO CON CERO CUATRO EUROS
03 08	3.8	Creación de interfaz gráfica (7 interfaces)	335,01 €/función	TRESCIENTRAS TREINTA Y CINCO CON CERO UN EUROS
03 09	3.9	Creación de herramientas auxiliares para la interfaz (líneas epipolares y verticales y función del perfil longitudinal)	260,57 €/función	DOSCIENTAS SESENTA CON CINCUENTA Y SIETE EUROS
03 10	3.10	Comprobación de resultados y depuración de programas (1 comprobación)	938,04 €/función	NOVECIENTAS TREINTA Y OCHO CON CERO CUATRO EUROS

Cuadro de precios número 2

Nº Actividad	Código	Descripción unidad de obra	Precio (€/día)	día/unidad	€/unidad	Cantidad	Total
1	1	**REVISIÓN BIBLIOGRAFICA**					
01 01	1.1	**Revisión Bibliográfica (89 revisiones en total, 10% de jornada)**					
		RECURSOS HUMANOS					
01 01 01	1.1.1	Ingeniero técnico en topografía	144,00 €	0,1	25,63 €	1	14,40 €
		CATORCE CON CUARENTA EUROS					
		RECURSOS MATERIALES					
01 01 02	1.1.2	Equipo informático	1,57 €	0,1	0,28 €	1	0,16 €
		CERO CON DIECISEIS EUROS					
01 01 03	1.1.3	Manuales y bibliografía	0,08 €	0,1	0,01 €	5	0,04 €
		CERO CON CERO CUATRO EUROS					
		CATORCE CON SESENTA EUROS				**Suma**	**14,60 €/revisión**

Nº Actividad	Código	Descripción unidad de obra	Precio (€/día)	día/unidad	€/unidad	Cantidad	Total
2	2	**TRABAJO DE CAMPO**					
02 01	2.1	**Toma de fotografías con el fin de calibrar sobre un plantilla modelo (10 fotos)**					
		RECURSOS HUMANOS					
02 01 01	2.1.1	Ingeniero técnico en topografía	144,00 €	0,10000	14,40 €	1	14,40 €
		CIENTO CUARENTA Y CUATRO EUROS					
		CIENTO UN EURO					
		RECURSOS MATERIALES					
02 01 02	2.1.2	Cámara Panasonic GS-200	14,00 €	0,10000	1,40 €	1	1,40 €
		UNO CON CUARENTA EUROS					
02 01 03	2.1.3	Equipo informático	1,57 €	0,10000	0,16 €	1	0,16 €
		CERO CON DIECISEIS EUROS					
		QUINCE CON NOVENTA Y SEIS EUROS				**Suma**	**15,96 €/foto**

02 02	2.2	Toma de fotografías para utilizar en el proyecto con el uso de "CHISME" (16 fotos)					
		RECURSOS HUMANOS					
02 02 01	2.2.1	Ingeniero técnico en topografía	144,00 €	0,125	18,00 €	1	18,00 €
		DIECIOCHO EUROS					
		RECURSOS MATERIALES					
02 02 02	2.2.2	Cámara Panasonic GS-200	14,00 €	0,125	1,75 €	1	1,75 €
		UNO CON SETENTA Y CINCO EUROS					
02 02 03	2.2.3	Equipo informático	1,57 €	0,125	0,20 €	1	0,20 €
		CERO CON VEINTE EUROS					
02 02 04	2.2.4	Marco "CHISME"	4,67 €	0,125	0,58 €	2	1,17 €
		UNO CON DIECISIESTE EUROS					
		VEINTIUNO CON ONCE EUROS				Suma	21,11 €/foto

3	3	TRABAJO DE GABINETE					
03 01	3.1	Calibración de la cámara utilizada con Photomodeler (10 fotos)					
		RECURSOS HUMANOS					
03 01 01	3.1.1	Ingeniero técnico en topografía	144,00 €	0,2	28,80 €	1	28,80 €
		CIENTO CUARENTA Y CUATRO EUROS					
		RECURSOS MATERIALES					
03 01 02	3.1.2	Equipo informático	1,57 €	0,2	0,31 €	1	0,31 €
		CERO CON TREINTA Y UN EUROS					
03 01 03	3.1.3	Software Photomodeler	210,00 €	0,2	42,00 €	1	42,00 €
		CUARENTA Y DOS EUROS					
		SETENTA Y UNO CON ONCE EUROS				Suma	71,11 €/foto

03 02	3.2	Uso de Orthoengine PCI Geomatica para obtener un primer resultado (16 fotos)					
		RECURSOS HUMANOS					
03 02 01	3.2.1	Ingeniero técnico en topografía	144,00 €	0,1875	27,00 €	1	27,00 €
		VEINTISIETE EUROS					
		RECURSOS MATERIALES					
03 02 02	3.2.2	Equipo informático	1,57 €	0,1875	0,29 €	1	0,29 €
		CERO CON VEINTINUEVE EUROS					
03 02 03	3.2.3	Software Orthoengine PCI Geomatic	140,00 €	0,1875	26,25 €	1	26,25 €
		VEINTISEIS CON VEINTICINCO EURS					
		CINCUENTA Y TRES CON CINCUENTA Y CUATRO EUROS				Suma	53,54 €/foto

03 03	3.3	**Implementación de la orientación interna (1 función)**					
		RECURSOS HUMANOS					
03 03 01	3.3.1	Ingeniero técnico en topografía	144,00 €	1	144,00 €	1	144,00 €
		CIENTO CUARENTA Y CUATRO EUROS					
		RECURSOS MATERIALES					
03 03 02	3.3.2	Equipo informático	1,57 €	1	1,57 €	1	1,57 €
		UNO CON CINCUENTA Y SIETE EUROS					
03 03 03	3.3.3	Software Matlab	10,77 €	1	10,77 €	1	10,77 €
		DIEZ CON SETENTA Y SIETE					
		CIENTOCIENCUENTA Y SEIS CON TREINTA Y CUATRO EUROS				**Suma**	**156,34 €/función**

03 04	3.4	**Implementación de captura de puntos en pantalla (1 función)**					
		RECURSOS HUMANOS					
03 04 01	3.4.1	Ingeniero técnico en topografía	144,00 €	1	144,00 €	1	144,00 €
		CIENTO CUARENTA Y CUATRO EUROS					
		RECURSOS MATERIALES					
03 04 02	3.4.2	Equipo informático	1,57 €	1	1,57 €	1	1,57 €
		UNO CON CINCUENTA Y SIETE EUROS					
03 04 03	3.4.3	Software Matlab	10,77 €	1	10,77 €	1	10,77 €
		DIEZ CON SETENTA Y SIETE					
		CIENTOCIENCUENTA Y SEIS CON TREINTA Y CUATRO EUROS				**Suma**	**156,34 €/función**

03 05	3.25	**Implementación de la orientación externa (1 función)**					
		RECURSOS HUMANOS					
03 05 01	3.5.1	Ingeniero técnico en topografía	144,00 €	3	432,00 €	1	432,00 €
		CUATROCIENTOS TREINTA Y DOS EUROS					
		RECURSOS MATERIALES					
03 05 02	3.5.2	Equipo informático	1,57 €	3	4,71 €	1	4,71 €
		CUATRO CON SETENTA Y UN EURO					
03 05 03	3.5.3	Software Matlab	10,77 €	3	32,31 €	1	32,31 €
		TREINTA Y DOS CON TREINTA Y UN EURO					
		CUATROCIENTOS SESENTA Y NUEVE CON CERO DOS EUROS				**Suma**	**469,02 €/función**

03 06	3.6	Implementación del algoritmo de creación del MDE (pruebas, implementación, ejecución, depuración y mejoras, 1 función)					
		RECURSOS HUMANOS					
03 06 01	3.6.1	Ingeniero técnico en topografía	144,00 €	40	5.760,00 €	1	5.760,00 €
		CINCOMIL SETECIENTAS SESENTA EUROS					
		RECURSOS MATERIALES					
03 06 02	3.6.2	Equipo informático	1,57 €	40	62,80 €	1	62,80 €
		SESENTA Y DOS CON OCHENTA EUROS					
03 06 03	3.6.3	Software Matlab	10,77 €	40	430,80 €	1	430,80 €
		CUATROCIENTOS TREINTA CON OCHENTA EUROS					
		SEISMIL DOSCIENTAS CINCUENTA Y TRES CON SESENTA EUROS				Suma	**6.253,60 €/función**

03 07	3.7	Implementación de algoritmo para generar ortofoto (1 función)					
		RECURSOS HUMANOS					
03 07 01	3.7.1	Ingeniero técnico en topografía	144,00 €	6	864,00 €	1	864,00 €
		OCHOCIENTAS SESENTA Y CUATRO EUROS					
		RECURSOS MATERIALES					
03 07 02	3.7.2	Equipo informático	1,57 €	6	9,42 €	1	9,42 €
		NUEVE CON CUARENTA Y DOS EUROS					
03 07 03	3.7.3	Software Matlab	10,77 €	6	64,62 €	1	64,62 €
		SESENTA Y CUATRO CON SESENTA Y DOS EUROS					
		NUEVECIENTAS TREINTA Y OCHO CON CERO CUATRO EUROS				Suma	**938,04 €/función**

03 08	3.8	Creación de interfaz gráfica (7 interfaces)					
		RECURSOS HUMANOS					
03 08 01	3.8.1	Ingeniero técnico en topografía	144,00 €	2,14	308,57 €	1	308,57 €
		TRESCIENTAS OCHO CON CINCUENTA Y SIETE EUROS					
		RECURSOS MATERIALES					
03 08 02	3.8.2	Equipo informático	1,57 €	2,14	3,36 €	1	3,36 €
		TRES CON TREINTA Y SEIS EUROS					
03 08 03	3.8.3	Software Matlab	10,77 €	2,14	23,08 €	1	23,08 €
		VEINTITRES CON CERO OCHO EUROS					
		TRESCIENTRAS TREINTA Y CINCO CON CERO UN EUROS				Suma	**335,01 €/interfaz**

03 09	3.9	Creación de herramientas auxiliares para la interfaz (líneas epipolares y verticales y función del perfil longitudinal)					
		RECURSOS HUMANOS					
03 09 01	3.9.1	Ingeniero técnico en topografía	144,00 €	1,67	240,00 €	1	240,00 €
		DOSCIENTOS CUARENTA EUROS					
		RECURSOS MATERIALES					
03 09 02	3.9.2	Equipo informático	1,57 €	1,67	2,62 €	1	2,62 €
		DOS CON SESENTA Y DOS EUROS					
03 09 03	3.9.3	Software Matlab	10,77 €	1,67	17,95 €	1	17,95 €
		DIECISIETE CON NOVENTA Y CINCO EUROS					
		DOSCIENTAS SESENTA CON CINCUENTA Y SIETE EUROS				Suma	**260,57 €/ herramienta**

03 10	3.10	Comprobación de resultados y depuración de programas (1 comprobación)					
		RECURSOS HUMANOS					
03 10 01	3.10.1	Ingeniero técnico en topografía	144,00 €	6,00	864,00 €	1	864,00 €
		OCHOCIENTAS SESENTA Y CUATRO EUROS					
		RECURSOS MATERIALES					
03 10 02	3.10.2	Equipo informático	1,57 €	6,00	9,42 €	1	9,42 €
		NUEVE CON CUARENTA Y DOS EUROS					
03 10 03	3.10.3	Software Matlab	10,77 €	6,00	64,62 €	1	64,62 €
		SESENTA Y CUATRO CON SESENTA Y DOS EUROS					
		NOVECIENTAS TREINTA Y OCHO CON CERO CUATRO EUROS				Suma	**938,04€/ comprobación**

Los rendimientos en un proyecto como el que se presenta son difíciles de calcular ya que no siempre se rinde por igual a la hora de programar. Al programar y pensar en como desarrollar un determinado algoritmo tenemos que buscar información previa en bibliografía, y sobre esta, idear nuevas técnicas.

Tomando distintas unidades para cada tarea podemos hacer una tabla de rendimientos con el fin de elaborar el presupuesto de este proyecto.

En la tabla podemos ver las unidades de cada tarea, las jornadas empleadas en cada tarea, la cantidad empleada por unidad y el rendimiento definido como:

$$Rendimiento = \frac{Jornada}{Cantidad(unidades)}$$

TAREA	Unidad	Jornadas	Cantidad	Rendimiento (jornadas/cantidad)
Revisión Bibliográfica (89 revisiones en total, 10% de jornada)	revisión	89	89	0,1000
Toma de fotografías con el fin de calibrar sobre un plantilla modelo (10 fotos)	foto	1	10	0,1000
Toma de fotografías para utilizar en el proyecto con el uso de "CHISME" (16 fotos)	foto	2	16	0,1250
Calibración de la cámara utilizada con Photomodeler (10 fotos)	foto	2	10	0,2000
Uso de Orthoengine PCI Geomatica para obtener un primer resultado (16 fotos)	foto	3	16	0,1875
Implementación de la orientación interna (1 función)	función	1	1	1,0000
Implementación de captura de puntos en pantalla (1 función)	función	1	1	1,0000
Implementación de la orientación externa (1 función)	función	3	1	3,0000
Implementación del algoritmo de creación del MDE (pruebas, implementación, ejecución, depuración y mejoras, 1 función)	función	40	1	40,0000
Implementación de algoritmo para generar ortofoto (1 función)	función	6	1	6,0000
Creación de interfaz gráfica (7 interfaces)	interfaz	15	7	2,1429
Creación de herramientas auxiliares para la interfaz (líneas epipolares y verticales y función del perfil longitudinal)	herramientas	5	1	5,0000
Comprobación de resultados y depuración de programas (1 comprobación)	comprobación	6	1	6,0000

Con lo anterior expuesto realizamos la siguiente tabla de mediciones:

Nº Actividad	Código	Descripción unidad de obra	Medición
1	1	**REVISIÓN BIBLIOGRÁFICA**	
01 01	1.1	Revisión bibliográfica	89 revisiones
2	2	**TRABAJO DE CAMPO**	
02 01	2.1	Toma de fotografías con el fin de calibrar sobre un plantilla modelo (10 fotos)	10 fotos
02 02	2.2	Toma de fotografías para utilizar en el proyecto con el uso de "CHISME" (16 fotos)	16 fotos
3	3	**TRABAJO DE GABINETE**	
03 01	3.1	Calibración de la cámara utilizada con Photomodeler (10 fotos)	10 fotos
03 02	3.2	Uso de Orthoengine PCI Geomatica para obtener un primer resultado (16 fotos)	16 fotos
03 03	3.3	Implementación de la orientación interna (1 función)	1 función
03 04	3.4	Implementación de captura de puntos en pantalla (1 función)	1 función
03 05	3.5	Implementación de la orientación externa (1 función)	1 función
03 06	3.6	Implementación del algoritmo de creación del MDE (pruebas, implementación, ejecución, depuración y mejoras, 1 función)	1 función
03 07	3.7	Implementación de algoritmo para generar ortofoto (1 función)	1 función
03 08	3.8	Creación de interfaz gráfica (7 interfaces)	7 interfaces
03 09	3.9	Creación de herramientas auxiliares para la interfaz (líneas epipolares y verticales y función del perfil longitudinal)	3 herramientas
03 10	3.10	Comprobación de resultados y depuración de programas (1 comprobación)	1 comprobación

Finalmente obtenemos el siguiente presupuesto:

Descripción unidad de obra	Cantidad	Coste (€)	Coste total
REVISIÓN BIBLIOGRÁFICA			
Revisión bibliográfica	89	14,60 €	1299.40 €
TRABAJO DE CAMPO			
Toma de fotografías con el fin de calibrar sobre un plantilla modelo (10 fotos)	10	15,96 €	159,57 €
Toma de fotografías para utilizar en el proyecto con el uso de "CHISME" (16 fotos)	16	21,11 €	337,82 €
TRABAJO DE GABINETE			
Calibración de la cámara utilizada con Photomodeler (10 fotos)	10	71,11 €	711,14 €
Uso de Orthoengine PCI Geomatica para obtener un primer resultado (16 fotos)	16	53,54 €	856,71 €
Implementación de la orientación interna (1 función)	1	156,34 €	156,34 €
Implementación de captura de puntos en pantalla (1 función)	1	156,34 €	156,34 €
Implementación de la orientación externa (1 función)	1	469,02 €	469,02 €
Implementación del algoritmo de creación del MDE (pruebas, implementación, ejecución, depuración y mejoras, 1 función)	1	6.253,60 €	6.253,60 €
Implementación de algoritmo para generar ortofoto (1 función)	1	938,04 €	938,04 €
Creación de interfaz gráfica (7 interfaces)	7	335,01 €	2.345,10 €
Creación de herramientas auxiliares para la interfaz (líneas epipolares y verticales y función del perfil longitudinal)	3	260,57 €	781,70 €
Comprobación de resultados y depuración de programas (1 comprobación)	1	938,04 €	938,04 €
PRESUPUESTO DE EJECUCION MATERIAL (PEM)			16.415,88 €
		Beneficio Industrial (6%)	924,17 €
		Gastos Generales (13%)	2.002,37 €
			18.329.36 €
		IVA (21%)	3.849.16 €
PRESUPUESTO DE EJECUCIÓN POR CONTRATA (PEC)		TOTAL	**22.178,52 €**

Veintidós mil ciento setenta y ocho con cincuenta y dos euros

ANEXO:
DVD del Proyecto en formato digital

Bibliografía

Felicísimo, Ángel M, 1994, "Modelos digitales del terreno: introducción y aplicaciones en las ciencias ambientales", Oviedo

García de Jalón, J, Rodríguez, J.I Vidal, Jesús, 2005, "Aprenda Matlab 7.0 como si estuviera en primero", Universidad Politécnica de Madrid

Gonzalez, R.C, 2004, "Digital image using Matlab Processing", ISBN 0-13-008519-7

Kraus, K, 2007, "Photogrammetry, Geometry from Images and Laser Scans", ISBN 978-3-11-019007-6

Lerma García, J.L, 2002, "Fotogrametría moderna: analítica y digital", editado por la Univesidad Politécnica de Valencia

Perez, C., 2004, "Fotogrametría II: Ayudas al Estudio", Editado por el autor. Universidad de Salamanca. ISBN:84-607-5819-2

Schenk, T, 2002, "Fotogrametría digital, Volumen 1", Editado por Isaura E.Alonso Martínez – EUITTopográfica – UPM