UNIVERSIDAD DE ALCALÁ
ESCUELA POLITÉCNICA SUPERIOR
INGENIERIA DE TELECOMUNICACIÓN

TRABAJO FIN DE CARRERA

VALIDACIÓN DE UNA HERRAMIENTA DE RECONSTRUCCIÓN DE ESTRUCTURA PARA SU APLICACIÓN A VÍDEOS TOMADOS DESDE UN AVIÓN NO TRIPULADO (UAV)

David Núñez Clemente

2010

Validación de una herramienta de reconstrucción de estructura para su aplicación a vídeos tomados desde un avión no tripulado (UAV)

David Núñez Clemente

16 de junio de 2010

A mis padres y a mi hermano.

A David Julio Rodríguez (Julito), allá donde estés.

It is difficult to say what is impossible,

for the dream of yesterday is the hope of today

and the reality of tomorrow.

It is not a simple matter to differenciate unsuccesful from succesful experiments ...

[Most] work that is finally succesful is the result of a series of unsuccesful tests

in which difficulties are gradually eliminated.

Robert H. Goddard

Conócete, acéptate, supérate

San Agustín

Agradecimientos

Tras todos estos años de estudio es inevitable volver la vista atrás. Son muchas las personas a las que me gustaría agradecerles su apoyo, ya que de una u otra forma han sido partícipes de mi esfuerzo. Este trabajo es el fruto final de varios años de estudio en las aulas de la Universidad de Alcalá, un largo tiempo en el que no sólo he aprendido en el aspecto académico, sino que además he tenido la suerte de aprender de la gente que he conocido a lo largo de estos años.

En primer lugar quiero agradecer al INTA la concesión de la beca de formación Rafael Calvo Rodés, cuyo fruto son los conocimientos que he aplicado en este trabajo. Gracias a María Jesús Gutiérrez de la Cámara Ara, Directora del Departamento de Observación de la Tierra, Teledetección y Atmósfera del INTA y a María del Mar Melgar Fernández, Jefa del Área de Sistemas de Observación por promover esta beca dentro del área y por darme la oportunidad de disfrutarla. Gracias también al Departamento de Formación y a su Jefa, María del Val Mínguez, por ofertar este tipo de becas para estudiantes y por elegirme para una de ellas. Gracias especialmente a Severino Fernández, mi tutor en el INTA. No sólo por darme la oportunidad de formarme con él y compartir conmigo sus conocimientos, sino también por la supervisión y paciencia que ha tenido para que yo pudiese adquirirlos. Gracias por revisar esta memoria, aportar ideas, correcciones y por todo el apoyo y ayuda en la realización de este trabajo. A Francisco José López por el interés mostrado por mi trabajo en las reuniones que hemos ido haciendo y por los comentarios, ideas y preguntas que me han permitido ampliar mis conocimientos. A Roberto Gil, por participar en algunas de esas reuniones y aportar también sus comentarios y preguntas. A Francisco Javier Sánchez, mi compañero de despacho, a quien deseo toda la suerte del mundo en la tesis que dentro de poco empieza, gracias por el apoyo y los ánimos en las horas que pasamos cada día en el INTA. A las secretarias, Maribel, Pilar, Ana y Miriam y a Agustín por su amabilidad y por la ayuda que nos prestan siempre que la necesitamos. A Enrique Nicolás por enseñarme a tomar medidas con el GPS. A Javier Vargas por sus comentarios sobre la implementación de la triangulación y por el tutorial y las charlas de visión que alguna vez hemos tenido. A Javier Andrey por sus consejos y ayuda sobre LaTeX. Y también gracias al resto de gente del Departamento y del INTA por haberme hecho el tiempo con vosotros tan agradable.

Gracias a Pablo Fernández Alcantarilla por sus consejos a la hora de entender algunos conceptos y algoritmos de visión y su inestimable ayuda al resolver alguno de los problemas con los que me he encontrado en este trabajo. Ánimo con la recta final de la tesis, ya sólo queda una última zancada. A Javier Yebes por las conversaciones

sobre visión y la documentación de OpenCV. A Noah Snavely, el autor de Bundler, por su ayuda a la hora de entender el funcionamiento y la implementación de su aplicación y resolver los problemas y las dudas que me han ido surgiendo sobre la marcha. Gracias por responder con rapidez y dedicación a todos los e-mails que le he enviado, que no han sido pocos. A David Lowe por sus respuestas, que me sirvieron para entender el funcionamiento de SIFT.

A Pilar García, Directora de este trabajo, gracias por toda la libertad y flexibilidad que me has dado a la hora de realizarlo, por la confianza que en mí has depositado y también por la amabilidad, el apoyo y los ánimos con los que siempre me tratas. A Sancho Salcedo, del Departamento de Teoría de la Señal y a Francisco José Álvarez, del Departamento de Física, por vuestra amabilidad y por ofrecerme vuestra ayuda. Al resto de profesores de la Universidad de Alcalá, por todo lo que he aprendido de vosotros. También a los que tuve en Linköping, a los de mi Colegio, el *"Doctora de Alcála"* y mi Instituto, el *"Ignacio Ellacuría"*. Aunque ya lejos en el tiempo, también pusisteis vuestro granito de arena.

A mis amigos de toda la vida, con los que he crecido y que me acompañais desde el colegio y desde el instituto, ya sabeis quienes sois. Gracias por estar a mi lado durante todos estos años, por ser mi mejor apoyo en los malos momentos y excelente compañía en los buenos. A la gente de mis primeros años de carrera, aquellos que habeis estado siempre cerca. Gracias por todo el tiempo que hemos pasado juntos, por las horas de trabajo y también por las de diversión, y muy en especial a Julito. Tú nos diste tu enorme sentido del humor, tu inolvidable risa y tu optimismo. Siempre nos tendiste la mano cuando nos hizo falta ayuda. Te fuiste un día de otoño, pero sigues vivo en nuestro recuerdo.

A la *"Cuadrilla"* por las interminables horas de biblioteca que hemos pasado juntos, sobre todo los últimos cursos, las *batallas* delante del ordenador para sacar los trabajos adelante, el apoyo en las horas de estudio, los momentos de diversión y risas y las tardes de fútbol vistiendo la camiseta de los *"Kebranta"*. Gracias también a otras personas que he conocido estos años y que han formado parte de mi vida universitaria. A la gente de mi Erasmus en Linkoping y de mi Leonardo en Berlín, con vosotros amplié el horizonte de mi mirada. A Vito, por todos los momentos que hemos vivido en nuestro querido *korridor* de *Alsättersgatan 1*, y a Fer, Álvaro, Gergö, Madina, Emil, Adel, Mario y el *Polish Team* por todo el tiempo que pasamos juntos en aquella tierra. A nuestra Elena por tu vitalidad y por escucharnos y animarnos en los momentos de flaqueza. Gracias también a mis compañeras de piso, a Eli por convertir el estudio de tus padres en nuestra biblioteca, a Caro por todas las veces que me has ahorrado tener que cocinar estas últimas semanas, y a las dos por vuestros ánimos en el último tramo de mi trabajo.

Y para terminar, los más importantes, mi familia. A mis padres Ana y Patricio por el esfuerzo y los sacrificios que habeis hecho todos estos años por mí y por Óscar, y también por vuestra comprensión y confianza. A Óscar, mi hermano, por todo su ánimo y ayuda que sin percibirlo siempre me brinda. A mis tías Carmen y

Pili por su apoyo, a mi abuelo Germán por sus *batallitas*, que me dieron el empujón que necesitaba para viajar a Alemania y a mi abuela Vítora por las velas encendidas en mis últimos exámenes.

Y llegada esta línea, si me olvido de alguien, quiero darle también las gracias y que perdone mi despiste.

Alcalá de Henares, a 16 de junio de 2010.

Resumen

En este trabajo se evalúa el uso de una herramienta de reconstrucción de estructura perteneciente al estado del arte para su aplicación a vídeos aéreos. El objetivo final es comprobar la viabilidad de aplicar esta herramienta en vídeos grabados desde plataformas aéreas no tripuladas, comúnmente conocidas como UAV. Los resultados aquí descritos presentan los problemas identificados, así como las soluciones adoptadas. Finalmente, se definen las futuras líneas de investigación que permiten continuar con el trabajo realizado.

Palabras clave: UAV, reconstrucción de estructura, Bundler, visión computacional, SIFT.

Abstract

In this project I evaluate the use of a state of the art Structure from Motion (SfM) tool for its application on aerial videos. The aim is to test the validity of its application on videos recorded by Unmanned Aerial Vehicles (UAVs). The results obtained describe the identified problems as well as the solutions that have been adopted. In addition, future lines of research for the extension of the work done in this project are defined.

Keywords: UAV, Structure from Motion, Bundler, Computer Vision, SIFT.

Acrónimos

ANN Approximated Nearest Neighbor

CCD Charge-Coupled Device

CMVS Clustering views for Multi-View Stereo

CPU Central Processing Unit

CUDA Compute Unified Device Architecture

DDR Double Data Rate

DEM Digital Elevation Model

DoG Difference of Gaussians

e.p. Elaboración propia

EXIF EXchangeable Image file Format

GIS Geographic Information Systems

GPL General Public License

GPU Graphics Processing Unit

HALE High Altitude Long Endurance

HTML HyperText Markup Language

IDE Integrated Development Environment

INTA Instituto Nacional de Técnica Aeroespacial

LMA Levenberg-Marquardt Algorithm

LoG Laplacian of Gaussian

LTA Lighter Than Air

MVS Multi-View Stereo

PMVS Patch-based Multi-View Stereo

PNG Portable Network Graphics

RAM Random Access Memory

RANSAC RANdom Sample Consensus

ROV Remotely Operated Vehicle

SBA Sparse Bundle Adjustment

SfM Structure from Motion

SIFT Scale-Invariant Feature Transform

SLAM Simultaneous Location And Mapping

SVD Singular Value Decomposition

SURF Speed-Up Robust Features

TFC Trabajo Fin de Carrera

UAH Universidad de Alcalá

UAS Unmanned Aircraft System

UAV Unmanned Aerial Vehicle

UCAV Unmanned Combat Air Vehicle

UGV Unmanned Ground Vehicle

UPM Universidad Politécnica de Madrid

UTM Universal Transverse Mercator

UUV Unmanned Undersea Vehicle

Índice general

Lista de figuras

Lista de tablas

Parte I

Memoria

Capítulo 1

Introducción

El objetivo principal del trabajo es el estudio de tecnologías y herramientas del campo de la reconstrucción de estructura de código abierto para ser aplicadas al contexto de los vehículos aéreos, en concreto a aviones no tripulados (UAV). De entre todas las opciones disponibles se ha elegido Bundler[67], al ser la aplicación que mejor combina tanto robustez como sencillez de manejo. La herramienta seleccionada se ha probado en diferentes escenarios con el objeto de validar los resultados por ésta obtenidos. Los escenarios usados han sido de diferentes características, por un lado se han tomado fotogramas de secuencias de vídeo aéreo con mayor o menor resolución en píxeles y por otro lado se han empleado fotografías tomadas en tierra con una cámara convencional en un escenario conocido. También se ha creado un prototipo de aplicación en Matlab, denominado Orto3D, con una interfaz gráfica de usuario que permite cargar los datos del escenario reconstruidos por Bundler, para a continuación trabajar con ellos.

Debido a varios factores, la visión aplicada a vehículos aéreos tiene unas características concretas de operación. Todas las comunicaciones realizadas con estos vehiculos usan lógicamente el canal aéreo, con todos los problemas que este tipo de comunicaciones acarrean, como ruido, ancho de banda limitado, condiciones de operación extrema, especialmente en sistemas militares o de emergencia donde los requirimientos exigen datos en tiempo real, fiabilidad, robustez, comunicaciones resistentes a errores, entre otros. Las imágenes obtenidas tienen unas limitaciones importantes de resolución y compresión determinadas por el ancho de banda y la necesidad de obtener imágenes en tiempo real, a lo que hay que añadir los problemas de estabilización de imagen debidos al propio movimiento del vehículo. Todas estas características hacen que sea interesante estudiar la aplicación de herramientas de visión computacional al campo de los UAVs a partir de herramientas disponibles de código abierto y de propósito más general. En concreto el objetivo de este trabajo es obtener reconstrucciones de estructura de conjuntos de imágenes para recuperar escenarios de los que se desconoce tanto la posición de las cámaras como los parámetros internos de calibración de estas. Éste es el escenario más complejo dentro de la reconstrucción de estructura, ya que no se dispone de absolutamente ninguna información a priori salvo los propios fotogramas, y todo dato necesario debe ser extraído de las imágenes.

El campo de la visión es de gran importancia en el desarrollo de futuros sistemas UAVs, dotándoles de un gran abanico de aplicaciones tanto en el sector civil como en el militar. En los próximos años se espera que los aviones no tripulados despeguen en el ámbito civil mientras que en el campo militar están plenamente integrados y serán un sistema de gran importancia en el futuro.

El proceso de reconstrucción de un objeto o un escenario a partir de una secuencia de imágenes trata de emular la capacidad de visión y aplicarla a ordenadores y robots. Para que la reconstrucción 3D sea posible es necesario disponer de un conjunto de fotogramas tomados con diferentes cámaras desde diferentes posiciones, o de forma análoga, una misma cámara colocada en cada una de esas posiciones. Un algoritmo de detección de puntos será el encargado de obtener los puntos similares entre fotografías. Cada punto detectado es la proyección en el espacio bidimensional de un punto perteneciente a un objeto en el espacio 3D. A partir de la detección de una serie de puntos correspondientes, es posible determinar las posiciones de las cámaras y de los puntos en el espacio tridimensional con la ayuda de algoritmos de visión.

Bundler, el programa de reconstrucción de estructura seleccionado, permite realizar esta tarea. A partir de una secuencia de imágenes de entrada en formato JPEG, es capaz de devolver las poses de las cámaras empleadas, así como una nube de puntos reconstruidos del escenario. El formato de salida es un fichero de texto cuyos datos pueden ser leídos por otra aplicación. En concreto, en el trabajo aquí presentado, se ha desarrollado la aplicación Orto3D para validar los resultados del escenario, la posición de las cámaras y la estructura de la nube de puntos. La validación de los resultados es el principal problema de estos puntos ya que en la mayoría de los casos, la estructura devuelta no permite reconocer el escenario inicial. En algunos casos esto se debe a que el número de puntos detectados es muy pequeño como para visualizar la estructura, en otros a que la reconstrucción de las cámaras no es correcta. A lo largo de este trabajo se han encontrado diferentes problemas que explican la razón por la que se obtienen malas reconstrucciones, incluso para una herramienta probadamente robusta como Bundler. Entre las razones, se encuentran la resolución baja de las imágenes de entrada (las del SIVA en concreto), que reduce el número de puntos característicos y correspondientes, y especialmente el desconocimiento de la longitud focal de las cámaras, causante de una mala inicialización del proceso de reconstrucción de estructura, y totalmente determinante en el resultado final.

La implementación en Matlab de esta aplicación de validación ha exigido realizar un análisis exhaustivo de varios algoritmos de la teoria de visión computacional, necesarios para entender adecuadamente los algoritmos internos de la aplicación Bundler así como los necesarios para implementar el programa de validación. Esta aplicación permite visualizar los resultados y comprobar que son coherentes con los objetos originales, que es la primera verificación de la validez de la reconstrucción 3D (a simple vista se puede ver que el escenario es correcto o no). Como ayuda en el reconocimiento visual de la estructura, se han desarrollado una serie de funciones que permiten añadir poligonales y aristas a partir del marcado manual de puntos con ayuda de la geometria epipolar (la condicion epipolar que define las líneas epipolares), y la triangulación a partir de los puntos marcados para añadirlos al escenario tridimensional. Para además convertir el escenario con una escala desconocida, a otro en metros, se incluye funcionalidad para georreferenciar escenarios a partir de ortoimágenes de referencia de las que se conocen sus coordenadas UTM. La georreferenciación se implementa con transformaciones (escalado, traslación y rotación) para ajustar ortoimagen y planta del escenario 3D. Para calcular los datos de transformación, el usuario toma unos puntos de referencia, a partir de los cuales se puede obtener la relación de giros, escalados, y traslaciones necesarias.

Los objetos reconstruidos para este trabajo corresponderán a edificios cuyos puntos pertenecerán a paredes, ventanas u otros objetos presentes en el escenario. La visualización de puntos pertenecientes a paredes, permitirá referenciar éstos a planos u ortoimágenes, tomando como referencia las plantas de edificios u otros objetos que sean visualmente reconocibles en planta.

1.1. Motivación

El campo de los UAV es un campo en crecimiento sobre todo en el campo civil. Existen numerosos vehículos de este tipo aplicados al sector militar, pero los UAV civiles todavía no han despegado en el mercado como se comenta en 3.1. Según las estimaciones de [75], la observación de la Tierra, junto con otras aplicaciones relacionadas con el campo de tratamiento de imágenes y la visión computacional, ocupan gran parte del campo de aplicación y de la cuota de mercado de futuro próximo.

Actualmente existen proyectos en esta línea bastante avanzados como por ejemplo 2D3, comentado en el capitulo 2 sobre el estado del arte, que dispone de software para uso con imágenes tomadas de UAV, o proyectos llevados a cabo por la UPM con su proyecto COLIBRÍ, empleado para la evaluación de tendidos electricos y que se comenta en los capítulos 2 y 3 entre otros trabajos.

Dados estos antecedentes, el desarrollo de un trabajo con relación a ambos temas intercombinados es de gran interés. En concreto, este proyecto se centra en la reconstrucción de estructura a partir de imágenes tomadas con cámaras embarcadas. El reto de este trabajo viene dado por el uso de imágenes de las cuales no se conoce ningun parámetro para a partir de ellos estimar tanto la estructura como el movimiento de cámara en el escenario. Este es el peor escenario posible en la reconstrucción, ya que no se tiene ninguna restricción sobre el escenario. A partir del movimiento de las cámaras se podrá recuperar la trayectoria del vehículo si los fotogramas empleados son consecutivos, y separados por una cierta distancia entre fotogramas.

Para desarrollar este trabajo se ha impuesto como requisito el uso de herramientas del estado del arte, de código abierto y robustas. De las diferentes opciones existentes, Bundler ha sido la elegida por diversos motivos que se explican en esta memoria. No obstante, otras aproximaciones son igualmente válidas. Esta herramienta se puede aplicar únicamente en postprocesado de la imagen y en ningún caso se plantea como un trabajo para aplicación en tiempo real. La reconstrucción de estructura es un proceso lento y pesado que salvo aplicando restricciones en las cámaras o en el movimiento, no puede aplicarse bajo condiciones de tiempo extrictas. El tiempo de procesamiento de las correspondencias, la búsqueda exhaustiva de puntos característicos y el procedimiento de *bundle adjustment*, aunque optimizado, al ser exhaustivo y general, consume una gran cantidad de tiempo y necesita de capacidades de cómputo elevadas, factores que hacen que el proceso sea lento e imposible de aplicar a procesadores embarcados. Para mejorar el rendimiento se pueden emplear técnicas de computación paralela, pero incluso en el mejor de los casos, no sería posible disponer de resultados en tiempo real.

La mayoría de los trabajos de aplicación de la visión a UAVs se centran en la implementación de sistemas en tiempo real, donde es necesario usar otras aproximaciones no tratadas en este trabajo, siendo SLAM visual la más extendida. Esta aproximación es más

eficiente a costa de trabajar con una cantidad inferior de datos y obteniendo únicamente un mapeado en tiempo real de las zonas en vez de estructuras densas. Para ello se usan estimaciones y restricciones en el movimiento de las cámaras, cuyos parámetros se conocen con certeza. La aproximación aquí desarrollada presenta un enfoque diferente. A diferencia de la técnica SLAM, no busca mapeado en tiempo real, sino la obtención de estructuras densas de escenarios, y poder así disponer de cámaras registradas de forma aproximada en un mapa, sin disponer de información previa sobre su posición y necesitando únicamente un reconomiento visual de la zona de vuelo. Este reconocimiento, que se trata básicamente de una tarea de fotointerpretación, es necesario para poder tomar una ortoimagen correspondiente al escenario grabado. En cierto modo, el tratamiento aquí buscado es similar al objetivo para el que Bundler fue diseñado. Ésta herramienta permite obtener reconstrucciones usando cualquier tipo de cámara, incluso imágenes obtenidas con búsquedas por palabras en servidores de imágenes como Flicker[19], sin necesidad de información sobre las poses de las cámaras. Siguiendo esta idea, y aplicándola a fotogramas de vídeos, se podrían hacer bases de datos de imágenes geolocalizadas en posiciones aproximadas.

1.2. Objetivos

Los objetivos perseguidos se pueden resumir en los siguientes puntos:

- Comprobar la posibilidad de usar alguna herramienta de visión computacional de código abierto para la reconstrucción de estructura, con vídeos o fotografías tomados desde plataformas aéreas no tripuladas. Los vídeos tomados desde avión, se realizan a cierta altura y con cierta distancia al objetivo, con además un ancho de banda muy limitado en el canal de comunicación que tendrá un límite en la transmisión de vídeo y que afectarán a la reconstrucción final. La mayoría de las herramientas disponibles están diseñadas con un propósito general sin un ámbito tan específico como el que aquí ocupa, por lo que la evaluación de alguna herramienta de SfM en este ámbito concreto es de gran interés.
- Seleccionar de entre las herramientas de código abierto del estado del arte disponibles, aquella que proporcione resultados más robustos y fiables, probando los resultados en diferentes escenarios. Esta herramienta debe ser además eficiente.
- Que esta herramienta sea de código abierto y que sea posible modificarla o adaptarla a nueva funcionalidad que pudiera definirse como futuro trabajo para desarrollar una aplicación propia y sencilla que permita la reconstrucción de estructura a partir de fotografía y vídeo.
- Buscar una herramienta de visualización de resultados, o crear una propia con la funcionalidad necesaria para validar el software seleccionado.
- Validar los escenarios en los que se prueba la herramienta, georreferenciando tanto puntos como cámaras obtenidas.
- Aprender y obtener conocimientos sobre una serie de algoritmos y técnicas de tratamiento de imagen y visión computacional que permiten realizar las reconstrucciones de estructura de un escenario dado, y en concreto sobre los incluidos en la aplicación empleada.
- Definir una serie de futuros trabajos que pudieran dar continuación al presente trabajo.

1.3. Problemas

Por un lado disponer de este tipo de vídeos para trabajar con ellos es caro, y poco accesible a no ser de disponer de algún vehículo de este tipo. Sin embargo existen múltiples vídeos aéreos para documentales o televisión que pueden emplearse para simular un entorno de operación con UAVs. Otro problema adicional es el desconocimiento de los parámetros de calibración de cámara, lo cual se repite en todos los vídeos empleados en este trabajo. La ausencia de datos sobre las focales hace que las reconstrucciones no sean precisas y en algunos casos, que éstas no sean ni fiables ni robustas. Los fotogramas empleados se han tomado tanto de vídeos tomados desde UAV como de vídeos tomados desde helicopteros de televisión para documentales. Los vídeos con mejor resolución empleados, obtenidos de secuencias tomadas por helicopteros de televisión, dan mejores resultados que los vídeos de ejemplo usados tomados desde el UAV SIVA. Otro problema añadido son las vibraciones asociadas al vehículo, al motor y al movimiento de la cámara, más si cabe en un UAV que en un avión tripulado, por su menor peso y mayor inestabilidad en la mayoría de los casos. La calidad de la reconstrucción viene también determinada por la distancia entre el objeto, la cámara y el vehículo aéreo. Otras fuentes de vídeo de buena resolución además de los documentales son las imágenes tomadas en eventos deportivos como la Vuelta a España, etc., donde se realizan horas de grabación en entornos urbanos interesantes para las pruebas de reconstrucción. No disponer de una fuente de información con coordenadas conocidas añade un problema adicional, al no poder comprobar fehacientemente los resultados de las posiciones del vehículo calculadas por Bundler con respecto a las posiciones reales. Al no conocer más datos sobre el vuelo, esta validación es imposible. Una posible validación adicional sería trabajar con vídeos con posiciones GPS del vuelo conocidas y comprobar la precisión de los resultados obtenidos.

En ningún caso este proyecto puede dar lugar a una herramienta comercial por varios factores. Por un lado, al no disponer de datos de las cámaras, el sistema no obtiene resultados precisos, y por otro lado la aproximación aquí presentada trabaja únicamente bajo unas condiciones de operación concretas. Sin embargo, las ideas aquí aportadas abren una línea de trabajo que pudiera dar lugar a largo plazo y con los recursos necesarios a una herramienta robusta y fiable.

1.4. Estructura del documento

El presente documento se estructura en tres partes bien diferenciadas, por un lado la memoria del documento, dividido a su vez en seis capítulos, por otro lado el pliego de condiciones y el presupuesto, y en tercer lugar, los apéndices con los manuales de Bundler y Orto3D y el contenido del DVD y algunas notas adicionales. La primera parte contiene el presente capítulo, con una pequeña introducción al contenido del trabajo y a los objetivos buscados. El segundo capítulo se centra en el estado del arte de la tecnología de visión computacional, en especial, de la reconstrucción de estructura, con el objetivo de enfocarlo al uso de plataformas aéreas. El tercer capítulo expone las características de las plataformas aéreas no tripuladas, a las que estaría enfocado el uso de esta tecnología. El objetivo perseguido con este capítulo es dar al lector una pequeña introducción al campo de los vehículos aéreos, incluyendo varios ejemplos y referencias donde ampliar información. El capítulo cuarto pretende explicar brevemente los fundamentos teóricos detrás del tema tratado, haciendo un resumen de los mismos para que sea posible seguir el resto de la memoria sin dificultad. El capítulo quinto presenta el trabajo realizado, incluyendo la

reconstrucción de estructura con Bundler, los resultados obtenidos con varios escenarios, y la implementación del prototipo Orto3D para visualización y validación de los datos de salida de Bundler. Por último, en el sexto capítulo se habla de las conclusiones a las que se ha llegado y de los trabajos futuros que pueden surgir del trabajo aquí presentado.

Capítulo 2

Estado del arte

La visión computacional se usa hasta la fecha en una gran variedad de aplicaciones y entornos. En el caso del trabajo presentado en esta memoria, se combinan dos campos interesantes, por un lado la visión computacional en concreto la reconstrucción de estructura, y por otro lado el uso de vehículos aéreos no tripulados para la toma de imágenes de los escenarios reconstruidos. La obtención de información tridimensional de un escenario a partir de imágenes exige procedimientos y algoritmos capaces de manejar un gran conjunto de parámetros cuyo número aumenta conforme lo hace el número de imágenes empleadas. En los siguientes párrafos se presenta por un lado el estado del arte en la reconstrucción de estructrua SfM y por otro lado el estado del arte de la aplicación de la visión computacional al entorno concreto de los UAV.

La reconstrucción de estructura (SfM) se define como el proceso de crear modelos tridimensionales a partir de colecciones de imágenes de entrada, recuperando los parámetros de cámara (las poses y calibración) y la geometría 3D (los puntos 3D). Existen diferentes aproximaciones para implementar este tipo de algoritmos, en base a restricciones conocidas impuestas al escenario, tales como posiciones de las cámaras o calibraciones internas. Cuando se conocen los parámetros internos de las cámaras, la reconstrucción es más precisa que cuando no se conocen en absoluto. De ahí que el peor caso en términos de complejidad sea aquel en el que no se conoce ni puntos de referencia en la estructura ni parámetros de las cámaras. Este proceso además se complica en mayor medida cuando se intenta reconstruir escenas con diferentes cámaras, luminosidad, escala, zoom(variaciones de longitud focal), etc.

La reconstrucción de estructura es un aréa de investigación muy activa en el campo de la visión computacional, por sus interesantes aplicaciones prácticas. El rango de soluciones que podrían basarse en esta tecnología es muy variado, por poner un ejemplo se pueden implementar modelos de ciudades en 3D para turismo o navegación asistida en base a contenido visual, implementación de simuladores, o aplicaciones de realidad aumentada para cine o televisión. Hasta la fecha existen diversas soluciones de código abierto que permiten realizar reconstrucciónes basadas en SfM, además de permitir la modificación del código fuente para incluir mejoras en éste o cambiar partes del mismo para un objetivo específico. Hay que tener en cuenta que en el campo de la visión no existe una única metodología que permite resolver un problema, sino que existen diferentes técnicas o procedimientos con mejores o peores resultados en función de la finalidad concreta que se busque. De todas las aproximaciones disponibles, las más relevantes para este trabajo son aquellas que permiten la recuperación del movimiento de cámara y la estructura de una forma robusta

y eficiente, sin que se tenga información alguna a priori sobre éstas.

Bundler [67] es un buen ejemplo de herramienta de reconstrucción robusta y eficiente, razón por la cual ha sido empleada en las pruebas de este trabajo. A priori, puede obtener datos de calibración de las cámaras sin tener datos ni sobre éstas ni sobre el escenario, usando únicamente las imágenes de entrada. Sin embargo, como se verá más adelante, la aplicación necesita conocer con certeza la focal del primer par. En cualquier caso, Bundler es una aplicación muy potente en comparación con otras existentes al estar implementado en un lenguaje de programación (C/C++) más eficiente en comparación con las aplicaciones basadas en Matlab. Además, al combinar algoritmos del estado del arte de la tecnología permite obtener resultados robustos bajo unas condiciones de operación. Otro aspecto positivo de Bundler es que se trata de una aplicación de código abierto basada en un trabajo anterior denominado Photo Tourism implementado por el mismo autor [69][70][71]. Se trata de una herramienta diseñada con el objeto de usar grandes conjuntos de imágenes de entrada, tomadas a partir de búsquedas en aplicaciones como Flickr[19] o Picasa[53]. Estos servicios web permiten compartir en la red multitud de fotos introducidas por multiples usuarios. La idea de la aplicación Photo Tourism es realizar reconstrucciones de edificios o lugares con gran cantidad de imagenes en la red (siendo los lugares turísticos los más prestables a ello). Cada fotografía, al estar realizada por diferentes usuarios y con diferentes cámaras podrá tener parámetros de calibración muy diferentes. Las imágenes también se tomarán a diferentes horas del dia con variadas condiciones lumínicas y desde diferentes puntos de la escena. Para manejar este conjunto de datos, la aplicación implementada debe ser lo suficientemente robusta como para ser capaz de trabajar adecuadamente con tantas variables, condición que Bundler cumple. La principal diferencia entre Photo Tourism y Bundler es que la primera tiene como funcionalidad la navegación por un álbum de fotos proporcionando al usuario una interfaz interactiva donde navegar por el conjunto de fotos en base a las posiciones de las cámaras en la escena, pero sin presentar la nube de puntos reconstruida. La segunda aplicación devuelve la estructura en una nube de puntos y las posiciones de las cámaras. En [3] se presenta un ejemplo de los resultados obtenidos con Bundler a partir de escenarios tomados de Flickr. En este ejemplo, el tiempo de reconstrucción se optimiza mediante el uso de técnicas de computación en paralelo usando múltiples procesadores. Los resultados se explican con mas detalle en la página web dedicada a los ejemplos [4], incluyendo las reconstrucciones, el número de imágenes empleadas, el número de procesadores usados y las horas de computación que ha llevado el proceso de reconstrucción.

A partir del trabajo realizado con Photo Tourism, la empresa Microsoft ha desarrollado una aplicación web de ámbito comercial gratuita, denominada Photosynth [44], siguiendo una idea de desarrollo similar y probablemente un código fuente y algoritmos internos muy parecidos[1]. El objeto de esta aplicación es visualizar interactivamente álbumes de fotos en función de la localización de las cámaras en el escenario visualizado. Los álbumes de fotos visualizados son subidos por el usuario al servidor de la aplicación. Por otro lado, también se ha creado un servicio web basado en Bundler denominado PhotoCity [83] con la idea de servir de juego y que permite reconstrucción de escenarios a partir de imágenes subidas al servidor ya sea con cámaras convencionales o con un teléfono móvil. En este servicio se puede ver un ejemplo trabajo cooperativo entre múltiples usuarios para mejorar escenarios creados con anterioridad. Para cada imagen añadida se obtiene su posición en

[1]No se trata de una herramienta de código abierto que permita comprobarlo y carece de documentación técnica disponible

Figura 2.1: SfM con Bundler

Reconstrucción de estructura del Coliseo de Roma a partir de múltiples imágenes. Fuente: [67]

el escenario y se añade a la reconstrucción los nuevos puntos encontrados. El interés de esta herramienta es que introduce la posibilidad de implementar un servicio de reconstrucción de estructura basada en internet y con la posibilidad de trabajar con teléfonos móviles.

Relacionado con el problema de reconstrucción de estructura, pero basado en una aproximación diferente, se encuentra el software de "Multi-View Stereo"(MVS). A diferencia de los programas de SfM, necesita disponer de datos sobre las cámaras para obtener la reconstrucción. Estos datos son tanto la calibración como la pose relativa de la cámara. El algoritmo calcula la posición de los puntos 3D y las normales a las superficies en estos puntos para proporcionar a cada uno un parche que se usará para obtener una reconstrucción densa del escenario.. El parche se obtiene en base a los puntos que rodean a los puntos característicos encontrados. Un ejemplo de este software es la aplicación de código abierto PMVS [21][23]. Esta aplicación toma como entrada un conjunto de imágenes y parámetros de cámaras para reconstruir la estructura de una escena visible en las imágenes. Como el algoritmo necesita introducir los parámetros de las cámaras para poder ejecutar la aplicación, y el software no recupera ningún parámetro de cámara a partir de las imágenes, es indispensable usar un software de SfM como paso previo. Esta necesidad hace que este software no se pueda aplicar para obtener escenarios densos si no se conocen inicialmente las cámaras. Sin embargo, debido a la calidad de los escenarios densos, como los mostrados en [63] y [64], combinando Bundler con este método[67], se pueden obtener reconstrucciones con pequeño error y además fácilmente reconocibles visualmente, mejor que únicamente con las nubes de puntos. Para ello Bundler incluye un conversor de la salida de Bundler al formato de entrada de PMVS, denominado Bundler2PMVS. La funcionalidad aportada en esta herramienta se mejora con CMVS [22][2] Este último se ha desarrollado con el objeto de manejar adecuadamente conjuntos grandes de imagenes, que la mayoría de algoritmos de reconstrucción no son capaces de escalar adecuadamente. Para ello toma la salida de un programa de reconstrucción de estructura (SfM) como entrada y descompone este conjunto de imágenes en subconjuntos de tamaño más fácilmente manejables. Cada subconjunto se puede procesar con un softaware de MVS (Multi View Stereo), como PMVS. El programa de SfM usado en este caso puede ser Bundler por ejemplo. La secuencia de trabajo sería la mostrada en la figura 2.2.

[2]El trabajo presentado en esta web se publicará en CVPR 2010 con el título "Towards Internet-scale Multi-view Stereo", Yasutaka Furukawa, Brian Curless, Steven M. Seitz and Richard Szeliski. Este paper no se encuentra disponible a la fecha de escribir esta memoria.

Otros trabajos realizados sobre el tema de reconstrucción de estructura son las herramien-

Figura 2.2: Bundler con PMVS y CMVS
Secuencia de ejecución de Bundler en combinación con PMVS y CMVS para la obtención de modelos densos de escenarios a partir de una reconstrucción de estructura.

tas de Philip Torr[73] y de Vincent Rabaud[57]. Ambas están implementadas en Matlab, por lo que son menos eficientes que las basadas en C/C++. Otro servicio web relacionado es ARC3D [1][2][80], publicado por el grupo de visión de la Katholieke Universiteit Leuven. Mediante la descarga de una aplicación cliente en el ordenador y el registro en la página de la aplicación, el usuario puede comenzar a subir imágenes al servidor para reconstruir el escenario. Con respecto a este tipo de soluciones en comparación con Bundler, las basadas en servicios web tienen un servidor con toda la parte de reconstrucción de estructura, dejando en el cliente únicamente la funcionalidad de elegir, subir las imágenes y ver las reconstrucciones. El código fuente usado en el servidor no es abierto, siendo la aplicación una caja negra para el usuario. Además, al tener que subir las imágenes al servidor, se pierde la confidencialidad que algunas imágenes pudieran tener. Un ejemplo práctico de combinación de SfM con UAVs es el trabajo presentado en [59], para la reconstrucción de estructura con objetivos de restauración arquitéctónica. El proyecto emplea helicópteros de radiocontrol modificados para poder tomar imágenes de las agujas de la Catedral de Milán que sirvan para planificar la restauración. El proceso de reconstrucción se basa en puntos obtenidos con SIFT, al que se le buscan las correspondencias para realizar a continuación SfM con optimización basada en bundle adjustment. Aunque no se especifica con detalle el método, es similar a grandes rasgos al método usado por Bundler.

Con respecto a la georreferenciación de escenarios reconstruidos con software SfM, solo se ha encontrado una referencia en [71], idea que se ha seguido en el prototipo de georreferenciación implementado en este trabajo. En las figuras 2.3 y 2.4 se presenta un ejemplo de escenario georreferenciado con Photo Tourism.

La reconstrucción de estructura es un procedimiento costoso en computo, de ahí que no sea posible aplicar programas de SfM pesados a sistemas de tiempo real. El enfoque aplicado a UAV para SfM sería la toma de imágenes para postprocesado, para reconstruir los escenarios después de la misión, o durante la misión pero sin estar los algoritmos embarcados en el vehículo aéreo. En el caso de requerir software de tiempo real, el enfoque aplicado a los UAV es diferente. En esta línea están los trabajos llevados a cabo por varios equipos, por ejemplo de la UPM [10] y de la Universidad de Sevilla [9], orientados al desarrollo de técnicas que permitan a los UAVs maniobrar de forma autónoma en espacios en base a información visual. Una implementación en esta línea es interesante ya que permite tener un método redundante de información cuando la señal GPS no es válida o fiable. Incluso, los algoritmos de procesamiento en tiempo real permiten seguir objetos en

Figura 2.3: Photo Tourism vista 1

Ejemplo de reconstrucción de un escenario con Photo Tourism. Fuente: [67]

Figura 2.4: Photo Tourism vista 2

Planta de la reconstrucción del mismo escenario sobre una ortoimagen en Photo Tourism. Fuente: [67]

movimiento (coches o similares). Una de las líneas interesantes de trabajo de este equipo es la aplicación de la técnica de visual SLAM a un UAV. Entre las aplicaciones posibles de un aparato con esta tecnología estaría la inspección de líneas eléctricas, seguimiento de objetivos móviles, estimación de distancia estéreo, mapeado y posicionamiento. El algoritmo SLAM implementado en este ejemplo, se basa en la detección de puntos con el detector de esquinas de Harris, o el algoritmo SIFT.

Hasta la fecha existen multitud de investigaciones que relacionan los campo de visión y UAVs, y que no se limitan a la reconstrucción de estructura ni al mapeado y navegación con SLAM. Hablar de todas las opciones de explotación de vídeo con vehículos aéreos autónomos se escapa del objetivo de este trabajo por lo que en las siguientes líneas se mencionarán algunos trabajos interesantes a partir de cuales se pueden encontrar más referencias sobre el tema. Por un lado está el trabajo realizado para demostrar la posibilidad de usar diferentes sistemas autónomos tanto aéreos como terrestres que interoperen y realicen un trabajo cooperativo compartiendo datos obtenidos por diferentes sensores en

Figura 2.5: UAV Colibrí de la UPM

Ejemplo de UAV usado para aplicaciones de visión computacional y desarrollado en la UPM. Fuente: [78]

cada vehículo, incluyendo cámaras para visión computacional [41][42]. También se pueden encontrar trabajos para inspección de carreteras, oleoductos y seguimiento de tráfico como el presentado en [56], para explotación de vídeos aéreos [30][61] o para mapeado de terreno y DEM [27].

Para cerrar el capítulo, al margen de los citados grupos de investigación, la empresa 2D3 [84] trabaja en el campo de los UAV con visión. Se trata de una empresa surgida a partir de miembros del grupo de visión (VGG) de la Oxford University. Las aplicaciones desarrolladas para UAV están orientadas al sector militar y gubernamental. VALS(Visually Assisted Automatic Landing System) es una aplicacion de asistencia al aterrizaje para UAV basada totalmente en visión y que actúa como método redundante cuando fallan los métodos de navegación del vehículo. Se basa en un algoritmo de detección de puntos característicos en la imagen que en tiempo real, con una frecuencia de muestreo de 30fps. A partir de los puntos, el sistema es capaz de recuperar el movimiento de la cámara (y por ende del vehículo), y la geometría de la escena, que al conocer datos sobre las pistas de aterrizaje (mapas) permite determinar la altura del vehículo y su distancia a la pista. El programa TacitView permite la realización de mosaicos de terreno desde el aire a partir de la recuperación de la geometría del escenario, seguimiento de vehículos, o realización de mapas de altitud a partir de imágenes aéreas. Incluso también permite la recuperación de estructura del escenario.

Capítulo 3

Vehículo aéreo no tripulado (UAV)

3.1. ¿Qué es un UAV?

Se define como UAV a todo vehículo aéreo sin piloto con la capacidad de volar de manera autónoma o cuando sea necesario, controlado desde tierra. Éstos vehículos son reusables y pueden operar en una gran variedad de misiones, siendo controlados y supervisados desde una estación terrena. La mayoría de los aparatos existentes hoy día han sido diseñados para uso militar, sin embargo, su utilización en aplicaciones civiles se hará más común en los próximos años [77]. El término describe un amplio rango de vehículos con diferentes configuraciones, capaces de operar en el espacio aéreo, ya sean aviones, helicópteros o dirigibles. Los más pequeños, denominados microUAVs, pueden tener el tamaño de un decímetro, y los más grandes pueden tener una envergadura de decenas de metros [48]. Estos últimos pueden volar a grandes altitudes durante largos periodos de tiempo, e incluso podrían ser usados como repetidores para comunicaciones. Un ejemplo de este tipo de avión de larga duración (HALE) es el RQ-4 Global Hawk capaz de volar hasta 35 horas sin parada[58] (figura 3.3). Para implementar el sistema de navegación, estos vehículos incorporan sistemas de control y sistemas de posicionamiento GPS, que combinados con sistemas de información geográfica (GIS) en la estación de control permiten que el vehículo se mueva a lo largo de una ruta. Estas rutas pueden ser preprogramadas, o controladas desde una posición remota. Incluso los sistemas autónomos más avanzados son capaces de reprogramar la ruta ante un evento no planificado que ocurra durante el vuelo, por ejemplo cancelación de la misión si hay riesgo para el aparato. De manera adicional algunos UAV usan la información visual tomada con camáras como fuente redundante para la navegación [10]. La CPU de a bordo se encarga de pilotar sin que sea necesario disponer de una persona a bordo. Dado que estas unidades deben ser especialmente fiables y robustas para evitar errores catastróficos, es necesario que sean diseñadas de manera redundante y tolerante a fallos.

Los UAV son muy flexibles en comparación con los vuelos tripulados y los sistemas satelitales. Pueden llevar a cabo lo que se denomina como tareas D3 : *dull, dangerous and dirty* (tareas tediosas, peligrosas y desagradables) [75]. Un sistema no tripulado tiene la gran ventaja de poder permanecer en localizaciones remotas por periodos largos de tiempo sin tener el problema asociado al cansancio de los pilotos. Al ser combinados con la carga útil apropiada, pueden proveer una plataforma sensora aérea extremadamente eficiente. Además, dado que son diseñados con gran modularidad, permiten cambiar sencillamente la carga útil incorporando en ellos las tecnologías de teledetección más modernas. También

Figura 3.1: Lanzamiento del UAV Fulmar

Algunos UAV necesitan iniciar el vuelo desde plataformas de lanzamiento como el UAV Fulmar de Aerovisión.

Fuente: [20]

ofrecen la posibilidad de servir como estaciones de repetición de comunicaciones aéreas, llenando un nicho tecnológico vacío hasta la fecha. Esta tecnología se basaría en el uso de UAVs tipo HALE. Otra posible aplicación de los UAVs es como transporte de mercancías.

Llegado a este punto es necesario diferenciar los términos que habitualmente se emplean para definir los vehículos aéreos no tripulados y las tecnologías con ellos relacionadas. Por un lado, UAV es el término general usado para referirse al vehículo, también denominado plataforma, al que se acoplarán las cargas útiles. En numerosas ocasiones se emplea también el término UAS (Unmanned Aircraft System), que incluye el sistema aéreo formado por el avión o helicóptero junto con las cargas útiles, el sistema terreno formado por la estación de control y los enlaces de comunicaciones que interconectan ambos sistemas, todos ellos integrados en un entorno de toma de decisiones [76]. Las cargas útiles son todos los subsistemas que se embarcan en el vehículo aéreo para cumplir una misión específica, en éstos se incluyen sistemas de comunicaciones, cámaras u otros dispositivos sensoriales. El enlace de comunicaciones se puede usar para control remoto, transmisión de datos de telemetría, así como para la recepción de datos tomados por las cargas útiles, por ejemplo vídeo en tiempo real. En algunas ocasiones puede encontrarse el término UAVS usado como sinónimo de UAS. Dentro de los UAVs de propósito militar, se emplea también el término UCAV, que hace referencia a aviones UAV de combate. La diferencia con el resto de vehículos del mismo tipo es su uso en misiones de ataque, embarcando armamento además de sensores. Este tipo de aviones están siendo usados en escenarios bélicos como Afganistán o Irak.

Existen diferentes métodos de clasificación de vehículos no tripulados, con mayor o menor detalle de diferenciación en función del rango de operación, altitud máxima, tiempo de operación, peso entre otros factores. Por simplicidad, aquí se muestra una clasificación en cuatro grupos en función del peso y del rango de operación y un quinto grupo reservado a los helicópteros autónomos. La clasificación está basada en las presentadas en [8] y [77]:

- MicroUAV: inferior a 7Kg de peso con altitud de vuelo inferior a 250 metros, pu-

Figura 3.2: Cámara del UAV Fulmar
Ejemplo de UAV con cámara embarcada. Fuente: [20]

diendo volar en interiores. Tienen la ventaja de ser fácilmente desplegables y recuperables. Pueden usarse como ayuda en el rescate de personas tras un terremoto u otro tipo de catástrofe. También pueden usarse como estaciones de repetición de comunicaciones.

- Mini-SmallUAV: peso en el rango de 8 a 400Kg con altitud máxima de vuelo entre 150 y 5000 metros. Pueden ser desplegados sin la necesidad de una pista de despegue o aterizaje. Para ello se usan lanzaderas y redes o paracaídas para recogerlos, proporcionándoles una mayor flexibilidad. Pueden usarse como sistemas de monitorización de incendios o tráfico, e incluso para la agricultura. Dentro de este grupo se pueden incluir los subgrupos Close Range (CR), Short Range (SR), Medium range (MR) y Long Range (LR) presentados en [8].

- Rotatorio: tienen la capacidad de despegar y aterrizar verticalmente y además pueden sostenerse en el aire, se pueden subdividir a su vez en helicópteros de un rotor, de rotores coaxiales y de rotor inclinado. Un ejemplo de este último es el US Navy Eagle Eye. Los helicópteros de un rotor tienen un bajo coste de desarrollo y mantenimiento, además de ser más sencillos que los helicópteros de rotor coaxial. Éstos últimos sin embargo tienen rendimientos superiores [77]. Las características de estos vehículos son similares a los que tienen los dos subgrupos anteriores.

- MALE: pesan entre 400 y 4000Kg con altitud de vuelo entre 5000 y 8000 metros. Pueden reemplazar pilotos que lleven aplicaciones similares. Requieren pistas de despegue y aterrizaje, y necesitan un mayor mantenimiento y reparación con las logísticas asociadas. Al compartir espacio aéreo con los aviones tripulados, necesitan medios de detección y prevención de colisiones. Pueden usarse para patrulla de costas, monitorización de inundaciones, mapeado digital de terrenos, ayuda en emergencias, búsqueda y rescate, monitorización y gestión de incendios, agricultura de precisión, monitorización de pesca y estudios medioambientales.

- HALE (High Altitude Long Endurance): también conocidos como UAVs estratégicos, tienen alturas de vuelo superiores a 15000 metros, y rangos de operación de más

de 2000 Km. el enlace con la estación terrena puede usar comunicaciones por satélite, y su tiempo de operación puede ser de hasta 48 horas. Requieren los sistemas necesarios de prevencion de colisiones para compartir espacio aéreo con aviones tripulados antes de alcanzar grandes altitudes, superiores a las de los vuelos comerciales. Este tipo de UAV opera en altitudes muy superiores a las de los vuelos tripulados, con temperaturas extremas y vientos de alta velocidad. Pueden complementar o reemplazar a sistemas satelitales por ejemplo para observación de la Tierra o como repetidores de comunicaciones.

De los anteriores grupos, los más interesantes para trabajar con reconstrucción de estructura son los tres primeros, porque operan a bajas altitudes, son sistemas sencillos en comparación con los dos últimos y tienen la capacidad suficiente como para cargar una cámara para tomar las imágenes. El rango de altitud es una condición necesaria para que las imágenes sean lo suficientemente buenas como para reconstruir robustamente un escenario.

3.2. Campos de aplicación

Los UAV tienen un gran rango de aplicaciones, muchas de ellas todavía en desarrollo y que seguro tendrán una gran evolución en los próximos años. Las aplicaciones se pueden clasificar en dos grandes sectores, por un lado el militar, cuyo uso está plenamente extendido, habiéndose probado en diferentes conflictos o situaciones de emergencia, y por otro las aplicaciones civiles, con un mercado emergente y poco desarrollado, pero que en los próximos años tendrá un crecimiento importante según las previsiones [75] [76].

Los UAV militares pueden tener las siguientes aplicaciones:

- Blancos de tiro: usados para prácticas de tiro de sistemas de artillería antiaérea, por ejemplo el sistema ALBA del INTA[1]. Éstos son usados para entrenamiento.
- Reconocimiento: servicios de inteligencia o apoyo a unidades en tierra, proporcionando vídeo en tiempo real a través de cámaras con sensores de espectro visible e infrarrojo.
- Combate: portan armamento, generalmente conocidos como UCAV.
- Logística: transporte de carga, por ejemplo, ayuda médica.
- Investigación y desarrollo: usados para probar nuevas aplicaciones.

Dentro de las aplicaciones civiles, se encuentran las siguientes:

- Monitorización medioambiental, investigaciones y misiones científicas: pueden embarcar instrumentos de medida, ya sean sensores químicos, sensores de temperaturas, sensores de medida de velocidad del viento, sensores biológicos, etc., ofreciendo una alternativa al uso de sondas metereológicas o vuelos tripulados. Hasta la fecha se han usado para medir la evolución de huracanes, realizar mapas de icebergs y hielo en el mar, o estudiar la actividad volcánica. En este último caso, el riesgo para las tripulaciones es muy elevado a causa de los gases nocivos y las altas temperaturas. En concreto, un vehículo denominado Silver Fox se utilizó para monitorizar la temperatura de la lava del volcán St. Helen. cerca de Washington [77]. Las misiones

[1] http://www.inta.es/doc/programasaltatecnologia/avionesnotripulados/alba.pdf

Figura 3.3: RQ-4 Global Hawk

Ejemplo de un UAV militar tipo HALE empleado para la toma de imágenes como apoyo a servicios de inteligencia. Tiene una envergadura de $35m$ y también está siendo empleando en un proyecto civil de la NASA para observación de la Tierra[49]. Fuente: Wikipedia.

ciéntificas en ambientes extremos como en la Antártida son un claro ejemplo de aplicación. Estos aparatos permiten también medir o seguir la contaminación que se produzca en tierra, mar y aire. En el aire se pueden tomar medidas de la contaminación atmosférica en los alrededores de grandes ciudades, en el mar se pueden tomar imágenes de la distribución de vertidos de petróleo u otras sustancias químicas arrojadas por barcos, y en tierra se pueden usar imágenes para visualizar vertidos químicos.

- Lucha contra incendios: permite disponer de información en tiempo real sobre el avance del incendio, de forma que los servicios de coordinación dispongan de información actualizada sin poner en riesgo la vida de los pilotos de vuelos tripulados. En estos escenarios el denso humo dificulta la operación de aeronaves, haciendo muy peligroso su uso. Pueden incluso operar durante la noche, sin los problemas asociados a la falta de visibilidad que obliga a evitar los vuelos tripulados.

- Equipos de búsqueda y rescate: permite llevar a cabo operaciones de búsqueda de naufragos sustituyendo a los vehículos tripulados y reduciendo por tanto el riesgo al que se ven sometidos esas tripulaciones por la mala climatología o el cansancio. Los mismos vehículos podrían embarcar elementos como barcas hinchables o chalecos salvavidas que sirvan de ayuda hasta que lleguen los efectivos de rescate.

- Seguridad en fronteras y costas: pueden vigilar fronteras o costas durante largos periodos de tiempo sin conllevar los problemas de cansancio asociados a las tripulaciones. Incluso pueden operar en condiciones metereológicas desfavorables sin poner en riesgo la vida de los tripulantes.

- Vigilancia y monitorización de infraestructuras: tales como plantas nucleares, edifi-

cios críticos, carreteras, vías férreas, puertos, aeropuertos, o redes de distribución de energía. Pueden usarse para la construcción y el mantenimiento de las infraestructuras como por ejemplo detección de fisuras en puentes o presas, detección de problemas en vías férreas, obstáculos, o la entrada y salida de barcos en el puerto. El matenimiento de puentes, presas o tendidos eléctricos se hace hasta la fecha con personas, siendo actividades de alto riesgo. El uso de UAVs fiables evitaría fatalidades en este tipo de accidentes.

- Catástrofes: ayudan a gestionar situaciones de extrema peligrosidad sin poner en riesgo la vida de los tripulantes, tales como accidentes nucleares, incendios forestales, accidentes químicos, terremotos, maremotos o inundaciones entre otros.

- Urbanismo: inspecciones de terrenos urbanos, para ayudar a la administración a monitorizar el uso de suelo, a planificar eficientemente su uso y a detectar posibles expansiones ilegales.

- Tráfico: gestión de tráfico y aparcamientos, proporcionando una respuesta rápida a accidentes y otros incidentes relacionados, tarea que hoy día se realiza con vuelos tripulados.

- Agricultura: para proporcionar métodos de agricultura de precisión, usando imágenes de cultivos tomadas desde los UAV. Pueden usarse para diseminación de semillas, fumigación y control de humedad, temperatura o crecimiento entre otros parámetros. La primera certificación para este uso se ha realizado en Japón a un Yamaha Rmax [77]. Existen otros equipos de trabajo en este campo por ejemplo para el control de temperaturas en viñedos [29].

- Pesca: seguimiento de bancos de peces como apoyo a las flotas pesqueras. Hay sistemas comerciales ya implementados y operativos como por ejemplo el Fulmar de AeroVisión [20].

- Oceanografía: monitorizacion de accidentes en el mar, erosion de costas, movimiento de tierras, fluctuaciones de vegetacion, o erosion de acantilados. También pueden usarse para vigilancia de trafico marítimo.

- Telecomunicaciones y tecnologías de la información: como repetidores de telecomunicaciones, para aplicaciones de banda ancha o televisión, supliendo en parte la tarea que hacen a día de hoy los satélites, con la ventaja de tener un menor retardo de transmisión, y poder disponer de la última tecnología a bordo.

- Transporte: puede ser tanto de mercancías como de personas. Hay investigaciones al respecto para implementar aviones capaces de llevar carga[75]. El uso de UAV para transporte de personas está mas lejos de implementarse debido principalmente a la barrera psicologica asociada [77].

3.3. Ejemplos de UAVs

A continuación se comentan algunos ejemplos de UAV disponibles en el mercado o usados como demostradores tecnológicos en los que se puede aplicar este trabajo. Los aquí comentados son una pequeña muestra del gran número de vehículos de este tipo que

actualmente existen. El primero de los sistemas a comentar es el SIVA, un UAV desarrollado en el INTA[2], y del que se han usado algunos fotogramas para el presente proyecto. Se trata de un sistema de aeronaves no tripuladas para uso civil y militar en múltiples aplicaciones. La principal aplicación es su uso como vehículo de observación en tiempo real, al disponer de sensores ópticos en el espectro visible(CCD) e infrarrojo(FLIR), montados sobre plataforma giroestabilizada. El vehículo tiene un peso máximo de 300 kg. con capacidad de carga de 100 kg. y autonomía de 6.5 horas para una carga útil de 40 kg. Para su recuperación pueden usarse paracaídas y airbags o un tren de aterrizaje para recuperación en pista. El sistema de radioenlace permite la transmisión de vídeo en tiempo real con una tasa de transmisión de hasta 4Mbps y alcance de más de 100Km de la estación base. Puede operar a mayor distancia registrando las imágenes o transmitiéndolas vía satélite. La envergadura del avión es de 581 cm. Otro sistema interesante es Fulmar, un sistema

Figura 3.4: SIVA

Imagen del UAV SIVA desarrollado en el INTA. Fuente: INTA.

UAV completamente desarrollado por Aerovisión, una empresa de capital español [20]. Se trata de un avión de tipo *flying wing* cuya principal aplicación es servir de ayuda a barcos pesqueros en la búsqueda de bancos de atún. Se ha diseñado con el objetivo de dotarle de habilidad para amerizar aunque también dispone de una versión alternativa que puede ser operada desde tierra firme, y que se recoge por medio de una red. El lanzamiento se realiza desde una catapulta. Entre las principales capacidades, permite tomar y transmitir vídeo en tiempo real a una estación de control terrena así como imágenes infrarrojas. Su diseño se ha realizado con el objetivo de proporcionar buena estabilidad y resistencia además de larga capacidad de operación gracias a un bajo consumo. La cámara que incluye está giroestabilizada además de georreferenciada.

Otro sistema comercial interesante es el desarrollado por la empresa Schiebel GmbH, denominado Camcopter (R) S-100[3]. Es un helicóptero no tripulado autónomo diseñado de forma que se le pueden agregar varias cargas útiles acorde con los requisitos específicos del usuario. La plataforma puede usarse para aplicaciones militares y civiles. La computadora

[2]http://www.inta.es/doc/programasaltatecnologia/avionesnotripulados/siva.pdf

[3]http://www.schiebel.net/

Figura 3.5: Schiebel CAMCOPTER S-100
Ejemplo de helicóptero UAV operativo para diferentes aplicaciones, tanto civiles como militares.
Fuente: Schiebel, *http://www.schiebel.net*

de vuelo dispone de triple redundancia y dispone de módulos INS y GPS redundantes para proporcionar precisión de navegación y estabilidad. El fuselaje está hecho de fibra de carbono y es capaz de llevar una carga útil de 34 kg durante 6 horas, incluso pudiendo llevar una carga de 50 Kg debajo del rotor principal. Ejemplos de dispositivos embarcables son una cámara visible infrarroja, radar de apertura sintética (SAR), LIDAR, o radar de penetración en tierra (Ground Penetrating Radar). La misión se puede planificar usando un sistema de información geográfica (GIS), en el que se pueden asignar zonas de no-vuelo, zonas de peligro, etc.

En Holanda un UAV civil fue probado para monitorizar las vías ferroviarias cerca de la estación central de Amsterdam, siguiendo también carreteras, grupos de personas, o actividades en los canales, incluso volando dentro del espacio aéreo controlado por el aeropuerto Schirpol. En varios países como Francia y Hungría, se han usado UAVs para simular su uso en la extinción de incendios.

3.4. Futuro y tendencias

El futuro traerá grandes retos en el desarrollo de UAVs, que verán expandido su dominio y su capacidad de operación en múltiples tareas. A día de hoy existen un buen número de UAVs operativos, aunque el número de ellos usados en aplicaciones civiles es muy pequeño. La mayoría son usados como sistemas de inteligencia en los ejércitos de algunos países, como Estados Unidos, Alemania, España, Francia, Israel o Reino Unido. Algunos ejemplos de aplicaciones civiles plenamente operativas son los UAVs civiles que patrullan la frontera de EEUU con México, los helicópteros usados como fumigadores de cultivos en Japón o la vigilancia de costas oceánicas en Australia.

Mientras que los UAV militares tienen un uso completamente aceptado, los UAV civiles

todavía tienen un largo camino que recorrer para ver extendido su mercado. No cabe ninguna duda de que las aplicaciones civiles se extenderán en los proximos años, a pesar de que su uso todavía se enfrenta a problemas de aceptación por parte de usuarios, industria y autoridades [77]. Por un lado los potenciales clientes se resisten a implementar o usar aplicaciones ya que la industria no provee hasta la fecha vehículos certificados o porque la legislación actual no permite usarlos en el espacio aereo, donde operan aviones y helicópteros tripulados. Por otro lado las autoridades son reacias a crear estándares de operación y certificación, debido a que la industria no ha sido capaz de convencer de que las aplicaciones civiles son una realidad a corto plazo. Y por último, la industria aeroespacial es reacia a acelerar el desarrollo y producción de éstos al desconocer cuáles serán los estándares de diseño que deben cumplir, además del todavía limitado número de potenciales clientes.

Según Frost and Sullivan [77], el mercado de observación de la tierra es probablemente el más interesante en el uso potencial de UAV con un 37 % de cuota, seguido de las telecomunicaciones (13 %), control de costas (13 %), lucha contra el fuego (12 %), y el control de fronteras un (11 %). En menor medida tambíen se aplicarán a la monitorización de líneas electricas un (5 %), monitorización de oleoductos/gaseoductos (6 %), y por último la lucha contra el crimen un (3 %). Según este mismo análisis, sistemas tipo MALE empezarán a ser usados en los próximos años por los servicios de Guardacostas y Aduanas de muchos países europeos, en especial aquellos con mayor número de kilómetros de costa, como Reino Unido, Francia, España o Italia. La razón de que la observación de la tierra sea un campo de uso potencial es sin duda la calidad de las imágenes proporcionadas, con resolución de 15-20cm, además de una mayor frecuencia de toma de imagenes incluso con posibilidad de operación estatica. En el campo de las telecomunicaciones, las empresas aeroespaciales están trabajando en el reto de diseñar plataformas LTA adecuadas que superen los retos de altitud de 20000 metros y tiempo de operacion de semanas o meses. Lo más probable es que las primeras plataformas se establezcan en Asia, quizás en Japón, Corea del Sur o Singapur, incluso con un pequeño número de nodos se podría dar cobertura a toda África o con uno sólo a todo Oriente Medio.

Pese a que los UAS proveen un medio eficiente en coste, eficaz y fiable para diversas actividades, el uso de UAV civiles es mínimo debido a costes de seguridad y restricciones operacionales. La mayoría de los vehículos hoy existentes son adaptaciones de UAV militares cuyos datos de diseños no están disponibles por motivos de confidencialidad o no están diseñados para cumplir con especificaciones de tráfico aéreo. Por tanto, para asegurar su uso en aplicaciones civiles es necesario desarrollos tecnológicos que permitan asumir su aplicacion sin riesgos en zonas pobladas. Los UAV tienen diferentes demandas, como son seguridad, fiabilidad, peso reducido, bajo coste, bajo impacto ambiental, alta eficiencia y satisfacción del cliente. Para ello es muy importante la reducción del coste hora/vuelo.

Los UAV ofrecen unas posibilidades de negocio muy grandes, que pueden ser afrontadas dentro de los marcos de investigación, en primer lugar para reducir la dependencia tecnológica, sobre todo de EEUU e Israel, y en segundo lugar para crear un empleo basado en alta tecnología con trabajadores muy cualificados. Los UAVs proveerán a la industria aeronáutica de una oportunidad única de innovación que mantenga la competitividad global y retenga inversiones. Serán un generador de tecnología y un motor de crecimiento económico debido al alto grado tecnológico involucrado. Gracias además a que los UAV civiles requieren de un bajo capital de inversión y una barrera de entrada pequeña, que

va en función del tamaño del aparato, empujará a desarrollar nuevas aplicaciones y servicios potenciales, pudiendo ser realizados por empresas pequeñas o medianas, start-ups o spin-offs.

Hay principalmente dos retos a los que se enfrenta el futuro de los UAV, el primero es el grado de autonomía, y el segundo la resistencia. Estos retos se relacionan con la seguridad del espacio aereo operado por pilotos así como la fiabilidad y el coste. La autonomía se define como la capacidad de tomar decisiones sin la intervención humana y su grado depende del desarrollo de sensores, de equipos comunicaciones y de sistemas de trazado de rutas entre otros. En cierto modo se trata de dotar a las máquinas de inteligencia mediante sistemas de control jerárquico que permitan al aparato ser capaz de manejar diferentes situaciones sin poner en riesgo ni a otros aparatos ni a él mismo. La resistencia depende del tipo de vehículo del que se trate, de su tamaño, peso, carga de combustible, peso de la carga útil. Un ejemplo de UAV de larga duración, es el prototipo Helios de la NASA, proyecto basado en el uso de energía solar. Una demanda creciente de UAV civiles para misiones atraerá soluciones a estos retos y eventualmente dirigirán su incorporación al espacio aereo civil.

Una idea que también se ha propuesto es el uso de enjambres[4] de UAV que realicen de manera conjunta trabajos, incluso dependiendo de un UAV principal con posibilidades de operación dificiles de imitar con vuelos tripulados. En el capítulo 2 se presenta un trabajo en esta línea.

En relación con este trabajo fin de carrera el mayor uso de UAVs en aplicaciones civiles, permitirá crear oportunidades de negocio en el campo de la visión computacional y del tratamiento de imágenes aplicado al sector aeroespacial. El menor coste de operación, de mantenimiento y de toma de imágenes hará que diferentes organismos o empresas hagan uso de este tipo de dispositivos y que necesiten aplicaciones de tratamiento de imagenes para la gestión de la información contenida en los vídeos grabados desde estas plataformas.

[4]Traducción de la bibliografía en inglés, dónde se usa el término "swarm".

Capítulo 4

Fundamentos teóricos

4.1. Visión computacional

La visión computacional es una rama de la tecnología basada en la aplicación de teorías de visión a computadores para que sean capaces de extraer información a través de imágenes. Las imágenes pueden ser de diferenes tipos, ya sean fotografías o fotogramas de vídeo y pueden ser de diferente naturaleza, tomadas con cámaras convencionales, del espectro visible, o de infrarrojos. El campo de la visión computacional en general puede dividirse a su vez en subcampos totalmente diferentes como la reconstrucción de estructura, la detección de eventos, seguimiento de objetos en vídeo, reconocimiento de objetos, aprendizaje, indexado de imágenes, estimacion de movimiento o restauracion de imagen entre otros. Esta tecnología es un campo de trabajo muy reciente, que ha disfrutado de un gran desarrollo en los últimos años gracias a los avances en la capacidad de procesamiento de los ordenadores. Debido a la novedad de esta tecnología y a la publicación continua de nuevos métodos o algoritmos para abordar problemas de visión, no existe un manual estandar de cómo se debe proceder a trabajar con imágenes en este campo. En su lugar existen un gran conjunto de métodos varios que permiten resolver varias tareas de visión bien definidas. Los métodos son muy dependientes del problema a resolver, y salvo algunos casos, no se pueden extender a un amplio rango de aplicaciones. Muchos de los métodos y aplicaciones existentes hoy día estan en el estado de investigación básica, pero más y más metodos han encontrado su espacio como productos comerciales, donde constituyen parte de un sistema mayor que resuelve tareas complejas. La mayoría de las aplicaciones están basadas en software específico para resolver una tarea concreta, pero estan surgiendo métodos basados en el aprendizaje, por ejemplo en robótica. Existen multitud de aplicaciones de visión, que aquí no se detallarán. Simplemente se nombrarán algunas posibilidades de implementación, por ejemplo en sistemas industriales de control de calidad, medidas de posicion u orientacion de objetos para brazos robóticos, agricultura de precisión, aplicaciones militares para proveer múltiple información sobre el campo de combate, detección de personas o vehículos, vehículos autónomos (aéreos (UAV), sumergibles (UUV) o terrestres (UGV)) para navegación, posicionamiento, detección de obstáculos, lucha contra incendios, sisteas de apoyo o de alerta, aplicaciones de realidad aumentada, sistemas autónomos para aterrizaje de aviones e incluso también en exploración planetaria como en los Mars Exploration Rovers [40].

Los sistemas de visión computacional, dependen mucho de la aplicación para la que se diseñen. Algunas implementaciones son independientes y realizan una tarea determinada,

mientras que otras son subsistemas de diseños más grandes y complejos, como por ejemplo en los UAV con navegación visual. Por lo general hay un conjunto de funciones típicas que se pueden encontrar en la mayoría de sistemas de visión:

- Adquisicion de imágenes: la imagen se obtiene mediante sensores, que para el caso de imágenes del espectro visible (las interesantes para este trabajo), pueden ser de dos tipos, CCD o CMOS.
- Preprocesado: después de la adquisición de las imágenes en el sensor, estas pueden ser comprimidas o codificadas para su transmisión o almacenamiento en disco.
- Extracción de puntos característicos: se pueden extraer puntos interesantes de las imágenes para buscar correspondencias entre ellas. Otras opciones son la búsqueda de esquinas, regiones de interés, líneas o bordes.
- Detección de correspondencias: a partir de los puntos extraídos en la etapa anterior, se pueden buscar similaridades entre las imágenes.
- Segmentación: se puede divir la imagen o las imágenes en segmentos de interés para trabajar con un tamaño inferior de imagen.
- Tratamiento de los datos extraídos: por ejemplo para reconstruir la estructura a partir de las correspondencias, o la búsqueda de objetos en las imágenes.

Para el caso concreto de la reconstrucción de estructura a partir de imágenes, en los últimos años se han producido significativos avances en el campo. Algunos de los sistemas de reconstrucción de estructura existentes requieren procedimientos de calibración precisos con hardware específico para poder obtener medidas fiables, aunque la tendencia y la exigencia por parte de la comunidad es el desarrollo de métodos más flexibles que no necesiten de calibración previa o ésta se reduzca[54]. Otra de las tendencias actuales de este campo es el desarrollo de aplicaciones basadas en sistemas de adquision de bajo coste, como por ejemplo cámaras de fotos de gran público o incluso con teléfonos móviles conectados vía internet a un servidor que realice la reconstrucción. Gracias a esta gran evolucion, a día de hoy algunas de las técnicas de reconstruccion 3D no necesitan más que una cámara y un ordenador como dispositivos físicos.

Las técnicas de SfM se basan en la particularidad de las imágenes de permitir extraer información sobre el escenario 3D a falta de un factor de escala. La naturaleza de la formación de las imágenes se basa en la proyección de la escena sobre éstas, pasando de un espacio tridimensional u otro bidimensional por medio de una transformación proyectiva. La posición del punto 3D se puede recuperar a través de las posiciones de las proyecciones de este punto sobre las imágenes, siempre y cuando se usen dos o más vistas suficientemente separadas y con solape (que compartan objetos en la imagen). Las proyecciones vienen determinadas por la intersección de las líneas de vista formadas por las dos cámaras, los dos centros de cámara y los puntos correspondientes sobre la imagen (ver figura 4.1). En resumen se necesitan tres elementos para realizar este proceso:

- Puntos correspondientes en las imágenes (matches).
- Posicion relativa de las cámaras en la escena (dado por la matriz de la cámara).
- Relación entre los puntos correspondientes y las líneas de vista.

El tercer punto viene definido por el modelo de cámara. En este trabajo se supone que todas las cámaras siguen el modelo "pinhole". En adelante, los parámetros de calibración de la cámara se conocerán como parámetros intrínsecos de cámara, mientras que la posición y la orientación se conocen como parámetros extrínsecos. Esto se comentará con más detalle en la sección 4.3. Igual que los puntos tridimensionales pueden obtenerse con las cámaras, también puede realizarse el proceso inverso, estimando los parámetros de éstas a partir de puntos correspondientes conocidos y que se explicará con más detalle en este capítulo. Las cámaras pinhole ideales se supone que siguen un modelo de lineal. Las cámaras reales tendrán un coeficiente de distorsión radial y tangencial que se comentará también en el capítulo.

En el caso de Bundler, la reconstrucción se basa en emplear un par inicial de imágenes a partir del cual se estiman el resto de cámaras y puntos de manera relativa, tomando una de estas cámaras como referencia. Las cámaras obtenidas y los puntos reconstruidos son refinados para obtener unos valores mas precisos. No es necesario que todos los puntos correspondientes estén presentes a lo largo de toda la secuencia ya que los algoritmos de reconstrucción son capaces de manejar correctamente la oclusión. Como etapa adicional, la autocalibración se emplea únicamente en el caso de usar cámaras no calibradas. Un ejemplo del proceso de reconstrucción de estructura a grandes rasgos se puede ver en la figura 4.2.

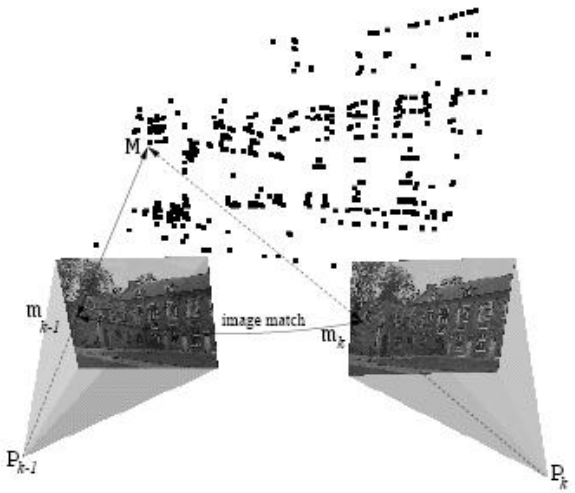

Figura 4.1: Detección de puntos correspondientes

Correspondencias entre dos cámaras para la reconstrucción de escenarios. Fuente: [54]

4.1.1. Algoritmos usados por Bundler

A continuación se enumeran los algoritmos de visión computacional empleados para la tarea de reconstrucción de estructura implementada en Bundler. Aquí se presenta únicamente una pequeña introducción. En el resto del capítulo se detallarán cada uno de los algoritmos a continuación:

- Detección de puntos característicos: SIFT para buscar puntos de interés en las distintas imágenes introducidas a la entrada.
- Detección de puntos correspondientes: ANN, para buscar posibles correspondencias.

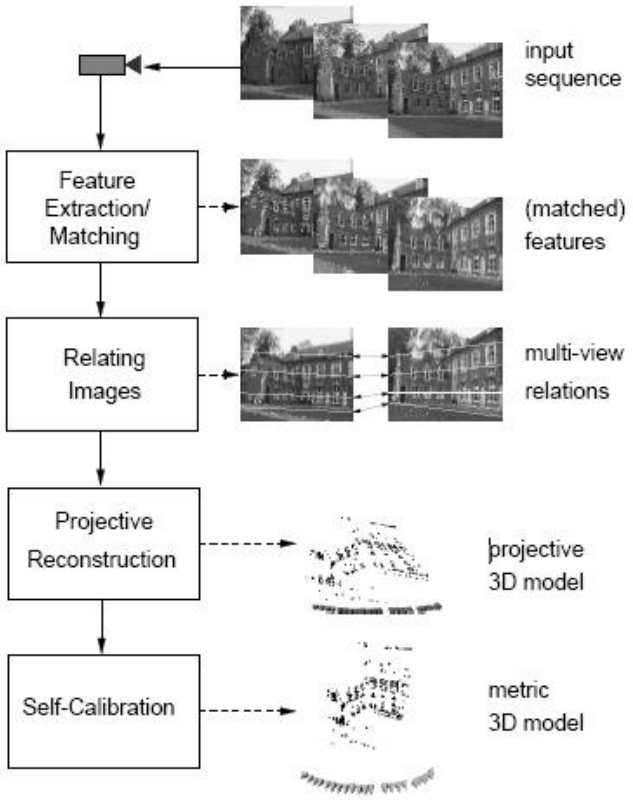

Figura 4.2: Secuencia de reconstrucción 3D de un escenario

Etapas del proceso de reconstrucción de escenarios a partir de imágenes. Fuente: [54]

- Reconstrucción de estructura: algoritmo de Níster, para recuperar la pose del primer par de cámaras, DLT para obtener el pose del resto de cámaras que se vayan añadiendo, SBA para reajustar el escenario y reducir el error de reproyección, Levenberg-Marquardt como algoritmo de minimización de errores no lineal basado en mínimos cuadrados no lineal, y RANSAC para elegir las soluciones más robustas dentro de un conjunto de valores de entrada.

4.2. Geometría proyectiva, afín y Euclídea

En esta sección se realiza una pequeña introducción a los tipos de geometría usados en visión computacional. Si se desea ampliar información se recomienda acudir a [25] donde se realiza un análisis más detallado sobre este tema.

La geometría proyectiva se usa en visión para modelar la proyección de una escena del espacio $\mathbb{R}^3$ en una imagen $\mathbb{R}^2$. Todos los puntos que estén en una línea de proyección se proyectarán en un punto común en la imagen. Por tanto, si se tiene un punto $P = (X, Y, Z)$ y otro $P' = (X', Y', Z')$, ambos serán equivalentes si existe un número real no nulo λ que cumpla la condición $P = \lambda P'$. De la misma forma, el punto P será equivalente al punto $P'' = (X/Z, Y/Z, 1)$ para $Z \neq 0$. En la representación proyectiva no se conserva el concepto de paralelismo. las líneas paralelas se cruzan en un punto y se puede calcular esta intersección. Se puede pensar por ejemplo en una fotografía con unas vías de ferrocaril. Éstas se cruzarán en un punto en el horizonte. Tampoco se conservan los conceptos de forma,

un círculo aparecerá como una elipse, ni los ángulos, distancias o ratios de distancias. Sin embargo si se conservará la condición de rectitud. Una línea recta seguirá siendo recta tras una transformación proyectiva. La transformación proyectiva es la la transformación fundamental de la geometría proyectiva. También se la conoce con el nombre de homografía. Si se asume un modelo de cámara pinhole (ver sección 4.3) y el movimiento (transformación) de las cámaras es una rotación pura, entonces se puede modelar la relación entre cámaras mediante una homografía. La geometría afín es una particularización de la geometría proyectiva, donde las transformaciones afines preservan los conceptos de paralelismo y ratios de distancia a lo largo de las líneas paralelas.

Por último, la geometría Euclídea es la más sencilla de entender. En ésta, un punto en el espacio Euclídeo $\mathbb{R}^n$ con n dimensiones tendrá una tupla de n números reales, que en el caso de este trabajo $n = 3$ por usar un espacio tridimensional. En este capítulo también se usarán espacios Euclídeos con mayor número de dimensiones, en concreto para la búsqueda de correspondencias basada en ANN (ver en apartado 4.6.1). Las transformaciones fundamentales son la traslación y la rotación. Tras una transformación de este tipo, las rectas paralelas seguirán siendo paralelas y se cruzarán en el infinito, se conservarán los ángulos, las formas de los objetos y también las distancias. Éstas últimas vendrán definidas por la distancia Euclídea, siendo ésta $d(P,Q) = \sqrt{\sum_{i=1}^{n}(P_i - Q_i)^2}$ con $P = (P_1, P_2, \cdots, P_n)$ y $Q = (Q_1, Q_2, \cdots, Q_n)$. En algún caso se podrá encontrar en la bibliografía el término espacio métrico, que no es más que una particularización del espacio Euclídeo.

4.3. Modelo de cámara

Una de las partes importantes de los fundamentos teóricos es el modelo de cámara empleado en este trabajo. Las cámaras proyectan los puntos 3D sobre imagenes 2D, es decir realizan una proyección que elimina la profundidad de la escena. El modelo de cámara empleado es una cámara pinhole. Luego se comentarán las características adicionales causadas por la distorsión radial de la lente de la cámara. El proceso se basa en elegir un centro de proyeccion de persepectiva en un punto C con un plano de proyección I (imagen). La mayoría de las cámaras convencionales se pueden aproximar por este modelo. Suponiendo el caso más simple, el centro de proyección se encuentra en el origen de las coordenadas mundo y el plano imagen se encuentra en $Z = 1$, el proceso de proyección puede modelarse como:

$$x = \frac{X}{Z} \tag{4.1}$$

$$y = \frac{Y}{Z} \tag{4.2}$$

Sea $(X, Y, Z, 1)$ el punto tridimensional y $(x, y, 1)$ el punto sobre la imagen, con la cámara en el origen de coordenadas, la proyección se modela como

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{4.3}$$

El eje óptico o eje principal, que se usará en la representacion de las cámaras en Orto3D (ver apéndice B), pasa a través del centro de proyección C y es ortogonal al plano imagen de proyección I. La intersección de este eje con el plano imagen define el punto principal

p. Este modelo no tiene en cuenta la calibración de la cámara.

Ahora teniendo en cuenta los parámetros extrínsecos e intrínsecos, y con f la longitud focal definida por la distancia entre el centro de la cámara Cy el plano imagen I. Con una cámara CCD, la focal en la vertical y en la horizontal así como su relación, depende del tamaño de la imagen, de la forma de los píxeles y de la posicion del chip CCD en la cámara. El modelo de cámara es entonces

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{4.4}$$

con s el factor de torsión (skew factor) para los píxeles no rectangulares. El píxel tiene tamaño $p_x \times p_y$ mm, en el resto de este trabajo se supondrá que se usan píxeles cuadrados con $p_x = p_y$, siendo $(c_x, c_y, 1)^T$ las coordenadas del punto principal, y f_x f_y las distancias focales para altura y anchura de pixel. Estos valores se supondrán iguales para el resto del trabajo, $f_x = f_y$, por trabajar con píxeles cuadrados. Estas distancias focales se calculan como $f_x = \frac{f}{p_x}$ y $f_y = \frac{f}{p_y}$. El factor de torsión skew se calcula como $s = tan(\alpha)\frac{f}{p_y}$. Esta matriz se conoce como la matriz de calibración K. Para las cámaras empleadas en este trabajo se puede asumir que $s = 0$. Además el punto principal se asume que es el centro de la imagen. Estas suposiciones se pueden tomar para simplificar el procedimiento de estimación iterativo del escenario durante la optimización, reduciendo el número de incógnitas. Para el caso de que se use el zoom de la cámara, la distancia focal varía. El modelo de la cámara quedaría entonces como el siguiente,

$$x_i = P_j X_i = K_j[R_j|t_j]X_i \tag{4.5}$$

que para el caso de que la cámara esta en el centro del sistema de referencia,

$$x_{ij} = P_j Xi = K_j[I|0]X_i \tag{4.6}$$

con x_i el punto i sobre la imagen j, X_i el punto tridimensional, P_j la matriz de proyección de la cámara, que a su vez contiene la matriz de calibración K_j, la matriz de rotación R_j y la matriz de traslación t_j.

A este modelo de cámara hay que caracterizarlo también por los parámetros de distorsión radial. También pudieran estar presentes los parámetros de distorsión tangencial, pero que para el presente trabajo se asume que son despreciables. Esta decisión se toma por el hecho de que Bundler ni los estima ni los emplea, usando únicamente los parámetros de distorsion radial. Este problema pertenece a lo que se conoce como aberraciones de la lente, efectos que pueden provocar grandes derivas en las medidas de las imágenes manifestándose como un desplazamiento de los puntos de forma radial con respecto al centro de la imagen. El efecto se pueden cancelar si se conocen los parámetros de modelado de la distorsión radial, que son aproximaciones no lineales a este efecto. De ahí que si lo puntos observados sin distorsion son (x, y), y (x_0, y_0) son los puntos con distorsión, y que el centro de coordenadas de la imagen es el punto principal de la imagen (se asumirá de ahora en adelante), la distorsión se puede modelar tal que

$$x = x_0(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \dots) \tag{4.7}$$

con el valor de r dado por

$$r = x_0^2 + y_0^2 \tag{4.8}$$

En este trabajo sólo se tendrán en cuenta los dos primeros coeficientes, es decir k_1 y k_2. Para poner un ejemplo visual del efecto de la distorsión radial, se presenta la figura 4.3.

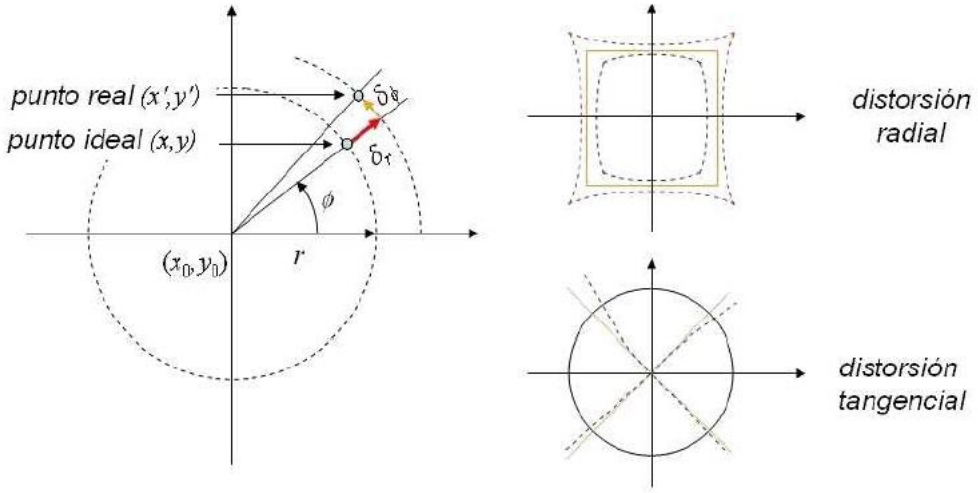

Figura 4.3: Distorsión radial

Las posiciones de los puntos ideales y reales serán diferentes debido al efecto de la distorsión radial que afecta a las segundas. También pudiera existir una distorsión tangencial, aunque en este trabajo se asume despreciable.

Fuente: [18]

4.4. Geometría epipolar

La geometría epipolar es la geometría intrínseca entre dos vistas, que es independiente de la estructura de la escena y únicamente depende de los parámetros internos de las cámaras y de la pose relativa. La matriz fundamental F encapsula la geometría intrínseca, siendo ésta una matriz 3×3 de rango 2, tal que siendo los puntos x en una imagen y x' en otra que dan lugar al punto tridimensional X, se satisface la condición epipolar:

$$x'^T F x = 0 \tag{4.9}$$

De la geometría epipolar se definen los siguientes conceptos, que además se presentan en las figuras 4.4 y 4.5:

- epipolo: es el punto de intersección de la línea que une los centros de cámaras con el plano imagen. El epipolo en una imagen es además, la posicion en esa vista de la cámara que genera la otra imagen.
- línea base (baseline): línea que une los centros de las cámaras.
- plano epipolar: es el plano conteniendo la línea base.
- línea epipolar: es la intersección del plano epipolar con el plano imagen. Todas las líneas epipolares de una imagen interseccionan en el epipolo. Estas líneas definen la correspondencia entre puntos de las imágenes (puntos correspondientes).

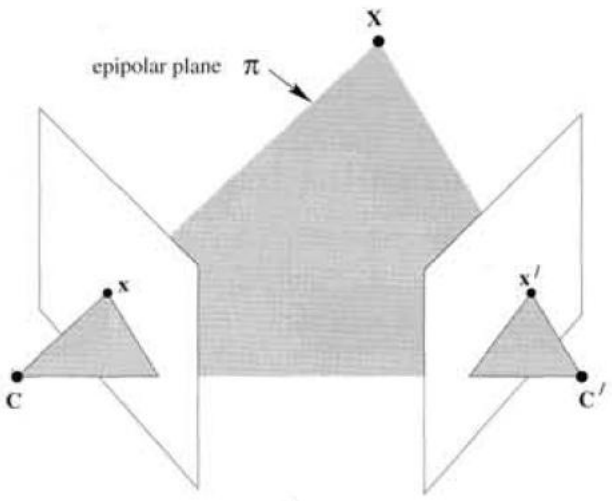

Figura 4.4: Geometría epipolar figura 1

En la imagen, los centros de las cámaras son C y C', los puntos correspondientes son x y x', el punto X es el punto tridimensional resultante de triangular los puntos correspondientes, y que se encuentra en el plano epipolar π. Fuente: [25]

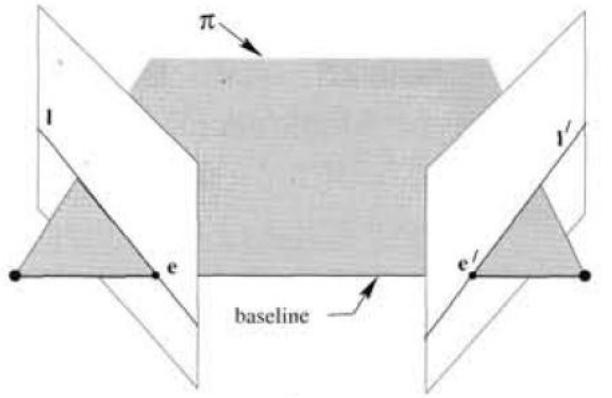

Figura 4.5: Geometría epipolar figura 2

En la imagen, la línea base (baseline) viene definida por la línea que une los centros de cámara y que intersecciona en los planos imagen en los puntos epipolares e y e'. La línea base se encuentra contenida en el plano epipolar π, y a su vez, los cortes formados por el plano π y los planos imágenes forman las líneas epipolares l y l'. Fuente: [25]

Del par de imágenes anterior, el punto x tendrá su punto correpondiente x' en la segunda imagen en la línea l' y análogamente, el punto x' tendrá su punto correspondiente x en la línea l. Esta condición se puede explotar para hayar los puntos correspondientes entre imágenes o descartar puntos que no cumplan esta condición. Como se verá, Orto3D calcula las líneas epipolares para servir de ayuda en el marcado manual de puntos correspondientes. Las líneas epipolares se calculan tal que,

$$l' = Fx \tag{4.10}$$

$$l = F^T x' \tag{4.11}$$

La matriz fundamental como se ha visto en la ecuación 4.9 debe cumplir la condición epipolar, que usando las líneas epipolares, se puede escribir como

$$x'^T F x = x'^T l' = 0 \tag{4.12}$$

Esta ecuación aporta un modo de caracterizar la matriz fundamental F únicamente a partir de correspondencias de imagen, necesitándo como mínimo 7 puntos para este cálculo [25].

Las propiedades de F son las siguientes:

- Es una matriz 3×3.
- Si F es la matriz que relaciona las cámaras (P, P'), la traspuesta F^T, relaciona las cámaras de forma opuesta (P', P).
- Los epipolos e, e' se pueden obtener tal que $Fe = 0$ y $F^T e'$.
- Debe cumplir que $det(F) = 0$, por tanto tien grado 2.
- F tiene 7 grados de libertad, ya que la escala es arbitraria y debe cumplir la condición $det(F) = 0$.

La matriz esencial es una especialización de la matriz fundamental cuando se dispone de las coordenadas normalizadas, asumiendo que las cámaras están calibradas. Tiene menos grados de libertad, y algunas propiedades adicionales a la matriz fundamental:

- Tiene 5 grados de libertad
- Una matriz 3×3 es esencial si y solo sí dos de los valores singulares son iguales y el tercero es cero

Nombrando E a la matriz esencial, con $P = K[R|t]$ la matriz de la cámara, y $x = PX = K[R|t]X$ la proyección del punto 3D sobre la imagen, cuando se conoce la matriz de la cámara, la relación entre el espacio imagen y el espacio tridimensional puede escribirse como $\widehat{x} = K^{-1}x = [R|t]X$. En ésta, $\widehat{x}$ es el punto imagen en coordenadas normalizadas, y $K^{-1}P = [R|t]$ es la cámara normalizada. Este cambio permite operar con la matriz esencial que se relaciona con los puntos correspondientes tal que

$$\widehat{x}'^T E \widehat{x} = 0 \tag{4.13}$$

que al sustituir los puntos normalizados por los puntos sin normalizar

$$x'^T K'^{-T} E K^{-1} x = 0 \tag{4.14}$$

y comparando con la relación entre la matriz fundamental y los puntos correspondientes $x'^T F x = 0$ se obtiene la relación entre las matrices fundamental y esencial,

$$E = K'^T F K \tag{4.15}$$

La matriz esencial permite la extracción de las poses de la cámara cuando la calibración de ésta es conocida, como se explica en la sección 4.7.1.

4.5. Detección de puntos característicos

La reconstrucción de estructura, al igual que el reconocimiento de objetos y otras tareas de visión computacional, se basan en la detección de características locales. Las características locales son puntos distintivos en la imagen tales como esquinas o regiones, que pueden encontrarse bajo diferentes condiciones de vista, como rotación o escala. En concreto, la detección de puntos se basa en la búsqueda de puntos interesantes en la imagen que además tengan una alta repetibilidad, es decir, que los puntos sean fáciles de detectar en múltiples imagenes. La extracción de característicos añade un descriptor para cada punto, de forma que puedan ser comparados y analizados. Los mas populares son los algoritmos de Harris, SIFT y SURF aunque existen otras variantes a partir de estos. En esta sección únicamente se hablará de SIFT, que es el algoritmo usado en Bundler. La detección de puntos se puede hacer guiada por restricciones adicionales si lo que se desea es reducir el tiempo de cómputo empleado en la búsqueda. Por ejemplo, se puede restringir el cálculo a una pequeña región de la imagen donde se conoce que va a localizarse el punto, siempre y cuando se asuman conocidos los parámetros de la cámara y una estimación del movimiento. Para el caso de Bundler, no hay ninguna restricción, no se conoce nada sobre las cámaras ni sobre el escenario, por lo que la busqueda se aplica a toda la imagen. Ésto hace que sea un cálculo pesado, aunque como se explica en [68], no es el cuello de botella de la aplicación.

4.5.1. SIFT

SIFT es un algoritmo de visión computacional que detecta y describe características locales en imágenes, siendo estas características invariantes a escala y rotación. El algoritmo fue desarrollado por David Lowe y publicado por primera vez en 1999. La librería de esta implementación está disponible de forma gratuita [35], aunque el código fuente de ésta no es abierto. También existen distribuciones abiertas similares como por ejemplo SIFT++ de Andrea Vedaldi[1]. El rango de aplicaciones para las que se ha usado este algoritmo es muy amplio, e incluye reconocimiento de objetos, navegación y mapeado en robots, superposición de imágenes o reconstrucción de estructura entre otras. El algoritmo está patentado a nombre de la Universidad de British Columbia[2]

El procedimiento que sigue el algoritmo es el siguiente:

1. Detección de extremos de la Diferencia de Gaussianos (DoG).
2. Localización de puntos característicos.
   (a) Determinar localización y escala.
   (b) Selección de puntos clave en base a medidas de estabilidad.
3. Asignación de orientaciones.
4. Cálculo del descriptor de punto característico.

En los siguientes párrafos se detallan cada una de estas partes.

[1] http://www.vlfeat.org/∼ vedaldi/code/siftpp.html

[2] US patent 6711293 *Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image.* Mar.23,2004

4.5.1.1. Diferencia de Gaussianos

La DoG es un procedimiento empleado para obtener los extremos en una imagen, que luego darán lugar a los puntos característicos. Para ello se crea inicialmente una pirámide espacio escala de diferencias entre convoluciones de una función Gaussiana con la imagen a diferentes escalas. A la salida de la diferencia de los Gaussianos se buscan los posibles puntos extremos locales. El espacio escala de una imagen se define como

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{4.16}$$

siendo la función Gaussiana

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{4.17}$$

Al calcular la diferencia de las imágenes a las que se ha aplicado la convolución con un Gaussiano, se obtiene la DoG,

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{4.18}$$

que es una aproximación al Laplaciando del Gaussiano normalizado a escala (LoG)

$$\nabla^2_{norm} L(x, y, \sigma) = \sigma^2 \nabla^2 L(x, y, \sigma) = \sigma^2 (L_{xx} + Lyy) \tag{4.19}$$

El LoG de una imagen da una respuesta positiva fuerte en las regiones oscuras y una respuesta negativa fuerte en las regiones brillantes. Esto se puede usar para encontrar puntos de interés en el espacio y en diferentes escalas. La siguiente relación se calcula a partir de la ecuación de difusión del calor:

$$\frac{\delta G}{\delta \sigma} = \sigma \nabla^2 G \tag{4.20}$$

que por aproximación de diferencias finitas,

$$\sigma \nabla^2 G(x, y, \sigma) = G_{xx} + G_{yy} \approx \frac{G(x, y, \sigma) - G(x, y, \sigma)}{k\sigma - \sigma} \tag{4.21}$$

tal que,

$$D(x, y, \sigma) \approx ((k-1)\sigma^2 \nabla^2 G(x, y, \sigma)) * I(x, y) \tag{4.22}$$

Después de calcular la DoG para cada escala, se aplica un umbral para encontrar máximos y mínimos. Todos los puntos por encima de este umbral se comparan con los 26 vecinos, 8 de la misma escala, 9 en la escala superior y 9 en la escala inferior, con el objeto de conocer si se trata de un mínimo o un máximo local, en cuyo caso se considera como posible punto característico.

4.5.1.2. Localización de un mapa de puntos

Para determinar con precisión la localización y escala de un punto característico candidato, se usa una función cuadrática 3D para encontrar el máximo interpolado. La expansión de Taylor cuadrática con origen en el punto candidato es:

$$D(x, y, \sigma) = D + \frac{\delta D^T}{\delta x} x + \frac{1}{2} x^T \frac{\delta^2 D}{\delta x^2} x \tag{4.23}$$

a partir de la cual, la localización interpolada del punto se encuentra igualando la derivada de $D(x)$ a cero, resultando en,

$$\widehat{x} = -\frac{\delta^2 D^{-1}}{\delta x^2}\frac{\delta D}{\delta x} \tag{4.24}$$

Las derivadas de D se aproximan usando diferencias entre puntos vecinos. si la localización resultante $\widehat{x}$ está más cercano a otro punto, entonces se ajusta la posición de comienzo y se repite la interpolación. El valor de $|D(\widehat{x})|$ en la localización interpolada se usa para eliminar puntos inestables con bajo contraste.

Los puntos característicos en bordes tienen respuestas altas a la DoG pero tienen una localización poco determinada y tienden a ser inestables. La curvatura principal de un punto borde es alta en la dirección longitudinal del borde pero pequeño en la transversal. Estas curvaturas se pueden calcular a partir del Hessiano de D (ver apéndice D) evaluado en la escala y localización actual de la forma

$$H = \begin{bmatrix} D_{xx} D_{xy} \\ D_{xy} D_{yy} \end{bmatrix} \tag{4.25}$$

De H se extraen dos autovalores, α y β, que son proporcionales a las curvaturas principales, que son la más grande y la más pequeña. Un ratio alto entre estos valores indica un borde. Únicamente se necesita el ratio para aceptar o descartar el punto, siendo suficiente con calcular:

$$Trace(H) = D_{xx} + D_{yy} = \alpha + \beta \tag{4.26}$$

$$Det(H) = D_{xx} D_{yy} - (D_{xy})^2 = \alpha\beta \tag{4.27}$$

$$\tag{4.28}$$

y

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r} \tag{4.29}$$

donde r es el ratio entre el autovalor más alto y más bajo ($\alpha = r\beta$). Para dar mayor estabilidad se aplica un umbral a r, siendo $r = 10$ el valor usado [36].

4.5.1.3. Asignación de orientación

A continuación, una vez determinada la localización y escala de los puntos, se calcula la para este punto la magnitud del gradiente

$$m(x,y) = \sqrt{(L(x+1,y,\sigma) - L(x-1,y,\Sigma))^2 + (L(x,y+1,\sigma) - L(x,y-1),\sigma)^2} \tag{4.30}$$

y la orientación,

$$\theta(x,y) = tan^{-1}(\frac{(L(x,y+1,\sigma) - L(x,y-1,\sigma))}{(L(x+1,y,\sigma) - L(x-1,y,\sigma))}) \tag{4.31}$$

a partir de los puntos que rodean al punto característico encontrado. Los gradientes se emplean para construir un histograma de orientación con 36 contenedores, donde cada gradiente se pesa por una magnitud y el Gaussiano con centro en el punto clave y 1,5 veces la escala actual. Los picos más altos del histograma se usan como orientaciones dominantes y se crean nuevos puntos clave para todas la orientaciones por encima del 80 % del máximo. Las orientaciones se calculan de forma precisa mediante el ajuste de parábolas al los picos de histograma.

4.5.1.4. Obtención del descriptor

El descriptor de puntos característicos se construye dividiendo la region alrededor del punto alineada con respecto a la orientación previamente calculada en 4×4 subregiones con 16×16 puntos de muestra. Para cada subregión se calcula un histograma de 8 contenedores para las orientaciones del gradiente, resultando en un vector de 128 componentes. El valor de este vector se normaliza a la longitud unidad para eliminar los efectos de cambios de iluminación lineal. Para reducir el efecto de la iluminacion no lineal y conseguir invarianza a ésta, se umbraliza el vector normalizado a un valor máximo de 0,2 y se renormaliza a la unidad de nuevo. La normalización a la unidad implica el uso de decimales, por lo que en la implementación de Lowe, esta normalización se multiplica por 256 para usar valores enteros, reduciendo así el tamaño de los ficheros de texto que almacenana los puntos característicos y mejorando la búsqueda de correspondencias.

4.6. Detección robusta de correspondencias

El paso siguiente es la búsqueda de puntos correspondientes. Los puntos correspondientes son puntos idénticos presentes en varias imágenes. Gracias al uso de algoritmos de detección como SIFT invariantes a escala, rotacion e iluminación, es posible encontrar estos puntos en varias imágenes bajo diferentes condiciones luminicas o de pose de cámara. En concreto, el algoritmo SIFT devuelve un vector que describe la region alrededor del punto característico encontrado. Este vector contiene 128 componentes, que describen la orientación de subregiones alrededor del punto. Cada vector en una imagen, se compara con los vectores de los puntos encontrados para otras imágenes de la reconstrucción. Si la distancia entre ambos vectores es inferior a un umbral definido, entonces se acepta como punto característico. La distancia empleada puede ser Euclídea o de Mahalanobis, aplicada a un espacio de dimension 128 (el número de componentes). Además la comparación se hace para todos los puntos de una imagen, que cuando la base de datos de imágenes es muy grande, hace que la busqueda bruta sea excesivamente pesada, con un coste computacional elevado. El uso de un método de búsqueda tradicional no es óptimo, por lo que es necesario encontrar otros métodos que permitan optimizar esta búsqueda. Un método típico de optimización de búsquedas clasico es el basado en kd-tree, sin embargo este método no produce una mejora significativa con respecto a la búsqueda sobre todo el conjunto de datos cuando la dimensión del problema es grande (128). Una primera solución es la basada en el algoritmo BBF, descrito en [36]. Sin embargo, Bundler emplea otra aproximación, denominada ANN, basada en la optimización por búsqueda de vecinos próximos aproximada, que se describe en el siguiente apartado. Después de la búsqueda, estima de forma robusta una matriz F que modele la relación entre pares de cámaras. Gracias a esta matriz, se pueden eliminar las correspondencias falsas (*outliers*) y quedarse con las verdaderas (*inliers*). Para ello emplea el método RANSAC para estimación robusta. La matriz F se puede calcular por varios métodos, aunque el utilizado por Bundler es el método de 8-puntos normalizado, que se explica también en este capítulo.

4.6.1. Approximated Nearest Neighbour

ANN es una librería desarrollada en C++ que implementa la búsqueda exacta y aproximada de vecinos más próximos (*nearest neighbors*) en espacios de varias dimensiones. Incluye además funciones para comprobar el resultado devuelto por el algoritmo así como para visualizar la estructura geométrica de los datos.

El algoritmo se basa en se buscan todos los puntos del conjunto P que están cercanos al punto q, con P un conjunto de puntos en d dimensiones, y un punto q una solicitud de búsqueda para el que se quieren obtener los más próximos. Gracias a este método de trabajo, sólo es necesario trabajar con los puntos más cercanos al punto, permitiendo que los subconjuntos de datos sean lo suficientemente pequeños como para trabajar en memoria primaria, y no en secundaria, optimizando el rendimiento. La distancia entre puntos puede definirse de varios modos, por ejemplo por medio de la distancia Euclídea.

Más en detalle, la aproximación consiste en preprocesar un conjunto de puntos en una estructura de datos desde la cual se pueden responder solicitudes de vecinos más proximos. El tiempo de ejecución o el espacio crecen exponencialmente en función de la dimensión, por lo que no son en todas ocasiones mejores que el método de fuerza bruta, salvo para pequeñas dimensiones. De ahí que para realmente optimizar la búsqueda en dimensiones grandes, se debe asumir un pequeño error, devolviendo un vecino que puede no ser el más próximo pero no estará mucho más lejos del punto solicitado que el vecino más próximo verdadero. ANN permite dar el resultado tanto exacto como aproximado.

En la búsqueda del vecino más próximo se da un conjunto de puntos P en d dimensiones reales del espacio $\mathbb{R}^d$ que construyen una estructura de datos tal que dada una solicitud de punto $q \in \mathbb{R}^d$, el punto más próximo a q se puede encontrar eficientemente. Dado $k \geqslant 1$ se busca devolver los k vecinos más próximos q en P. También se proporciona un error límite $\epsilon \geqslant 0$. Como valor de retorno, el algoritmo de búsqueda devuelve k puntos del conjunto P tal que se encuentre un punto i dentro del rango $1 \geqslant i \geqslant k$, con ratio $1 + \epsilon$ entre la distancia entre éste punto y el punto solicitado.

La librería soporta diferentes métodos para construir la estructura de datos. Para más información sobre éstas ir a [46]. Aquí solo se comenta la configuración por defecto, que usa la distancia Euclídea. Cada punto en d dimensiones se considera expresado como un d vector de coordenadas,

$$p = (p_0, p_1, p_2, \ldots, p_{d-1}) \tag{4.32}$$

La distancia entre dos puntos será la distancia Euclídea al cuadrado,

$$dist(p, q) = (\sum_{0 \leqslant i < d} (p_i - q_i)^2) \tag{4.33}$$

La razón de usar la distancia cuadrada en vez de la distancia real es ahorrar tiempo de cálculo en la raíz cuadrada y trabajar únicamente con enteros siempre y cuando los datos de entrada sean también enteros. La estructura de datos kd está basada en la subdivisión recursiva del espacio en regiones hiperrectangulares separadas llamadas células. Cada nodo del árbol se asocia con una región B de la caja, y está asociada con el conjunto de puntos de datos que caen dentro de esta caja. El nodo raíz está asociado con la caja que contiene todos los puntos. Considerando un nodo arbitrario en el árbol, si el número de puntos asociado con el nodo es mayor que una cantidad pequeña denominada tamaño de cubeta, se divide la caja en dos subcajas mediante un hiperplano ortogonal al eje que intersecciona esta caja. Hay varias reglas de partición que determinan cómo se selecciona el hiperplano y que se pueden leer en [46].

Las dos subcajas son dos hijos del nodo, y los puntos de la caja original se dividen entre las dos subcajas dependiendo de su posición en el hiperplano. Los puntos contenidos en

el hiperplano pueden asociarse con alguno de los dos hijos. Cuando el número de puntos asociados a la caja actual cae por debajo del tamaño de cubeta, el nodo resultante se declara como nodo hoja y se termina la división. El problema viene dado cuando los puntos están muy agrupados, donde se pueden tomar muchas divisiones para los puntos o las divisiones pueden resultar en cajas alargadas, las cuales son problemáticas cuando se ejecuta la búsqueda. Para este tipo de datos muy agrupados se incluye en ANN la implementación de estructura de datos en árbol bd. La diferencia con kd es que tiene una operación adicional de reducción. Para saber más sobre esta reducción acudir a [46].

ANN incluye dos métodos diferentes de búsqueda en árboles bd y kd, la búsqueda estándar y la búsqueda prioritaria. La primera, visita la celda en el orden de la estructura jerárquica del árbol de búsqueda. La segunda, visita cada celda en orden de distancia creciente al punto solicitado, y debería converger más rápidamente al vecino más próximo verdadero a costa de una mayor sobrecarga. En la referencia se comenta que cuando el límite de error es pequeño, la búsqueda estándar es ligeramente más rápida, y que cuando se usan límites mayores o terminaciones tempranas, la búsqueda prioritaria es superior.

4.6.2. Algoritmo de 8 puntos normalizado

El algoritmo de 8-puntos es el método más sencillo para calcular la matriz fundamental. Únicamente implica resolver un conjunto de ecuaciones lineales y tomar una solución de mínimos cuadrados. Para que el resultado sea bueno, es necesario normalizar los datos de entrada antes del calculo. La normalización es simplemente una transformación de traslación y escala de los puntos antes de aplicar las ecuaciones lineales. Gracias a esta normalización se estabiliza el resultado, y la sobrecarga de cálculo es minima.

Sean $n \geq 8$ correspondencias de imagen (x_i, x_i'), se estima la matriz fundamental con la condición epipolar $(x_i')^T F x_i = 0$ El algoritmo se puede resumir en cuatro etapas:

1. Normalización: transformación de traslación y escala para cada imagen de forma que el centroide de los puntos de referencia se encuentre en el origen de coordenadas y que la distancia media cuadrática de los puntos al origen sea igual a $sqrt2$. Si T es la matriz de transformación para una imagen y T' para la otra, los puntos son $\widehat{x}_i = Tx_i$ y $\widehat{x}_i' = T'x_i'$.

2. Calcular la matriz F mediante una solución lineal: busca la matriz $\widehat{F}'$ para los puntos correspondientes normalizados. Para ello se toma $\widehat{F}'$ como el vector singular correspondiente al menor valor singular de la descomposición SVD de $\widehat{A}$, que corresponde con la última columna de V^T. $\widehat{A}$ es la matriz formada al operar con la condición epipolar para los ocho puntos y poner los parámetros de la matriz fundamental en un vector columna:

$$Af = \begin{bmatrix} \widehat{x}_1'\widehat{x}_1 & \widehat{x}_1'\widehat{y}_1 & \widehat{x}_1' & \widehat{y}_1'\widehat{x}_1 & \widehat{y}_1' & \widehat{x}_1 & \widehat{y}_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \widehat{x}_8'\widehat{x}_8 & \widehat{x}_8'\widehat{y}_8 & \widehat{x}_8' & \widehat{y}_8'\widehat{x}_8 & \widehat{y}_8' & \widehat{x}_8 & \widehat{y}_8 & 1 \end{bmatrix} \tag{4.34}$$

con

$$f = (f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33}) \tag{4.35}$$

y las coordenadas de los puntos $\widehat{x}_i = (\widehat{x}_i, \widehat{y}_i, 1)$ y $\widehat{x}_i' = (\widehat{x}_i', \widehat{y}_i', 1)$.

El problema de esta matriz $\widehat{F}$ calculada es que no cumplirá la restricción de tener rango 2, de ahí que el siguiente paso sea reforzar esta condición.

3. Refuerzo de la restricción: se calcula con SVD la matriz $\widehat{F}$ por $\widehat{F}'$ tal que cumpla que $\det \widehat{F}' = 0$. Si $\widehat{F} = UDV^T$ con D una matriz diagonal tal que $D = diag(r, s, t)$, cumpliendo que $r \geq s \geq t$, la matriz $\widehat{F}' = Udiag(r, s, 0)V^T$. $\widehat{F}'$ minimiza la norma de Frobenius $\|F - F'\|$ cumpliendo que $\det \widehat{F}' = 0$.

4. Denormalización: deshace la normalización aplicada al inicio, obteniendo la matriz fundamental para los puntos (x_i, x_i'), con $F = T'^T \widehat{F}' T$.

4.6.3. RANSAC

RANSAC es un método iterativo para estimar los parámetros de un modelo matemático a partir de un conjunto de datos observados que contienen puntos anómalos. El resultado obtenido es válido bajo una cierta probabilidad, que crece conforme se realizan más iteraciones. Primeramente se asume que los datos se dividen en *inliers* que se ajustan al modelo y *outliers* que no. El propósito de este algoritmo es realizar una estimacion tobusta de los parámetros del modelo, encontrando parámetros de precisión incluso cuando se tiene una gran cantidad de outliers. El problema que presenta es que no dispone de un límite de iteraciones, necesitando un valor límite. Esto lleva a que puede que la solución encontrada con RANSAC tras la última iteración no sea la más óptima. El algoritmo se emplea para diversos problemas tanto de visión compuacional como de otros campos. En concreto la aplicación de RANSAC para la estimación robusta de la matriz F se puede resumir en:

1. Se selecciona una muestra de puntos correspondientes del conjunto encontrado, considerandolos como *inliers* hipotéticos.

2. Se encuentra un modelo válido para esta muestra, en este caso una matriz F para la muestra

3. Se buscan todos los puntos del conjunto completo que se ajustan al modelo encontrado, que se clasifican como *inliers*, y si no se ajustan se clasifican como *outliers*. A esta parte se le puede denominar votación del modelo. Aplicado a F se buscan todos los puntos correspondientes del conjunto que cumple la condicion epipolar para el modelo F calculado.

4. Se reestima el modelo F a partir de todos los inliers encontrados además de los inliers hipotéticos

5. Para cada iteración se compara el modelo calculado en la iteración con el modelo tomado como mejor modelo, calculando el error de la estimación relativo al modelo. Si el resultado es mejor, entonces se descarta el anterior y se guarda éste. En la primera iteración, este resultado es considerado como el mejor para inicializar el algoritmo. Al llegar a la condición de parada, tras i iteraciones, se toma el mejor modelo como la mejor matriz F, y se trabaja con este valor.

4.7. Reconstrucción de estructura

En el campo de la visión computacional se denomina reconstrucción de estructura o en inglés Structure from Motion (SfM) al proceso de recuperar los parámetros de las cámaras

y las coordenadas en el espacio 3D de un escenario a partir de puntos tomados en las imágenes. Los puntos de las imágenes son los puntos característicos encontrados en las etapas anteriores de la reconstrucción y que ya se han comentado en la sección 4.6. La reconstrucción se basa en el uso de varios algoritmos para poder recuperar los parámetros de las cámaras y la geometría del escenario. De los algoritmos existentes, aquí en esta sección se explican los implementados en Bundler, intentado seguir el orden en que son usados por Bundler. Para obtener los parámetros del primer par de cámaras es necesario usar el algoritmo de Níster, que obtiene estos datos a partir de cinco puntos, y un método de estimación robusto RANSAC usando el resto de correspondencias entra las dos imágenes del primer par. Conocidos los parámetros del par de cámaras, es posible obtener la posición 3D de los puntos correspondientes mediante una triangulación. El resto de cámaras se añaden al par inicial estimando su pose y calibración usando la DLT, y sus puntos mediante la triangulación de manera análoga al primer par. Todo este procedimiento, es optimizado por SBA que se ejecuta cada vez que se añade una nueva cámara al escenario[3].

4.7.1. Algoritmo de Nistér

El algoritmo de Níster es una solución algorítmica eficiente al problema de recuperación de pose a partir de cinco puntos en dos imágenes [50]. El resultado son las soluciones al movimiento relativo de la cámara con dos vistas calibradas. Para ello, el algoritmo calcula los coeficientes de un polinomio de décimo grado y encuentra sus raíces. El uso de una calibración intrínseca previa produce una mayor estabilidad y unicidad a la solución, aunque también aumenta la complejidad en el uso de aplicaciones de este tipo. Los algoritmos que no usan calibración interna son más flexibles y menos complejos pero a costa de perder estabilidad [50]. El procedimiento seguido para estimaciones iniciales sin calibración es aplicar a continuación un refinado iterativo para hacer que la estimación cumpla con las restricciones de calibración. Cuando se conocen los parametros intrínsecos a priori (calibración), el algoritmo de Níster es un buen método para obtener las poses de las cámaras.

Dados cinco puntos comunes a dos imágenes diferentes, se puede recuperar la moción relativa de la cámara y la posición tanto de puntos como cámaras, a excepción de la escala, que no puede ser recuperada únicamente a través de las imágenes. Sin embargo, dado que los puntos en las imágenes presentan ruido Gaussiano, si se desea una reconstrucción de estructura robusta, se necesita usar más de cinco puntos aplicando RANSAC. Éste método toma varias muestras de cinco puntos correspondientes. Cada muestra produce una hipótesis para la orientación relativa, que es votada por el resto de puntos correspondientes. Los puntos se dividiran en válidos (*inliers*) y anómalos (*outliers*). La hipótesis escogida será aquella con más puntos válidos.

El algoritmo se puede resumir en los siguientes pasos, que después se describirán con más detalle:

1. Extracción del espacio nulo derecho de una matriz 5×9 (cinco puntos y nueve incógnitas en E.

2. Expansión de las restricciones de E al cubo.

3. Eliminación de Gauss-Jordan de la matriz A de tamaño 9×20.

[3]También puede añadirse más de una cámara a la vez, como se explicará a continuación.

4. Expansión de los polinomios del determinante de las dos matrices 4×4 B y C seguidas de eliminación para obtener el polinomio de grado 10.

5. Obtener las raíces del polinomio de grado 10.

6. Obtener de R y t correspondientes a las raíces reales y buscar de las cuatro soluciones aquella que elimine las ambiguedades.

Sea i el número de puntos, y j el número de cámaras, sean $x_i^{(1)}$ y $x_i^{(2)}$ los puntos 2D en las imagenes 1 y 2, definido por tres valores, y X el punto 3D definido por cuatro valores, y la matriz de cámara $P_j = K_j[R_j|t_j]$ una matriz 3×4, la proyección de imagen es tal que $x_i^j \sim P_j X_i$ a excepción de la escala que es desconocida. t_j es el vector de traslación tal que

$$[t]_x = \begin{pmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{pmatrix} \tag{4.36}$$

con $[t]_x x = t \times x$ (ver apéndice D). Las matrices de las cámaras para cada vista se toman como $P_1 = K_1[I|0]$ y $P_2 = K_2[R_2|t_2]$, donde la matriz fundamental se puede calcular como

$$F = K_2^{-T}[t]_x R K_1^{-1} \tag{4.37}$$

siendo

$$x_2^T F x_1 = 0 \tag{4.38}$$

Si se conocen los valores de K_1 y K_2, las cámaras están calibradas, y dado que la matriz esencial es $E = [t]_x R$, suponiendo además que los puntos están multiplicados por las matrices de calibración, la ecuación anterior puede escribirse tal que

$$x_2^T E x_1 = 0 \tag{4.39}$$

Por las características de la matriz esencial, debe ser de rango 2 y con valores singulares no nulos iguales, cumpliendo además la siguiente:

$$EE^T E - \frac{1}{2} trace(EE^T)E = 0 \tag{4.40}$$

que permitirá obtener la matriz esencial y a partir de ésta, las matrices R, t y P.

Sean los puntos $x = (x_1, x_2, x_3)^T$ y $x' = (x'_1, x'_2, x'_3)^T$ y sea la matriz esencial,

$$E = \begin{pmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{pmatrix} \tag{4.41}$$

que operando da una ecuación similar a la presentada en el caso de la matriz fundamental

$$x_q E_q = 0 \tag{4.42}$$

donde

$$x_q \equiv \begin{pmatrix} x_1x'_1 + x_1x'_2 + x_1x'_3 + x_2x'_1 + x_2x'_2 + x_2x'_3 + x_3x'_1 + x_3x'_2 + x_3x'_3 \end{pmatrix}^T \tag{4.43}$$

y

$$E_q \equiv \begin{pmatrix} e_{11} + e_{12} + e_{13} + e_{21} + e_{22} + e_{23} + e_{31} + e_{32} + e_{33} \end{pmatrix}^T \tag{4.44}$$

Usando los vectores x_q^T para los cinco puntos, se obtiene una matriz 5×9.

La extracción del espacio nulo derecho se puede calcular a partir de cuatro vectores X_q, Y_q, Z_q y W_q, lo que se puede calcular con SVD o factorización QR (ver apéndice D), siendo la segunda más eficiente. Los vectores corresponden a cuatro matrices de 3×3, X, Y, Z, W, en la que la matriz esencial debe ser de la forma

$$E = xX + yY + zZ + wW \tag{4.45}$$

para cuatro escalares x,y,z y w definidos a un factor de escala común asumiendo que $w = 1$. Esta ecuación se inserta en la ecuación 4.40 y se aplica una eliminación de Gauss-Jordan basada en pivotaje parcial.

A	x^3	y^3	x^2y	xy^2	x^2z	y^2z	x^2	y^2	xyz	xy	xz^2	xz	x	yz^2	yz	y	1
(a)	1	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	[3]
(b)		1	·	·	·	·	·	·	·	·	·	·	·	·	·	·	[3]
(c)			1							·	·	·	·	·	·	·	[3]
(d)				1						·	·	·	·	·	·	·	[3]
(e)					1					·	·	·	·	·	·	·	[3]
(f)						1				·	·	·	·	·	·	·	[3]
(g)							1			L	·	·	·	M	N	O	[3]
(h)								1		P	Q	R	S	·	·	·	[3]
(i)									1	·	·	·	·	·	·	·	[3]

Tabla 4.1: Algoritmo de Níster, sistema de ecuaciones

$L, \ldots, S$ son los valores escalares y $[n]$ denota un polinomio de grado n en la variable z. Además se definen las siguientes ecuaciones:

$$(j) \equiv (e) - z(g) \tag{4.46}$$

$$(k) \equiv (f) - z(h) \tag{4.47}$$

$$(l) \equiv (d) - x(h) + P(c) + zQ(e) + R(e) + S(g) \tag{4.48}$$

$$(m) \equiv (c) - y(g) + L(d) + zM(f) + N(f) + O(h) \tag{4.49}$$

que dan las siguiente cinco ecuaciones para obtener z

$$(i) = xy[1] + x[2] + y[2] + [3] = 0 \tag{4.50}$$

$$(j) = xy[1] + x[3] + y[3] + [4] = 0 \tag{4.51}$$

$$(k) = xy[1] + x[3] + y[3] + [4] = 0 \tag{4.52}$$

$$(l) = xy[2] + x[3] + y[3] + [4] = 0 \tag{4.53}$$

$$(m) = xy[2] + x[3] + y[3] + [4] = 0 \tag{4.54}$$

Los valores de las ecuaciones se colocan en dos matrices 4×4 conteniendo polinomios en z.

B	xy	x	y	1	C	xy	x	y	1
(i)	[1]	[2]	[2]	[3]	(i)	[1]	[2]	[2]	[3]
(j)	[1]	[3]	[3]	[4]	(j)	[1]	[3]	[3]	[4]
(k)	[1]	[3]	[3]	[4]	(k)	[1]	[3]	[3]	[4]
(l)	[2]	[3]	[3]	[4]	(l)	[2]	[3]	[3]	[4]

Tabla 4.2: Algoritmo de Níster, polinomios

Como el valor del vector $[xy, x, y, 1]^T$ es un vector nulo a estas matrices, los polinomios del determinante deben ser 0. Si se nombran a los dos polinomios de determinante como (n) y (o), se puede tener un polinomio de grado diez de la siguiente forma:

$$(p) \equiv (n)o_{11} - (o)n_{11} \tag{4.55}$$

A partir de este polinomio se calculan las raíces mediante diferentes métodos, decritos en [50]. Para cada raíz z, se pueden encontrar las variables x,y, usando la ecuación B.

Los valores de R y t se pueden recuperar a partir de la matriz esencial, en base al teorema 2 de [50] que aquí se reproduce:

Sea $E \sim Udiag(1,1,0)V^T$ *la descomposición singular en valores de la matriz esencial, con* U *y* V *elegidos tal que* $det(U) > 0$ *y que* $det(V) > 0$*, entonces,* $t \sim t_u \equiv [u_1 3 u_2 3 u_3 3]^T$ *y* R *igual a* $R_a \equiv UDV^T$ *o* $R_b \equiv UD^TV^T$*, con* D*:*

$$D \equiv \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.56}$$

Cualquier combinación de las R y t obtenidas de acuerdo a la anterior condición satisfará la restricción epipolar, de las que sólo una será correcta. Para encontrar la correcta y evitar ambiguedades se asumirá que la primera cámara tiene matriz $P = [I|0]$ y que t tiene longitud unidad. La segunda cámara podrá tener 4 posibles soluciones:

1. $P_A \equiv [R_a|t_u]$
2. $P_B \equiv [R_b|-t_u]$
3. $P_C \equiv [R_b|t_u]$
4. $P_D \equiv [R_a|-t_u]$

De éstas, una de las soluciones corresponde a la configuración correcta, y otra a la configuración obtenida al girar la cámara 180º con respecto a la línea base. Las otras dos configuraciones corresponden a las reflexiones de la configuración verdadera y la girada 180. Para obtener la verdadera solución simplemente hay que buscar que los puntos se encuentren en frente de las cámaras y no a su espalda, restricción que se denomina *cheirality constraint* en la bibliografía de referencia [25]. Un punto es suficiente para resolver esta ambigüedad, en base a la triangulación de éste con la cámara de referencia $[I|0]$ y cada una de las cuatro posible soluciones. La correcta será la que cumpla la condición anterior.

Este algoritmo, publicado por Nistér en [50], fue luego mejorado en una nueva publicación también realizada en parte por Níster [72]. Según las referencias, este método mejorado

del algoritmo de los 5 puntos, es superior en la mayoría de las situaciones al algoritmo de los 5 puntos previamente desarrollado, así como a los basados en 6, 7 ó 8 puntos, además de ser más estable numéricamente [72]. Este algoritmo puede incluso, trabajar con escenas casi-planares o superficies casi-críticas. La diferencia introducida por este nuevo método es el uso de geometría algebraica calculando una base de Gröbner. La geometría algebraica trata de resolver conjuntos de ecuaciones polinomiales multivariable como la ecuación cúbica de E. El algoritmo de Níster forma parte de los denominados métodos directos, ya que su resultado no se basa en minimizar el error de reproyección en la imagen, lo que tiene la ventaja de que no quedan estancados en mínimos locales de la función de coste. La base de Gröbner se emplea para la matriz esencial generada por las restricciones de los polinomios, que construye una matriz 10×10 denominada matriz de acción cuyos autovalores y autovectores contienen las soluciones, siendo un método estándar en la resolución de polinomiales.

El algoritmo mejorado es idéntico al algoritmo hasta 4.45, que insertando E en la restricción cúbica de E, da lugar a diez ecuaciones polinomiales cúbicas en las variables (x, y, z), asumiendo que $w = 1$ (escala arbitraria) y que $x > y > z$ (notación GrLex, de graded reverse lexicographical monomial order), quedando el siguiente vector monomial:

$$P = [x^3, x^2y, x^2z, xy^2, xyz, xz^2, y^3, y^2z, yz^2, z^3, x^2, xy, xz, y^2, yz, z^2, x, y, z, 1]^T \quad (4.57)$$

Las diez ecuaciones pueden escribirse como,

$$MP = 0 \quad (4.58)$$

siendo M una matriz 10×20 con las filas linealente independientes.

Después de aplicar eliminación de Gauss-Jordan, el sistema puede escribirse como

$$[IB]X = 0 \quad (4.59)$$

donde I es una matriz identidad 10×10 y B una matriz 10×10. Este conjunto de ecuaciones deben ser una base de Gröbner. Los diez conjuntos de soluciones pueden obtenerse de los autovectores de A_x^t después de normalizar el último elemento a 1.

La implementación interna de Bundler incluye ambos métodos, es decir el algoritmo de los 5 puntos original y el mejorado, siendo el segundo el usado por defecto en la reconstrucción. Para usar el primero simplemente habría que cambiar un método por otro y recompilar el código. El problema de este método es que necesita trabajar con cámaras calibradas, en otras palabras, necesita la distancia focal. En 5.6 se verá que esta condición es la causa directa de que la estimación de la focal del primer par y del resto de las cámaras sea errónea en alguno de los escenarios de este trabajo. Esta sección es un pequeño resumen del algoritmo, para conocer en más detalle sobre éste, acudir a [72].

4.7.2. Levenberg-Marquardt

Levenberg-Marquardt es un algoritmo de ajuste de mínimos cuadrados no lineal que interpola entre el algoritmo de Gauss-Newton y el método de Gradient-Descent. Da una solución numérica al problema de minimizar una función, generalmente no lineal en un espacio de parámetros de la función. Éstos problemas de minimización aparecen sobre todo en ajuste de curvas por mínimos cuadrados y programación no lineal. Tiende a ser

más robusto que GNA, pero también es más lento. Puede encontrar una buena solución incluso cuando empieza lejos del mínimo. En otros casos el algoritmo converge sólo si la estimación inicial está de alguna forma cercana a la solución final. El algoritmo trata de minimizar la siguiente función de coste,

$$S(\beta) = \sum_{i=1}^{m} [y_i - f(x_i, \beta)]^2 \tag{4.60}$$

siendo el valor β el valor a optimizar en la función de coste. A continuación se comentan los diferentes parámetros de la optimización. Las distancias para cada uno de los puntos se pueden calcular como,

$$d(x_i, P_i X)^2 = (x_i' - x_i)^2 + (y_i' - y_i)^2 \tag{4.61}$$

donde x_i el punto medido por la cámara i, tal que

$$x_i = \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \tag{4.62}$$

y los valores de x_i' son los puntos reproyectados a partir de X_i,

$$P_i = K_i[R_i|t_i] \tag{4.63}$$

con X el punto 3D, tal que,

$$X = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{4.64}$$

y P_i la matriz de la cámara,

$$x_i' = P_i X \tag{4.65}$$

El proceso de minimación del error con n puntos y m cámaras consiste básicamente en obligar a que la suma del cuadrado de las desviaciones se haga mínima. Se trata además, de un procedimiento iterativo que necesita un valor inicial para β. Sean los valores a optimizar,

$$\beta = (X_0, \ldots, X_n, K_0, \ldots, K_m, R_0, \ldots, R_m, t_0, \ldots, t_m) \tag{4.66}$$

En cada nueva iteración, se sustituye en $f(x_i, \beta)$ el valor de β por un nuevo valor $\beta + \delta$ quedando como $f(x_i, \beta + \delta)$ con

$$\delta = (\delta_{X0}, \ldots, \delta_{Xn}, \delta_{K0}, \ldots, \delta_{Kn}, \delta_{R0}, \ldots, \delta_{Rn}, \delta_{t0}, \ldots, \delta_{tn}) \tag{4.67}$$

$$\beta + \delta = (X_0 + \delta_{X0}, \ldots, X_n + \delta_{Xn}, K_0 + \delta_{K0}, \ldots, R_0 + \delta_{R0}, \ldots, t_0 + \delta_{t0}, \ldots, t_n + \delta_{tn}) \tag{4.68}$$

El valor de la función después de añadir δ será:

$$S(\beta + \delta) = \sum_{i=1}^{m} [y_i - f(x_i, \beta + \delta)]^2 \tag{4.69}$$

donde el valor de f actualizado se puede aproximar como:

$$f(x_i, \beta + \delta) \approx f(x_i, \beta) + J_i \delta \tag{4.70}$$

Habiendo llegado hasta aquí, el objetivo ahora es minimizar la función de coste. El mínimo de una función se encuentra en el punto donde la primera derivada se hace cero. Sea $f(x_{min})$ el mínimo de la función $f(x)$ que se localiza en el punto $x = x_{min}$ para el que $\nabla f(x) = 0$, la búsqueda de este mínimo se puede escribir en función del Jacobiano (ver apéndice D),

$$\nabla f(x_{min}) = J^T(x_{min})r(x_{min}) = 0 \tag{4.71}$$

donde $r(x_{min})$ es el residual, que sustituyéndolo por $J\delta - \epsilon$), queda la siguiente ecuación,

$$J^T(J\delta - \epsilon) = 0 \tag{4.72}$$

y operando,

$$J^T J\delta = J^T\epsilon \tag{4.73}$$

esta ecuación se modifica para el algoritmo de Levenberg-Marquardt, quedando de la forma

$$N\delta = J^t\epsilon \tag{4.74}$$

donde N y ϵ se sustituyen,

$$(J^T J + \lambda diag(J^T J))\delta = J^T[y - f(\beta)] \tag{4.75}$$

$$\delta = (J^T J + \lambda diag(J^T J))^{-1} J^T(y - f(\beta)) \tag{4.76}$$

que aplicado a β queda

$$\beta_{i+1} = \beta_i + \delta_i = \beta_i + (J^T J + \lambda diag(J^T J))^{-1} J^T(y - f(\beta)) \tag{4.77}$$

Hecho esto, se evalúa el valor de $f^2(\beta + \delta_\beta)$,

- si $f^2(\beta + \delta_) \geq f^2(\beta)$ entonces se descarta el valor calculado $\beta + \delta$ y se incrementa λ por 10 para la siguiente iteración.
- si $f^2(\beta + \delta) < f^2(\beta)$ entonces se acepta el valor de $\beta + \delta_\beta$, se actualiza la solución y se decrementa λ por 10 para la siguiente iteración.

La iteración continúa hasta que se llega a la condición de parada definida por la tolerancia(tol), de forma que termina si

$$f^2(\beta + \delta) < tol \tag{4.78}$$

Un ejemplo de función de coste en el caso de tener m cámaras y n puntos sería

$$f(x, x') = \sum_{i=0}^{n-1}\sum_{j=0}^{m-1} d(x_i j, P_{ij}Xi)^2 = \sum_{i=1}^{m} d(x_{ij}, x'_{ij})^2 \tag{4.79}$$

con x los puntos medidos en las imágenes, y x' los puntos reproyectados.

4.7.3. Direct Linear Transform

La transformada lineal directa permite obtener el mapeado entre las coordenadas de la imagen y las coordenadas 3D de ese punto. Sea un conjunto de j cámaras definidas por P_j y un conjunto de i puntos 3D X_i que vienen definidos por puntos en las imágenes denominados x_{ij}, la relación viene definida por

$$x_{ij} = P_j X_i \tag{4.80}$$

El resultado de operar con la DLT es una matriz 3×4 de proyección P, que conocidos los puntos sobre las imágenes x_{ij} y los puntos 3D X_i se puede estimar. Para ello se usa la DLT cuyo algoritmo requiere un conjunto de puntos, al menos 6, ya que la matriz P tiene 11 grados de libertad. En realidad bastaría con $5\frac{1}{2}$ puntos pero por razones obvias el mínimo es 6.

Si se opera con la ecuación $x_{ij} = P_j X_i$ se obtiene la siguiente ecuación:

$$\begin{pmatrix} 0^T & -w_i X_i^T & y_i X_i^T \\ w_i X i^T & 0^T & -x_i X_i T \\ -y_i X_i^T & x_i X_i^T & 0^T \end{pmatrix} \begin{pmatrix} p^1 \\ p^2 \\ p^3 \end{pmatrix} \tag{4.81}$$

cuyas filas son linealmente dependientes, quedando el sistema de ecuaciones como:

$$\begin{pmatrix} 0^T & -w_i X_i^T & y_i X_i^T \\ w_i X i^T & 0^T & -x_i X_i T \end{pmatrix} \begin{pmatrix} p^1 \\ p^2 \\ p^3 \end{pmatrix} \tag{4.82}$$

De un conjunto de i correspondencias de puntos, se obtiene una matriz $2n \times 12$ que se denominará A. La matriz de proyección se podrá obtener por tanto de resolver el conjunto de ecuaciones

$$Ap = 0 \tag{4.83}$$

Aplicado al caso de Bundler, se conocen los puntos en la imagen y los puntos 3D del escenario calculados a partir del primer par de cámaras, calculados con el algoritmo de Nistér. Los puntos sobre las imágenes se conocen a partir de los puntos obtenidos con SIFT y las correspondencias encontradas. Los puntos 3D se toman de los puntos del primar par de imágenes que comparten con una tercera imagen de la que se quiere estimar la matriz de la cámara. Esta matriz de la cámara es la que se estima con el algoritmo DLT y que se usará como inicializacion del proceo de bundle adjustment. Estos puntos que comparten imágenes se dice que están en una trayectoria (*track* en la documentación de Bundler). A partir de la estimación de esta matriz de cámara P se triangulan los puntos X para estas tres cámaras. Así se hace sucesivamente para el resto de las cámaras del escenario. Para cada punto visible se calcula el producto vectorial $x \times (PX) = 0$, quedando las siguientes ecuaciones:

$$x(p^{3T}X) - (p^{1T}X) = 0 \tag{4.84}$$
$$y(p^{3T}X) - (p^{2T}X) = 0 \tag{4.85}$$
$$x(p^{2T}X) - (p^{1T}X) = 0 \tag{4.86}$$
$$\tag{4.87}$$

Combinando las ecuaciones tal que $Ap = 0$, con la condición de que $||p|| = 1$, se puede obtener el valor de p descomponiendo A con SVD y tomando la última columna de V^T. El resultado del algoritmo DLT para homografías depende del marco de coordenadas en el

que se expresen los puntos, Por esta razón es aconsejable normalizar los datos de entrada antes de aplicar el algoritmo DLT. La normalización mejora la precisión de los resultados y asegura que el algoritmo sea invariante a elecciones arbitrarias de escala y sistema de coordenadas. La normalización se realiza de al siguiente forma:

- Las coordenadas de los puntos se transladan de forma que el centroide se sitúe sobre el origen de coordenadas.
- Las coordenadas se escalan de forma que la distancia media al origen sea $\sqrt{3}$, quedando el punto medio 3D en las coordenadas $(1, 1, 1, 1)$.

Después de normalizar se aplica la DLT, y luego de aplicar la DLT es necesario denormalizar los datos. Para ello se debe revertir la transformada de normalizacion. Sea T la transformada de similaridad que normaliza las coordenadas del espacio imagen, U una segunda transformada de similaridad para normalizar las coordeandas del espacio mundo y P' la matriz de proyeccion normalizada, la matriz de proyección denormalizada será entonces

$$P = T_{-1}P'U \tag{4.88}$$

Como la matriz P tiene 12 valores y por tanto 11 grados de libertad, se necesitan 11 ecuaciones para resolver p, con p el vector que contiene las entradas de la matriz P de la forma

$$p = (p_{11}, p_{12}, p_{13}, \ldots, p_{34}) \tag{4.89}$$

Se requieren como mínimo $5\frac{1}{2}$ correspondencias ya que cada una da lugar a dos ecuaciones. La solución se obtiene resolviendo $Ap = 0$, donde A es una matriz 11×12. Si los datos contienen errores debido al ruido en las coordenadas de los puntos y se tienen más de 6 puntos, no habrá una solución exacta a $Ap = 0$ con $||p|| = 1$. La solución se puede obtener minimizando un error algebraico o geométrico. Para el algebraico, consiste en minimizar $||Ap||$ sujeto a que $||p|| = 1$ por medio de un algoritmo de mínimos cuadrados no iterativo, en el caso de este trabajo, el algoritmo Levenberg-Marquardt que se explica en 4.7.2.

4.7.4. Triangulación

Cuando se conocen con certeza los parámetros de las cámaras, y no hay ruido en la selección de puntos en la imagen, la tarea de buscar el punto 3D a partir de puntos correspondientes sobre las imagenes es sencilla. Los problemas aparecen cuando hay presencia de ruido, siendo críticos en reconstrucciones proyectivas y afines en las que no hay información métrica sobre el objeto espacio. De ahí que el objetivo sea buscar un método de triangulación invariante a transformaciones proyectivas de espacio.

Cuando los rayos no coinciden, elegir el punto medio de la perpendicular común a ambos rayos no es el mejor método. Éste se conoce como método del punto medio (*midpoint method* en la bibliografía en inglés), y no da resultados óptimos. Además, la condición de perpendicularidad para el espacio proyectivo no tiene sentido alguno, de ahí que sea interesante buscar otros métodos. La minimización en el espacio 3D tampoco es interesante ni apropiada, ya que no funciona con la reconstrucción proyectiva, donde no son válidos los conceptos de distancia y perpendicularidad, además de que un método basado en reducir el error en 3D no sería invariante a proyección[26]. Por estas razones, el método elegido para triangular y minimizar el error se basa en calcular la distancia entre el punto medido

en cada imagen y los puntos reproyectados a partir de una estimación inicial del punto 3D. Para ello se usa la siguiente función de coste (ejemplo para dos imágenes):

$$d(u, \widehat{u})^2 + d(u', \widehat{u}')^2 \tag{4.90}$$

con $d(,)$ la distancia Euclídea y $\widehat{u}$, $\widehat{u}'$ los valores más cercanos a los puntos correctos, de forma que cumplan la condición epipolar, $\widehat{u}' F \widehat{u} = 0$. Conocidos con certeza $\widehat{u}$ y $\widehat{u}'$, se puede calcular el punto 3D.

Existen varios métodos para la triangulación, siendo uno de ellos el basado en una triangulación lineal seguida de una triangulación no lineal. Éste es el método implementado en Bundler y en el prototipo Orto3D. La triangulación lineal sirve para encontrar una estimación inicial que será minimizada por medio de una triangulación no lineal iterativa que busca minimizar el error de triangulación. La minimización se basa en el algoritmo de Levenberg-Marquardt(descrito en el apartado 4.7.2). El método lineal consiste en triangular dados dos puntos $x = (x, y, 1)^T$, $x' = (x', y', 1)$, correspondientes entre sí, tomados de dos imágenes (o más aunque este ejemplo es con dos), y conocidas también las matrices de las cámaras P, P' que dan lugar al punto tridimensional X mediante la relación $x = PX$ y $x' = P'X$. Esta relación define seis ecuaciones lineales en $X = (X, Y, Z, 1)^T$, que se reducen a cuatro linealmente independientes. En forma matricial queda como

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{4.91}$$

Aplicando las matrices de la misma forma al punto de la otra cámara y operando las matrices anteriores y eliminando las ecuaciones linealmente dependientes, se obtiene que:

$$p_3^T X x = p_1^T X \tag{4.92}$$
$$p_3^T X y = p_2^T X \tag{4.93}$$
$$p_3^T X x' = p_1^T X \tag{4.94}$$
$$p_3^T X y' = p_2^T X \tag{4.95}$$

$$(p_{31}X + p_{32}Y + p_{33}Z + p_{34})x = p_{11}X + p_{12}Y + p_{13}Z + p_{14} \tag{4.96}$$
$$(p_{31}X + p_{32}Y + p_{33}Z + p_{34})y = p_{21}X + p_{22}Y + p_{23}Z + p_{24} \tag{4.97}$$
$$(p'_{31}X + p'_{32}Y + p'_{33}Z + p'_{34})x' = p'_{11}X + p'_{12}Y + p'_{13}Z + p'_{14} \tag{4.98}$$
$$(p'_{31}X + p'_{32}Y + p'_{33}Z + p'_{34})y' = p'_{21}X + p'_{22}Y + p'_{23}Z + p'_{24} \tag{4.99}$$

Expresando estas ecuaciones de forma matricial, queda:

$$\begin{pmatrix} (p_{31}x - p_{11}) & (p_{32}x - p_{12}) & (p_{33}x - p_{13}) & (p_{34}x - p_{14}) \\ (p_{31}y - p_{21}) & (p_{32}y - p_{22}) & (p_{33}y - p_{23}) & (p_{34}y - p_{24}) \\ (p'_{31}x' - p'_{11}) & (p'_{32}x' - p'_{12}) & (p'_{33}x' - p'_{13}) & (p'_{34}x' - p'_{14}) \\ (p'_{31}y' - p'_{21}) & (p'_{32}y' - p'_{22}) & (p'_{33}y' - p'_{23}) & (p'_{34}y' - p'_{24}) \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{4.100}$$

Si a la matriz izquierda se le denomina A y al vector de la derecha q, el problema queda en resolver

$$Aq = 0 \tag{4.101}$$

que al tener los puntos error Gaussiano, los rayos no se cruzarán en un punto en el espacio y el producto Aq no será nulo. Por tanto el objetivo de la triangulación es buscar un valor q que minimice $||Aq||$ sujeto a que $||q|| = 1$, lo cual se puede obtener utilizando SVD de forma que:

$$[U, D, V] = SVD(A) \tag{4.102}$$

donde el valor de q se obtiene de la última columna de V. Este valor de q sirve como estimación inicial para el algoritmo no lineal iterativo que se aplica a continuación. Sea el punto $X = q$ el punto estimado, el método consiste en tomar la ecuación 4.90 función de coste, donde se conoce el valor de los puntos medidos sobre las imágenes (x,x'), y los puntos reproyectados se pueden obtener a partir de P, P' y X. El método no lineal usando LMA permite reducir este error en base a modificar el valor de X con nuevas estimaciones hasta llegar a una condición de parada definida por una tolerancia de error o un límite de iteraciones. Para más detalles sobre cómo se aplica LMA, ir a la subsección 4.7.2.

La triangulación es un paso indispensable para la reconstrucción de estructura, ya que es necesario usarla para añadir cualquier punto al escenario a partir de las medidas en las imágenes. Hay condiciones en los que la triangulación puede tener problemas para recuperar correctamente los puntos debido a determinados movimientos de cámara o resolución de imagen, o ruido [25]. El ruido provoca que las coordenadas de los puntos no sean correctas y que por tanto los rayos generados no intersecciones. Si el ruido es grande, encontrar el punto correcto será más difícil. Con la resolución de la imagen, también ocurre un problema similar, ya que a menor resolución menor precisión en la medida del escenario. No es lo mismo una imagen de 1200×1000 que otra de 600×500 para el mismo objeto, obviamente en la segunda será más difícil visualizar los detalles, lo que ocurre en el marcado manual de puntos usado en Orto3D y que se verá más adelante. Esto puede hacer que el error final de triangulación sea grande cuando no se seleccionan los píxeles correctos. Por último, el movimiento de cámara influye en la percepción de perspectiva que se tiene del escenario. Un movimiento de cámara tipo hacia delante provocará que la línea base sea pequeña con respecto al escenaro si este último es grande, y que cualquier punto triangulado pueda tener un error elevado. Esto se analiza en la subsección 4.7.6. Para la explicación se han usado únicamente dos cámaras, aunque conceptualmente es igual para más de dos. Hay que tener en cuenta que la triangulación manual en el prototipo Ort3D usa tres cámaras y Bundler todas las vistas disponibles, como mínimo dos.

4.7.5. Sparse Bundle Adjustment

Los algoritmos de Bundle Adjustment buscan la optimización de los resultados de triangulación y pose de cámara mediante el ajuste de rayos. Cada cámara tiene 11 grados de libertad y cada punto 3D otros 3 grados. Una reconstrucción con n puntos y m cámaras requiere la minimización de $3n + 11m$ parámetros. Al usar LMA, se deben factorizar matrices $(3n + 11m) * (3n + 11m)$ lo cual es muy costoso en términos de cálculo y en ocasiones imposible. Para evitar esto se pueden tomar las siguientes dos aproximaciones:

1. Usar subconjuntos de la reconstrucción. Se basa en dividir la optimización en problemas más pequeños con un subconjunto de las vistas y los puntos. Cada subconjunto será ajustado por separado y el resultado final será fusionado.

2. Usar métodos poco densos (sparse methods). Se basa en explotar las características poco densas de los datos utilizados.

SBA implementa la segunda opción. Los datos utilizados tienen una estructura en bloque de poca densidad. Las ecuaciones de proyección tendrán un gran número de entradas nulas, ya que los puntos no serán vistos por todas las cámaras. De hecho lo más probable es que la mayoría de las correspondencias entre imágenes se encuentren entre fotogramas consecutivos en secuencias de vídeo por ejemplo. Esto permite un ahorro computacional significativo. La optimización se basa en refinar la moción y la estructura mediante la minimización con LMA del error de reproyección entre los puntos medidos y los puntos reproyectados.

Considerando la situación en la cual un conjunto de puntos 3D X_j es visto por un conjunto de cámaras con matrices P_i, y x_{ij} las coordenadas del punto j visto por la cámara i, el objetivo es resolver el siguiente problema de reconstrucción: dado el conjunto de coordenadas imagen x_{ij}, encontrar el conjunto de matrices de cámara P_i y los puntos X_j tal que $x_{ij} = P_i X_j$.

Al tratarse de medidas con ruido Gaussiano, las ecuaciones $x_{ij} = P_i X_j$, no se cumplirán, siendo necesario buscar una solución que tenga un error mínimo. El algoritmo por tanto consiste en estimar la matriz de proyección P'_i y los puntos 3D X'_j que se proyectan en los puntos imagen x'_{ij} a partir de una estimación inicial de sus valores. La función de coste a optimizar es entonces la distancia entre el punto medido y el punto reproyectado $x'_{ij} = P'_i X'_j$, por lo que el problema consiste en minimizar esta función de coste

$$\sum_{i,j} d(P'^i X'_j, x^i_j)^2 \tag{4.103}$$

El código está implementado en C/C++ y ha sido desarrollado bajo licencia GPL [32]. En su implementación se ha puesto especial énfasis en flexibilidad y eficiencia en funcionamiento, usando BLAS[4], LAPACK[5] para las operaciones de álgebra lineal y MINPACK[6] para resolución de sistemas no lineales y minimizaciones de mínimos cuadrados.

El método de bundle adjustment se aplica como punto final en cualquier algoritmo de reconstrucción. En concreto SBA es una implentación interesante por su tolerancia a falta de datos y eficiencia. Sin embargo tiene el problema de necesitar una buena inicialización para no caer en un mínimo local. El principal inconveniente de la optimización con SBA es que puede ser un problema muy grande de minimización debido al gran número de parámetros involucrados.

4.7.6. Efectos de la línea base y del movimiento

La línea base es un parámetro importante para la reconstrucción de estructura. Este valor es determinante para el resultado final. Por un lado, una línea base pequeña con respecto al escenario significa que las cámaras están muy cerca y que se pierde la profundidad. Por otro, una línea base grande permite recuperar mejor la profundidad, pero puede que las imágenes no tengan solape y que el algoritmo de búsqueda de correspondencias sea incapaz de encontrar puntos correspondientes entre las cámaras. Por tanto habrá un

[4] http://www.netlib.org/blas/
[5] http://www.netlib.org/lapack/
[6] http://www.netlib.org/minpack/

rango de valores de línea de base que permitan una reconstrucción robusta con bajo error.

Para estudiar teóricamente el efecto de la línea base, se suponen aquí únicamente dos cámaras, en concreto el par inicial de la reconstrucción. Los escenarios obtenidos tienen múltiples cámaras, pero realizar un estudio teórico sobre varias cámaras de este problema es muy complejo. Por ello esta sección se reduce a hablar del primer par, además de por simplificación porque es la relación de cámaras más importante, ya que todo el escenario se reconstruirá y optimizará a partir éstas. Un error en esta primera estimación se arrastrá durante toda la reconstrucción, por ello interesa que la línea base entre estas dos cámaras sea adecuada [69]. La teoría aquí presentada está basada en el estudio teórico sobre este tema descrito en [18].

Para simplificar el modelo de estudio se suponen dos cámaras P_1 y P_2, con movimiento

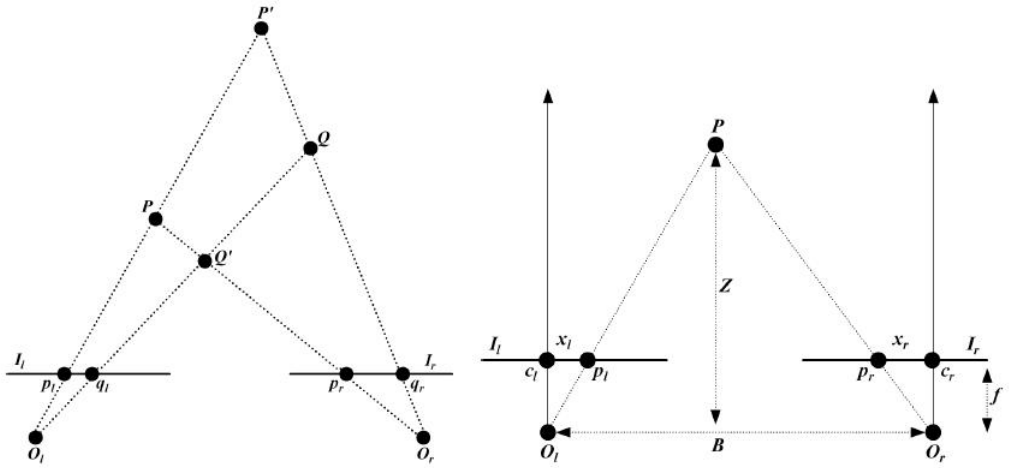

Figura 4.6: Línea base entre dos cámaras

En la figura se pueden ver dos modelos para tratar el tema de la línea base. Por un lado el esquema de la izquierda presenta un ejemplo visual de cómo un desplazamiento en la posición de los puntos correspondientes afecta a la triangulación. Este modelo es una generalización del error presente en las medidas de los puntos. Por otro lado, a la derecha se prsenta el escenario del que se extraen las ecuaciones usadas aquí para modelar el problema.

Fuente: [18]

lateral al objeto (dirección de movimiento longitudinal con respecto al objeto), y con la misma focal f. Sean las imágenes I_1 e I_2, el punto 3D X viene indicado por la triangulación con los puntos x_1 y x_2 de cada cámara. Sean también C_1 y C_2 los centros de las cámaras, y Z la profundidad del punto, o lo que es lo mismo, la distancia entre la línea base y el punto X. En la figura 4.6 se puede ver el escenario explicado. Haciendo un tratamiento descriptivo del problema, no se obtendrá el resultado X sino otro, debido al error introducido por el ruido en la medida de los puntos. Para el caso de estas dos cámaras, si debido al ruido en la imagen o al error de marcado manual, en vez de tomar los puntos x_1, x_2 se tomasen los puntos q_1, q_2, al triangular se obtendría el punto Q, que estará más o menos alejado de la solución verdadera X en función de los puntos elegidos q_1, q_2 y de la línea base definida por los centros de las cámaras. Si el error estuviese en una de las cámaras pero en la otra no, el resultado sería P' para la combinación (x_1,q_2), y Q' para (q_1,x_2). Como se puede ver el efecto del ruido provoca que el punto 3D reconstruido se aleje del punto correcto. Aplicando un análisis similar, suponiendo que los puntos son

ideales, tomando como d_1 la distancia entre x_1 y el punto c_1 que es el punto principal, y d_2 como la distancia entre x_2 y el punto c_2. Los triangulos similares definidos por (X, x_1, x_2) y (X, C_1, C_2), permiten hacer un analisis de la relación entre línea base y profundidad, con las siguientes ecuaciones:

$$\frac{B + d_1 - d_2}{Z - f} = \frac{B}{Z} \Rightarrow Z = f\frac{B}{d} \tag{4.104}$$

siendo d la disparidad entre cámaras $d = d_2 - d_1$. Poniendo estos valores en píxeles, suponiendo que

$$u_i = \frac{x_i}{l_{CCD}} + u_0 \tag{4.105}$$

con (u, v) las coordenadas del punto en píxels, y (u_0, v_0) el punto principal en píxeles, y l_{CCD} las dimensiones en mm de un píxel en el chip CCD de la cámara, con el píxel cuadrado, tenemos

$$Z = f\frac{B}{d} = f\frac{B}{x_2 - x_1} = f\frac{B}{(u_2 - u_0)d_2 - (u_1 - u_0)d_1} = \frac{f}{d_x}\frac{B}{u_2 - u_1} = f_x\frac{B}{d_u} \tag{4.106}$$

si el tamaño en píxels de una imagen tomada con las cámaras es (W, H), la mínima distancia medible viene definida por la máxima disparidad $d_u = W - 1$, y la máxima profundidad por la mínima disparidad, $d_u = 1$. La precisión de la medida viene dada entonces por la diferencia entre la distancia medida al tomar un píxel y la medida al tomar un píxel adyacente.

$$\Delta Z_i = Z_i - Z_{i-1} = f_x B(\frac{1}{d_{ui} - 1} - \frac{1}{d_{ui}}) = f_x B\frac{1}{d_{ui}^2 - d_{ui}} \tag{4.107}$$

El rango máximo de distancia viene dado por la focal y la línea base tal que:

$$Z = f_x B \tag{4.108}$$

Tras estas ecuaciones queda claro que la línea base influye directamente en la precisión de la profundidad y en el rango máximo. Conforme aumenta la profundidad de los objetos, disminuye la precisión, efecto que se ve incrementado cuando la separación entre cámaras es pequeña. Para que el rango máximo de distancia sea más grande, es necesario incrementar la línea base tomando cámaras más separadas.

El error de profundidad Z_e se puede calcular como:

$$\Delta Z_e|_B = \frac{\delta Z}{\delta B}\Delta B = \frac{f_x}{d_u}\Delta B \tag{4.109}$$

$$\Delta Z_e|_d = \frac{\delta Z}{\delta d_u}\Delta d = -f_x\frac{B}{d_u^2}\Delta d_u \tag{4.110}$$

que queda como:

$$\Delta Z_e^2 = \Delta Z_e|_B^2 + \Delta Z_e|_d^2 = \frac{f_x^2}{d_u^2}\Delta B^2 + \frac{f_x^2 B^2}{d_u^4}\Delta d_u^2 = \ [\Delta B^2 + \frac{B^2}{d_u^2}\Delta d^2] \tag{4.111}$$

Con respecto al movimiento de la cámara (son secuencias de vídeo), éste tendrá un claro efecto sobre la reconstrucción. Se definen dos tipos de movimientos críticos, el movimiento hacia delante (*forward motion*) y el movimiendo lateral (*sideways*). El movimiento hacia delante es un tipo de movimiento crítico porque tiene mas difícil la recuperación de la

estructura y de la profundidad, más si cabe si se trata de objetos enfocados a muy larga distancia. El movimiento lateral aporta una mayor facilidad a la hora de reconstruir, ya que en algunos casos se harán incluso vuelos cercanos, donde se puede tomar una buena relacion de la profundidad a partir de las imágenes. El problema viene cuando las escenas son casi-afines, es decir, imágenes de vídeo donde las cámaras están muy lejanas al escenario. En éstas se pierde la profundidad debido a que ésta es mucho menor que la distancia entre la cámara y el objeto medido. Para ilustrar estos problemas se presenta la figura 4.7.

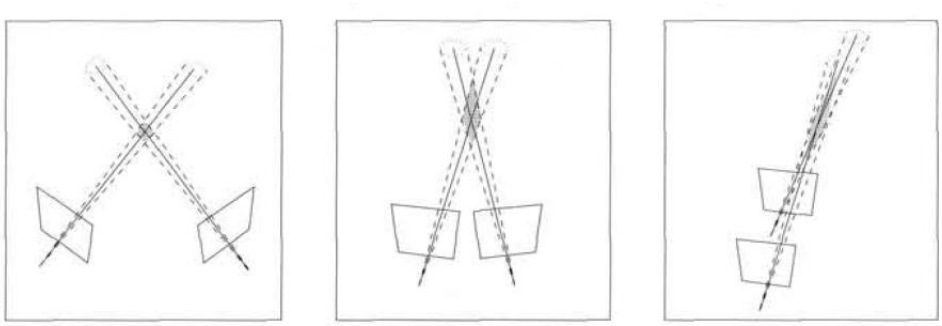

Figura 4.7: Movimientos críticos de cámara

En la figura se pueden ver tres ejemplos de movimientos de cámara. El escenario de la izquierda es el mejor de los tres presentados. En éste la separación entre cámaras es adecuada, y el movimiento de la cámara es en círculo alrededor del objeto visualizado. El segundo escenario presenta un movimiento lateral, con una línea de base pequeña. La zona de incertidumbre es mayor que para el primer escenario. Por último el escenario más critico es el tercero, donde se presenta un movimiento hacia delante, con un error de profundidad elevado en la zona de incertidumbre. Fuente: [25]

Capítulo 5

Reconstrucción de estructura con UAV

En este capítulo se presentan los resultados obtenidos con Bundler al usar vídeos aéreos. El capítulo se divide en varias secciones. La primera sección comenta Bundler, la herramienta de reconstrucción de estructura usada en este trabajo. Se habla sobre la implementación interna de la aplicación, en base a los algoritmos explicados en el capítulo 4. La segunda sección presenta Orto3D e incluye una descripción de su implementación. Para ambas aplicaciones se añaden unos manuales en los apéndices A y B. La tercera sección contiene un comentario general de los resultados, los problemas y las soluciones propuestas. En primer lugar, se presentan los resultados iniciales obtenidos con los vídeos tomados con el SIVA, seguido de un apartado donde se explica el filtro introducido para eliminar el efecto producido por los caracteres sobreimpresos. A continuación, se presentan los resultados con otras imágenes, por un lado, fotografías tomadas con una cámara convencional, y por otro, fotogramas tomados de una secuencia de vídeo aéreo de documentales de televisión. Los últimos apartados comentan en detalle los problemas de estimación de focales, estimación de distorsión radial y los errores de SfM, que son los principales problemas encontrados. La cuarta sección presenta cuatro de los escenarios. En concreto se presenta un escenario con imágenes del SIVA, otro con fotografías llamado POLITECNICA, y dos escenarios con fotogramas de vídeo de documentales, TORRE_PICASSO_AZCA_MOD_1 y MECO_MOD_1. Éstos resultados se presentan con un mayor detalle que en la sección 5.3, donde se realiza un análisis general. Gracias al análisis de los resultados obtenidos, se define una secuencia de trabajo recomendada para operar con Bundler y con Orto3D, que se expone en la quinta sección. Las dos últimas secciones comentan las tareas realizadas y las herramientas empleadas. Para las tareas se realiza un comentario general sobre éstas, dividiendo el trabajo en fases bien diferenciadas. Las herramientas se dividen en tres subgrupos, el hardware empleado, el software y las fotografías y vídeos fuente empleados en las reconstrucciones.

5.1. Bundler

Bundler es una aplicación de reconstrucción de estructura (SfM) para colecciones de imágenes desordenadas tomadas de internet, con código fuente desarrollado en C y C++, y que usa algoritmos y paquetes de software del estado del arte. Bundler toma un conjunto de correspondencias entre las imágenes de entrada y produce una reconstrucción 3D del escenario, tanto de las cámaras como de la geometría del escenario. El proceso se realiza

incrementalmente, con un subconjunto del total de imágenes en cada iteración, optimizando el resultado con una versión modificada del Sparse Bundle Adjustment de Lourakis y Argyros [33][34]. La aplicación se ha desarrollado para Linux y Windows, necesitando en este segundo caso un emulador de Linux denominado Cygwin. También se puede usar con Mac OS, pero conviene consultar al autor sobre el procedimiento de instalación, ya que no aparece publicado en la web de Bundler[67]. Una versión inicial de esta aplicación fue usada en el trabajo Photo Tourism publicado en [67][71]. La distribución Bundler incluye implementaciones de varios algoritmos aplicados a la visión computacional, como la búsqueda aproximada de vecinos cercanos (ANN), SBA o el método de Níster que se comentan en el capítulo 4. Bundler se distribuye bajo licencia GNU GPL[1].

La forma más sencilla de usar Bundler es ejecutarlo desde el script *RunBundler.sh* incluido en la distribución. Este script llama a todos los ejecutables de la secuencia de reconstrucción de estructura de Bundler, necesitando como entrada únicamente imágenes JPEG. Previo a ejecutar la aplicación, es necesario realizar algunas modificaciones en los ficheros script de la aplicación, que se detallan en el manual adjunto en el apéndice A. La reconstrucción de estructura es el último paso de una secuencia que comienza con la extracción de información sobre las focales, sigue con la detección de puntos característicos en la imagen, y continúa con la búsqueda de correspondencias antes de proceder a iniciar el algoritmo SfM. Cada uno de estos pasos se han descrito de forma general en el capítulo 4 sobre los fundamentos teóricos. En este capítulo se concreta cómo Bundler usa estos algoritmos para obtener la reconstrucción.

5.1.1. Implementación de la aplicación

El procedimiento completo de Bundler puede describirse en los siguientes puntos:

- Inicialización del programa
    1. Creación de una lista de imágenes JPEG para la reconstrucción, guardada en *list_tmp.txt*.
    2. Extracción de focales de las cabeceras EXIF y creación de una lista de imágenes JPEG con las focales extraídas, guardada en *list.txt*.
- Detección de puntos característicos y búsqueda de correspondencias
    1. Búsqueda con SIFT de puntos característicos en las imágenes de la carpeta.
    2. Búsqueda de correspondencias basada en ANN para los pares de imágenes.
    3. Estimación robusta con RANSAC y el algoritmo de 8 puntos de una matriz fundamental candidata F que relacione cada par de imágenes.
    4. Optimización mediante Levenberg-Marquardt de la matriz fundamental previamente estimada.
    5. Organización de las correspondencias en trayectorias (*tracks*)[2], eliminando aquellas que sean inconsistentes.

[1] http://www.gnu.org/licenses/gpl-3.0.txt

[2] La documentación de Bundler usa el término *track* para referirse a un punto común a varias imágenes, que se podría traducir como trayectoria en referencia a la trayectoria de este punto en la secuencia de fotogramas.

- Reconstrucción de estructura (Structure from Motion).

    1. Calcular una homografía para cada par de imágenes de forma robusta con RANSAC para encontrar el par con mayor número de outliers a la homografía estimada.
    2. Estimación de los parámetros de cámara del par con mayor número de outliers y al menos 100 correspondencias usando el algoritmo de 5 puntos de Níster.
    3. Triangulación de puntos a partir de las poses de cámara.
    4. Optimización de bundle adjustment con SBA para el par inicial y sus puntos.
    5. Añadir una o más cámaras a la reconstrucción estimando su pose con la DLT a partir de puntos 3D obtenidos con las cámaras ya reconstruidas y comunes a éstas y a las nuevas cámaras.
    6. Optimización con SBA usando las nuevas cámaras.
    7. Añadir nuevos puntos vistos por las cámaras añadidas y por al menos otra de las cámaras ya reconstruidas.
    8. Optimizar con SBA y repetir los cuatro puntos anteriores hasta que se hayan añadido todas las cámaras y puntos.

5.1.1.1. Inicialización del programa

Antes de ejecutar el programa es necesario realizar algunas operaciones. En primer lugar es necesario poner en una carpeta todas las imágenes JPEG que se desea emplear en la reconstrucción. Luego es necesario copiar el fichero *RunBundler.sh* en la carpeta donde están las imágenes. Se trata de un script que inicializa la reconstrucción y llama a cada uno de los pasos de la secuencia de Bundler. La primera operación que realiza es la creación de una lista de imágenes, que guarda en *list_tmp.txt*. Para cada una de estas imágenes lee su cabecera EXIF con una llamada al ejecutable *jhead.exe* y extrae las focales con *extract_focal.pl*. Las focales extraidas se guardan en *list.txt*, que se usarán más adelante en la reconstrucción. La lista de imágenes JPEG sirve para ejecutar SIFT con cada una de las imágenes como se comenta en el siguiente apartado.

5.1.1.2. Detección de puntos y correspondencias

El primer paso que realiza Bundler es la búsqueda de puntos característicos mediante el algoritmo SIFT. Éste es invariante a cambios de escala, orientación, distorsión afín y parcialmente invariante a iluminación. Es además robusto a ruido. Estas características hacen que su uso sea interesante para aplicaciones de reconstrucción de estructura. Como se comenta en 4.5.1, SIFT da como resultado un descriptor local para cada punto característico. Para cada par de imágenes se realiza una búsqueda de correspondencias entre puntos característicos usando la búsqueda aproximada de vecinos próximos (ver en apartado 4.6.1). Si se buscan las correspondencias en dos imágenes I y J, se crea un árbol kd para los descriptores de los característicos de I. A continuación, para cada característico en I se busca el vecino más próximo en J usando el método de búsqueda prioritaria, limitando la solicitud a la búsqueda en un máximo de 200 contenedores en el árbol. Para encontrar las correspondencias, se encuentran los dos vecinos más próximos con distancias d_1 y d_2, para aceptar el primero si $\frac{d_1}{d_2} < 0{,}6$ Si hay más de un punto en I que se corresponde con el mismo característico en J, entonces se elimina esta correspondencia. Después de encontrar

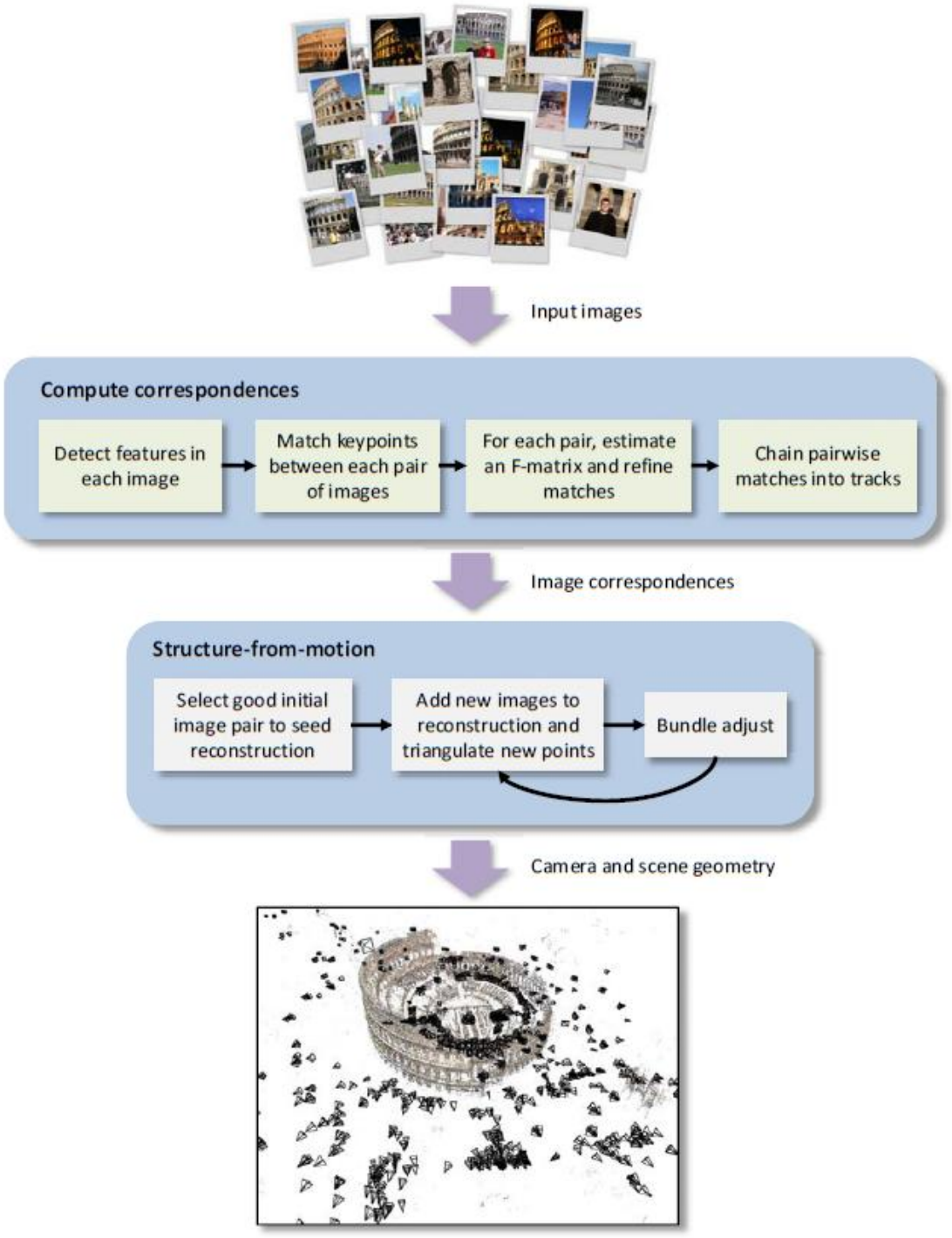

Figura 5.1: Secuencia de ejecución de Bundler

Bundler toma un conjunto de imágenes como entrada, busca las correspondencias entre ellas en base a puntos característicos detectados con SIFT, y a partir de éstas realiza la reconstrucción de estructura. Fuente: [68]

las correspondencias para el par de imágenes (I, J), se estima una matriz fundamental F de forma robusta usando RANSAC. Para cada iteración RANSAC, se calcula F con el algoritmo de 8 puntos normalizado, y se vota el resultado con el resto de puntos correspon-

dientes encontrados. El ratio de *outliers*(puntos falsos) para la votación se ajusta al umbral de 0,6 % de la dimensión máxima de la imagen ($0{,}006max(ancho\ imagen, alto\ imagen)$). El resultado final con RANSAC es la matriz F con el menor número de $outliers$ de todas las matrices estimadas. Esta matriz se optimiza con el método de Levenberg-Marquardt en base a los *inliers*(puntos verdaderos) a F encontrados. Para este valor final de F se eliminan todos los *outliers*, quedando como resultado únicamente los *inliers*. Si se tienen menos de 20 *inliers*, entonces se descartan tanto la cámara como las correspondencias, al no ser un número suficiente como para obtener una reconstrucción robusta de esa cámara.

Por último, se organizan las correspondencias en trayectorias, donde una trayectoria es

Figura 5.2: Puntos SIFT detectados

Puntos característicos encontrados en uno de los fotogramas del escenario MECO_3_MOD_1. Cada punto viene representado por un vector que indica su escala y rotación. Fuente: e.p.

un conjunto de puntos correspondientes a lo largo de varias imágenes. Si una trayectoria contiene más de un punto clave para la misma cámara, se considera inconsistente y ésta se elimina de la reconstrucción. Finalmente se guardan todas las trayectorias consistentes (que contengan al menos dos puntos correspondientes). En la figura 5.3 se muestra un ejemplo de correspondencias entre dos imágenes.

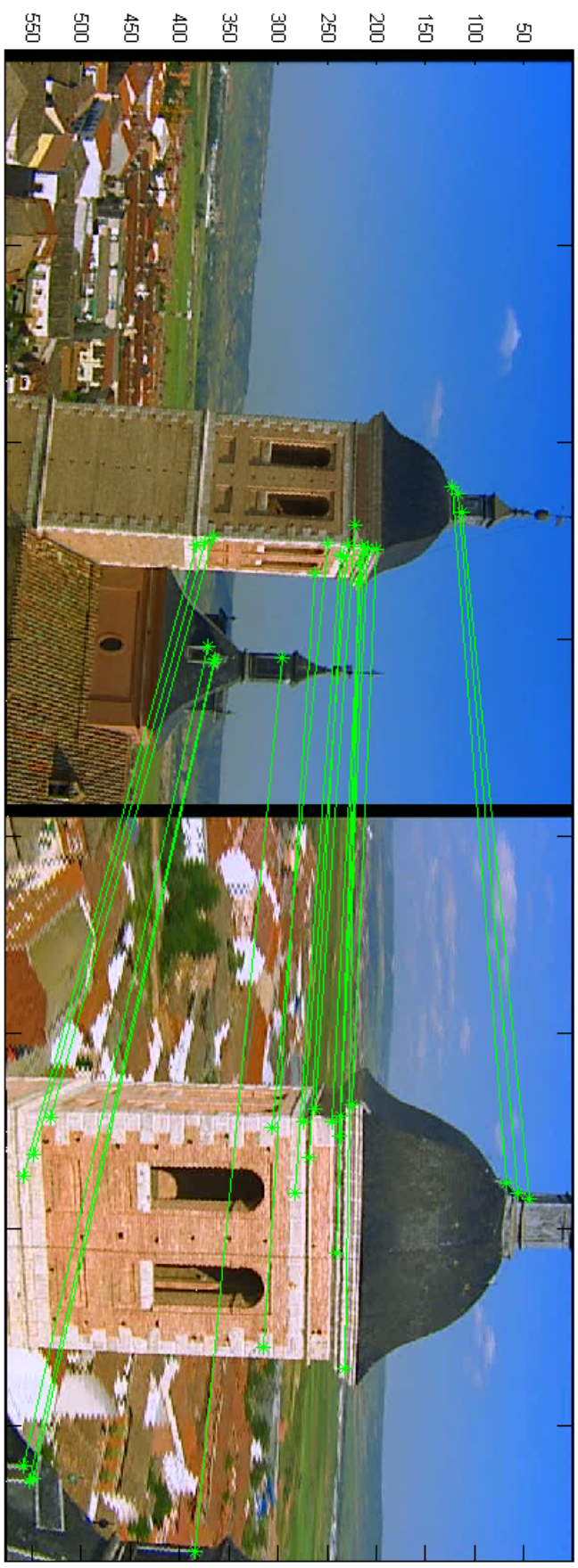

Figura 5.3: Puntos correspondientes entre dos imágenes

Correspondencias encontradas entre dos fotogramas de MECO_3_MOD_1. Fuente: e.p.

5.1.1.3. Reconstrucción de estructura

Para realizar la reconstrucción de estructura, Bundler necesita recuperar un conjunto de parámetros de cámara y de puntos 3D para cada trayectoria encontrada. Los parámetros recuperados deben cumplir que el error de reproyección calculado como la suma de las distancias entre las proyecciones de cada trayectoria y los puntos correspondientes se minimice. Este problema de minimización se formula como un problema de mínimos cuadrados no lineal, que se resuelve con ayuda del algoritmo de Levenberg-Marquardt (ver 4.7.2). Este algoritmo sólo garantiza encontrar mínimos locales, y los problemas de reconstrucción son particularmente susceptibles a caer en mínimos locales no óptimos. De ahí la importancia de obtener unas buenas estimaciones iniciales para los parámetros.

El proceso de reconstrucción de estructura se basa en el algoritmo SBA desarrollado por Lourakis y Argyros [33][34], pero con algunas modificaciones para adaptar su uso al escenario concreto de operación de Bundler. Es conveniente señalar que Bundler es una herramienta de reconstrucción de estructura diseñada para ser usada con colecciones de fotos tomadas por diferentes tipos de cámaras para las cuales no se conocen los parámetros de calibración (aunque como a continuación se verá, es necesario conocerlos al menos para el par inicial). La razón de usar una aproximación incremental a la hora de incluir las cámaras, es evitar que las estimaciones caigan en valores erróneos, empezando la reconstrucción por un par de cámaras fiable (de las que conoce su distancia focal teórica gracias a la cabecera EXIF).

El proceso de reconstrucción con SBA es sencillo, primero se toma el conjunto de trayectorias para estimar la cámara 3D y la geometría de la escena que mejor concuerde con las trayectorias detectadas. La notación aquí seguida es la siguiente, X_j son los puntos 3D de la reconstrucción que se corresponden a cada pista de puntos j. Una cámara se describe por sus parámetros extrínsecos e intrínsecos. Los primeros definen la pose, es decir, la orientación y la traslación con respecto a un origen de coordenadas de la escena. Los intrínsecos modelan el proceso interno de formación de la imagen. Los parámetros extrínsecos de la cámara i son la matriz de rotación R_i, y la traslación t_i a partir de los cuales se puede calcular el centro de la cámara C_i despejando la ecuación $t_i = -R_i C_i$. Los intrísecos de la cámara se representan con una matriz diagonal superior $K_i = diag(f_x i, f_y i, 1)$. Bundler asume que los píxeles son cuadrados y que por tanto los valores de la focal en el eje x e y son iguales. De ahí que se pueda modelar la matriz de calibración tal que $K_i = diag(f_i, f_i, 1)$. La razón de aplicar esta restricción es reducir el número de parámetros a estimar e incrementar la estabilidad del sistema [68]. Además de la matriz de la cámara y la matriz de calibración, Bundler debe recuperar los parámetros que modelan la distorsión radial presente en las cámaras. Esta distorsión radial puede modelarse con un polinomio de cuarto orden,

$$r(p) = 1 + k_1 p^2 + k_2 p^4 \tag{5.1}$$

$$p = \sqrt{(x_{ij})^2 + (y_{ij})^2} \tag{5.2}$$

donde son k_1 y k_2 los parámetros de distorsión y (x_i^j, y_i^j) las coordenadas 2D del punto j en la cámara i.

Las trayectorias de puntos son medidas ruidosas de la escena, lo que dificulta el proceso de reconstrucción. Además de esto, hay que tener en cuenta que no todas las trayectorias serán vistas por todas las cámaras, reduciendo la información disponible para reconstruir.

A su vez, esta última característica se puede explotar para mejorar la eficiencia de la optimización, tal como hace SBA. La optimización se puede resumir a decir que SBA optimiza el conjunto de parámetros de cámara y escena

$$P = P_1, P_2, \ldots, P_n \tag{5.3}$$
$$X = X_1, X_2, \ldots, X_m \tag{5.4}$$
$$\tag{5.5}$$

con n cámaras y m trayectorias(puntos 3D), donde la función de coste mide la discrepancia entre la medida de la posición del punto 2D y aquellos predecidos por la ecuación de proyección. La función de coste es

$$f(P, X) = \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} ||x_{ij} - proy(P_i, Xj)||^2 \tag{5.6}$$

donde $proy(P_i, X_j)$ es la proyección del punto X_j sobre la cámara i, w_{ij} es una variable indicador con valor $w_{ij} = 1$ si la cámara i observa la trayectoria j y $w_{ij} = 0$ en caso contrario. La expresión entre corchetes es el error de reproyección de la trayectoria j en la cámara i. Por tanto, la función de coste f es la suma de los errores cuadráticos de proyección pesados por la variable indicador. El objetivo de SfM es encontrar los parámetros de cámara y escena que minimicen esta función de coste con bundle adjustment.

La función de proyección es una función no lineal de mínimos cuadrados, de ahí que sea necesario aplicar bundle adjustment, que a su vez se basa en LMA. El gran incoveniente de este método es que sólo garantiza encontrar mínimos locales. Como ya se ha comentado, los problemas a gran escala SfM son especialmente susceptibles de verse afectados por mínimos locales [68], de ahí la importancia de disponer de una buena estimación inicial de los parámetros. En vez de inicializar los parámetros para todas las cámaras y puntos a la par, la estrategia de Bundler es tomar una aproximación incremental, comenzando con dos cámaras, buscando sus parámetros óptimos y los de los puntos 3D que ven, y luego añadir iterativamente una cámara cada vez a la optimización.

La elección del par inicial es un paso crítico en el proceso de reconstrucción. Si la reconstrucción del par inicial cae en un mínimo local erróneo, es muy probable que la optimización no se recupere, al depender todo el escenario de este valor inicial [71]. Uno de los problemas al seleccionar este par es que tenga una línea base insuficiente. Para evitarlo, se elige un par con una gran número de puntos correspondientes, pero también con una línea base amplia, de forma que la reconstrucción pueda estimarse de forma robusta. La condición es elegir el par de imágenes con el mayor número de correspondencias sujeto a la condición de que éstas no se puedan modelar mediante una homografía. Una homografía modela la transformación entre dos imágenes de un plano, o dos imágenes tomadas desde la misma posición pero posiblemente con las cámaras mirando en diferentes direcciones. Si no se puede ajustar una homografía a las correspondencias entre las imágenes, indica que las imágenes tienen una cierta separación (línea base) y que es posible recuperar su estructura 3D. El proceso consiste en estimar una homografía de forma robusta con RANSAC entre cada par de imágenes correspondientes usando un umbral de *outliers* de 0.4 % el tamaño más grande de la imagen ($0{,}004 max(ancho\ imagen, alto\ imagen)$).

El porcentaje de *inliers* a la homografía para cada par de imágenes se guarda, y se elige el par con el valor más bajo, con la condición de que tenga al menos 100 correspondencias.

Los parámetros para este par inicial son estimados con el algoritmo de Nistér de 5 puntos para cámaras calibradas, que usa a su vez un método RANSAC para estimar los parámetros de forma robusta. Para usar este procedimiento es necesario cumplir dos condiciones. La primera, es que se conozca la matriz de calibración K de la cámara, o lo que es lo mismo, que se conozca el valor de la longitud focal. La segunda, es que se considere que una de las dos cámaras está en el origen de coordenadas $P = K[I|0]$. La focal de inicialización se extrae de la cabecera EXIF y con ella se inicializa la matriz K para cada imagen. En caso de no disponer de una focal en la cabecera, Bundler inicilizará la focal a $f = 532$ por defecto. Esta inicialización puede llevar a reconstrucciones erróneas como se verá más adelante en este capítulo. Con los parámetros estimados, se triangulan todos los puntos correspondientes comunes a este par. Por último, se toman los parámetros y los puntos y se realiza una optimización *bundle adjustment* basada en el paquete SBA de Lourakis y Argyros. Otro problema de este método viene cuando el par elegido de forma automática tiene un número grande de correspondencias erróneas. Si éstas aparecen como outliers de una homografía dominante, Bundler puede creer que tienen una línea base suficiente cuando en realidad no es así. Cuando esto ocurra, una solución es tomar manualmente el par inicial.

El siguiente paso es añadir nuevas cámaras y puntos a la reconstrucción. El procedimiento básico consiste en seleccionar la cámara con mayor cantidad de trayectorias comunes a los puntos ya reconstruidos. Para inicializar la pose de la nueva cámara, primero se estima su matriz de proyección P usando la transformada directa lineal (DLT), dentro de un procedimiento RANSAC. Para este paso RANSAC se usa un umbral de *outliers* (puntos falsos) de 0.4 % el tamaño mayor de la imagen $(0{,}004max(ancho\ imagen, alto\ imagen))$, con la matriz de proyección P

$$P = K[R|t] \tag{5.7}$$

A partir de la matriz de proyección P se obtienen K, R y t mediante

$$P = K[R|t] = [KR|Kt] \tag{5.8}$$

donde K y R se pueden calcular usando la factorización QR en la submatriz izquierda de tamaño 3×3 y t se toma de la última columna de $K^{-1}P$. El modelo con la nueva cámara se optimiza con SBA, fijando todos los parámetros excepto los de la nueva cámara añadida. Para la optimización se usa la K estimada con la DLT (f_1) o el valor de la cabecera EXIF (f_2). Si el valor de f_2 es muy diferente con respecto a f_1, se emplea el valor calculado con la DLT. Para evaluar esta condición se usa $0{,}7f_1 < f_2 < 1{,}4f_1$. En caso de que se tome la focal EXIF, se añade $\gamma(f - f_2)^2$ a la función de coste para que la focal permanezca cercana a la estimación inicial. En la documentación se indica que el valor usado es $\gamma = 0{,}001$ [70].

A continuación, se añaden los puntos observados por esta nueva cámara y al menos por otra de las cámaras ya reconstruidas. Los puntos añadidos deben cumplir que sea posible obtener una buena estimación de su localización. Para ello, se calculan todos los rayos que puedan modelar el nuevo punto y se toma el par con el máximo ángulo de separación θ. Si $\theta > 2$ se acepta el punto y se triangula. Este método se usa para eliminar puntos lejanos, ya que éstos pueden resultar en localizaciones erróneas al usar cámaras con parámetros ruidosos. Tras triangular todos los nuevos puntos, se optimiza con SBA de nuevo, pero esta vez usando el modelo completo. Este proceso de inicialización de cámara, optimización con bundle adjustment para la nueva cámara, triangulación de puntos, y ejecución de bundle adjustment con el modelo completo, se repite cada vez que se añada una nueva cámara hasta que no haya ni más cámaras ni más puntos. Aquellas cámaras que observen menos

de 20 puntos no se añadirán a la reconstrucción. Por esta razón, sólo se reconstruirá un subconjunto de imágenes de todas las usadas. El resto, será considerado como inválido para la reconstrucción. Este subconjunto viene determinado por las restricciones usadas en la implementación (número de puntos vistos por las cámaras, umbrales de *outliers*, etc.).

Para mayor robustez y velocidad, se aplican una serie de modificaciones al procedimiento SBA básico descrito. La primera modificación consiste en la robustez en el conjunto de trayectorias de puntos. Los puntos falsos pueden tener un efecto importante en la reconstrucción, haciendo que el resultado sea erróneo. Manejar estos *outliers* de una forma robusta es crítico para la solución. Después de cada ejecución de bundle adjustment, se eliminan todas las trayectorias que contengan como mínimo un punto con un alto error de reproyección. El umbral de *outliers* para una imagen dada se adapta a los errores de reproyección de ésta. El procedimiento consiste en que para una imagen I se calcula el 80 percentil del error de reproyección de ésta, que se denominará como d_{80}. El umbral se definirá como $min(max(2.4d_{80}, 4.0), 16.0)$, es decir, su valor variará entre 4.0 y 16.0. La consecuencia es que se eliminarán todos los puntos con un error de reproyección superior a 16 píxeles y se guardarán todos aquellos con un error inferior a 4 píxeles. La validez de los puntos dentro de este rango dependerá de la imagen en concreto. Por último, se vuelve a ejecutar la optimización ya sin los *outliers* eliminados.

La segunda modificación consiste en que en vez de añadir una cámara cada vez a la optimización, se añaden varias cámaras. Para seleccionar las cámaras a añadir, primero se busca aquella con el mayor numero de correspondencias M comunes a los puntos 3D ya añadidos a la reconstrucción, y luego se añade cualquier cámara con al menos $0{,}75M$ correspondencias. El resultado de añadir varias cámaras a la vez es un menor número de iteraciones de bundle adjustment y por tanto una mayor eficiencia.

La optimización con bundle adjustment, no sólo optimiza los parámetros de cámara y los puntos, sino que además estima los parámetros de distorsión de cámara. Los parámetros de distorsión pueden tener un efecto significativo en la precisión de la reconstrucción. Como ya se ha comentado, se usan los dos primeros parámetros, k_1 y k_2. Estos parámetros se inicializan a $k_1 = k_2 = 0$ para cada cámara añadida y se optimizan con SBA añadiendo a la función de coste la siguiente ecuación para cada cámara:

$$\lambda(k_1^2 + k_2^2) \tag{5.9}$$

con $\lambda = 10{,}0$ para la implementación. El objetivo es evitar que los parámetros tengan valores elevados.

Para los escenarios probados en [3][4][71], el tiempo de ejecución es de unas pocas horas hasta alguna semana. Este tiempo viene determinado principalmente por la búsqueda de correspondencias, y el ajuste de *bundle adjustment* iterativo. La complejidad de la etapa de matching es cuadrática con respecto al número de fotos de entrada pero puede mejorarse ejecutando esta etapa en paralelo en diferentes máquinas. La etapa de bundle adjustment depende del número de fotos y puntos triangulados y el número de correspondencias entre cámaras. Si hay mucho acoplamiento entre las imágenes el proceso tiene más variables que optimizar y por tanto será más lento. En las tablas 5.1 y 5.2 se muestran dos ejemplos del tiempo de ejecución de cada una de las etapas de Bundler. Los resultados corresponden a los escenarios MECO_3_MOD_1 con 70 cámaras, y TORRE_PICASSO_AZCA_MOD_1 con 54, ejecutados ambos en el portátil 2 (ver apartado 5.7.1). Como se puede ver, la etapa que

requiere más tiempo de procesamiento es la búsqueda de correspondencias, justo una de las etapas que se pueden paralelizar en varios procesadores. Para el caso de la búsqueda de característicos, aunque tiene menos impacto en términos de tiempo, también puede optimizarse mediante ejecución en paralelo. La única etapa que no podría paralelizarse sería la etapa SfM basada en *bundle adjustment*.

Etapa de la ejecución	Tiempo		Porcentaje
Característicos SIFT	$7m$	$4{,}818s$	19,23 %
Correspondencias	$16m$	$34{,}967s$	45,05 %
Reconstrucción SfM	$13m$	$8{,}812s$	35,70 %
Total	$36m$	$48{,}597s$	100,00 %

Tabla 5.1: Tiempo de ejecución para MECO_3_MOD_1

Etapa de la ejecución	Tiempo		Porcentaje
Característicos SIFT	$8m$	$5{,}057s$	21,50 %
Correspondencias	$18m$	$32{,}119s$	49,29 %
Reconstrucción SfM	$10m$	$59{,}312s$	29,22 %
Total	$37m$	$36{,}488s$	100,00 %

Tabla 5.2: Tiempo de ejecución para TORRE_PICASSO_AZCA_MOD_1

5.1.2. Resultados de Bundler

Bundler produce varios ficheros de salida que guarda en la carpeta *bundle*. Ésta cuelga de la carpeta donde se encuentran las imágenes. Se pueden especificar otras direcciones, pero esta es la configuración por defecto. Los ficheros producidos son nombrados como *bundle_<n>.out* con <n>igual al número de cámaras registradas para ese fichero. Hay que tener en cuenta que Bundler aplica el algoritmo SBA de forma incremental, añadiendo una cámara o un pequeño número de cámaras a cada paso, de ahí que use esta notación para los ficheros. Después de que todas las imágenes hayan sido registradas, Bundler escribe un fichero final llamado *bundle.out*, además de un fichero PLY para visualización. Ambos ficheros contienen la posición de los puntos y datos de las cámaras en una estructura que se comenta en el manual del apéndice A. El fichero *bundle.out* es un fichero de texto que puede ser leído con otra aplicación tal como hace Orto3D para el análisis de resultados. Para visualizar los ficheros PLY, puede usarse la aplicación Meshlab o Scanalyze. Ambas son válidas, pero Meshlab es claramente superior en términos de sencillez de interfaz de usuario y visualización del escenario. Las salidas de las versiones $v0.1$ y $v0.2$ de Bundler no son totalmente compatibles con Meshlab, por lo que es necesario realizar unos cambios con el editor de texto en los ficheros PLY. En la versión $v0.3$ no hay ningún problema de este tipo.

El índice del conjunto de cámaras comienza con 0 que se corresponde con la primera imagen (cámara) en *list.txt*[3]. El sistema de coordenadas de la imagen tiene como origen el centro de ésta, el eje x crece hacia la derecha, y el eje y crece hacia arriba en la imagen. Por tanto $(-w/2, -h/2)$ son las coordenadas de la esquina inferior izquierda de la imagen

[3]En Orto3D los índices de las cámaras comienzan con 1.

y $(w/2, h/2)$ es la esquina superior derecha donde w y h son el ancho y el alto de la imagen respectivamente.

El modelo de cámara usado por Bundler es una cámara *pinhole* para la que se estiman la focal, los coeficientes de distorsión, la matriz de rotación y la matriz de traslación[4]. Para proyectar un punto 3D en la imagen se debe realizar una proyección de las coordendas espacio a las coordenadas imagen,

$$P = RX + t \tag{5.10}$$

$$p = -P/P.z \tag{5.11}$$

$$p' = fr(p)p \tag{5.12}$$

donde $P.z$ es la tercera coordenada de P. En la última ecuación, $r(p)$ es una función que calcula el factor de escala para corregir el problema de la distorsión radial:

$$r(p) = 1{,}0 + k1||p||^2 + k2||p||^4 \tag{5.13}$$

Adicionalmente, en el sistema de coordenadas de la cámara, el eje z positivo apunta hacia atrás tal como si la cámara mirase hacia el eje negativo z, como en OpenGL. Finalmente, las ecuaciones de arriba implican que la dirección de vista de la cámara es:

$$D = R^t * [0, 0, -1]^t \tag{5.14}$$

y la posición 3D de la cámara es:

$$C - R^t * t \tag{5.15}$$

Se añade un manual en el apéndice A con más detalles sobre cómo usar esta aplicación y sobre el formato de los resultados obtenidos.

5.2. Prototipo Orto3D

Orto3D es una aplicación completamente desarrollada en este trabajo con el objetivo de ser usada en combinación con Bundler. La funcionalidad de esta aplicación es validar los resultados obtenidos con el software de reconstrucción de estructura. Se ha implementado en Matlab con una interfaz gráfica que permite al usuario operar visualmente con los resultados. Esta interfaz se ha realizado usando el editor GUIDE. La aplicación incluye diversa funcionalidad para la validación de resultados, estando ésta basada en el uso de los datos de salida escritos en el fichero *bundle.out*. El manual de usuario del programa se encuentra en el apéndice B. En esta sección se explica el desarrollo interno del programa y la funcionalidad implementada.

El objetivo de esta aplicación es aportar un conjunto de funciones para visualizar los resultados obtenidos, así como para seleccionar manualmente correspondencias en las imágenes. Los puntos seleccionados manualmente se pueden triangular y añadir al espacio tridimensional. La lectura de los resultados se realiza a partir del fichero *bundle.out* y los nombres de las imágenes se obtienen del fichero *list.txt*. Los puntos marcados manualmente y usados para seleccionar rectas o polígonos se triangulan a partir de tres imágenes del conjunto de imágenes de la reconstrucción.

[4]Las ecuaciones se han extraído del *README.txt* de Bundler.

Del fichero *bundle.out* se obtienen los puntos y los parámetros de las cámaras. A partir de los segundos, se triangulan los puntos manuales y se calculan los centros de las cámaras y los ejes principales, ambos representados en el prototipo. La representación de este eje permite al usuario reconocer la dirección de observación de la cámara en el escenario, pudiento validar o rechazar los escenarios en función de si esta representacion es coherente o no con las imágenes (por ejemplo, si las imágenes corresponden a un edificio, los ejes principales deben ir dirigidos a éste). Los puntos manuales permiten marcar rectas o cuadriláteros. La primera recta tomada manualmente se empleará como eje Z de la reconstrucción. Esto sirve para registrar todo el escenario a un nuevo marco de coordenadas donde sea más factible el reconocimiento de los objetos o estructuras reconstruidos. El objetivo es dejar todo el escenario de forma que se pueda ver la planta de éste. Si se trata de escenas urbanas, o con estructuras o edificios, será fácil el reconocimiento de éstos en planta. Opcionalmente se usa la correlación para el marcado manual de los puntos con el objeto de afinar la posición de las correspondencias en las imágenes con respecto al punto marcado en la primera (ver apéndice B). La precisión de estos datos es muy relevante en el resultado de la reconstrucción, de ahí que se necesite tomar con el menor error posible. A continuación, Orto3D permite georreferenciar las reconstrucciones obtenidas a ortoimagenes de las que se conocen las coordenadas métricas. Con este método se pueden recuperar datos métricos aproximados del escenario. Esta aplicación no permite obtener resultados precisos debido a los errores introducidos por varias causas. En primer lugar, la reconstrucción de estructura con Bundler depende del uso de correspondencias robustas y precisas. Además, Bundler emplea cámaras no calibradas y si no se dispone de un valor de inicialización para la cámara cercano al valor real, las estimaciones pueden introducir errores adicionales. Por último, la georreferenciación con Orto3D se basa en el uso de puntos marcados manualmente, que depende directamente de la resolución de la imagen. Si la resolución es baja será difícil distinguir objetos en la imagen y poder marcar puntos correspondendientes en ellos. Para tener escenarios precisos se deberán añadir restricciones adicionales a las reconstrucciones.

5.2.1. Implementación de la aplicación

Como se ha comentado anteriormente, la aplicación ha sido desarrollada en Matlab, y la interfaz gráfica se ha creado mediante el editor GUIDE de esta aplicación. El desarrollo de interfaces gráficas de usuario en Matlab es un proceso sencillo, ya que el editor permite generar automáticamente la estructura del código para manejar las ventanas. El desarrollador únicamente debe darle la funcionlidad a cada botón, función, gráfico, etc. El principal inconveniente a la hora de comenzar a desarrollar este tipo de aplicaciones es la falta de un buen manual con todas las funcionalidades existentes bien documentadas. Sin embargo, es fácil hacerse con la herramienta una vez se empieza a trabajar con ella. En cuanto al código en Matlab, sus características hacen que sea fácil y rápido desarrollarlo, aunque el inconveniente es que su ejecución es bastante más lenta que la de un código desarrollado en lenguajes más eficientes.

El código se presenta documentado con comentarios y con ayudas HTML, por lo que en esta sección se explicará únicamente la funcionalidad implementada más destacable. Ésta es la siguiente:

- Marcado de puntos manuales y triangulación

- Cálculo del eje Z y transformación del escenario
- Marcado del punto XY
- Marcado de rectas de referencia y transformación del escenario
- Georreferenciación
- Estimación de la relación metros píxel
- Cálculo de las medidas en metros

A continuación se explica cada uno de los puntos con más detalle.

Marcado de puntos manuales y triangulación

Como ya se ha comentado, se denominan puntos manuales a los puntos correspondientes marcados por el usuario sobre las imágenes. Esta tarea incluye el cálculo de las matrices fundamentales para hallar las líneas epipolares que sirvan de apoyo a la hora de marcar los puntos. La precisión de esta tarea es vital para que el resultado de la triangulación sea bueno. Si el marcado se hace de manera poco precisa, el punto triangulado puede resultar en una posición lejana al punto real. El error es dependiente del escenario reconstruido en cuanto a que viene determinado por el rango de distancias, las posiciones de las cámaras (su línea base), los errores en los parámetros de cámara estimados por Bundler y el número de cámaras usado para reconstruir un punto. Para la triangulación con Orto3D se usan los parámetros estimados por Bundler, que se consideran correctos y no se modifican en la aplicación.

El marcado se emplea para la selección de aristas, poligonales (cuadriláteros) y un punto en el plano XY del sistema de referencia. El código es idéntico para los tres casos, por lo que se explicará de forma general. La particularidad de cada caso reside en el número de puntos que se marcan y se triangulan pero a efectos de implementación es idéntico. Para el marcado sin correlación, se abren tres ventanas con tres imágenes seleccionadas previamente por el usuario en una etapa anterior (ver el manual de Orto3D en el apéndice B). Se marca un punto en la imagen 1 y para éste se calcularán las epipolares en las otras dos imágenes. Para la imagen 2 se usa la epipolar con respecto a la imagen 1 y para la imagen 3 con respecto a 1 y 2. El cálculo de la epipolar es sencillo una vez se tiene la matriz fundamental y las coordenadas de los puntos normalizadas. Hay que tener en cuenta que las coordenadas de las imágenes en Matlab tienen como origen la esquina superior izquierda, pero Bundler usa el centro de la imagen. Antes de operar con los puntos marcados es necesario pasarlos al sistema de coordenadas de Bundler. También se calculan las matrices fundamentales usando la función *F_de_KRt*. Esta función permite calcular las matrices fundamentales para dos cámaras a partir de los parámetros de cámara definidos por las matrices de proyección P para cada cámara. Como se podrá observar al usar Orto3D, las matrices fundamentales estimadas a partir de los datos de Bundler no cumplirán extrictamente la condición de tener rango 2. Esto se comprueba calculando su determinante y comprobando si es nulo. En cualquier caso, el valor calculado del determinante de cada cámara, aunque no sea nulo, es un valor despreciable para la mayoría de los escenarios aquí presentados. Para que el usuario sea consciente de ésto, en la línea de órdenes se muestra el valor del determinante cuando no es nulo.

Cuando se han marcado todos los puntos, se aplica la triangulación a todos ellos. La

función *fncn_triangula_3* es la que contiene esta parte del código. Aunque no se ha comentado antes, las funciones implementadas se encuentran todas en un fichero del mismo nombre, de forma que sea fácil encontrarlas en el directorio *orto3D*. La triangulación consiste en dos partes. Primero se triangula con un método lineal, y después se optimiza mediante un método no lineal de mínimos cuadrados basado en LMA. Las funciones que implementan esto son *vgg_X_from_xP_nonlin_lma* y *vgg_X_from_xP_lin_non_hom* respectivamente, basadas en funciones del grupo VGG [52] desarrolladas a partir de la teoría de [25]. Una vez se ha triangulado el punto, éste se puede añadir a la reconstrucción final. El método no lineal tiene como función de coste el error de reproyección de los puntos marcados sobre las tres imágenes, que se ejecuta para cada trío de puntos correspondientes de manera independiente. Este método de añadir puntos a la reconstrucción es diferente al método interno de Bundler, sin usar muchas de las restricciones impuestas en su algoritmo de SfM. Por un lado es más simple, pero por otro es más susceptible de tener errores. En primer lugar no se chequea la condición de 2 entre los rayos determinados por los puntos y usada en Bundler para eliminar puntos lejanos que pudieran tener un gran error de profundidad. Esto obliga a que el trío de cámaras seleccionado en Orto3D necesite tener una buena línea base con respecto al escenario (que las cámaras estén separadas). Por otro lado, no se eliminan trayectorias con un error grande de reproyección, por lo que no hay forma de evitarlo si ésto ocurre en el prototipo. Y por último, el método es mucho más limitado que en Bundler. Se emplean únicamente tres cámaras, se usa únicamente un punto en la optimización, y no se emplea el método SBA de Bundler. Usando SBA dentro de Orto3D, con más cámaras, y con más puntos manuales a la vez podría dar mejores resultados. Sin embargo uno de los objetivos de este prototipo es disponer de una herramienta sencilla y simple, razón por la cual se ha seguido la aproximación aquí presentada. Por un lado, seleccionar manualmente puntos en más de tres cámaras es una tarea tediosa para el usuario, a la vez que el método SBA integrado en Matlab puede ser lento al usar varios puntos y el escenario completo. En siguientes versiones se puede plantear una mejora del prototipo con estas consideraciones y esta funcionalidad integrada.

Cálculo del eje Z y transformación del escenario

El eje Z se toma a partir de una arista marcada manualmente por un usuario. Para marcarla, sólo es necesario tomar dos puntos por imagen. Estos puntos definen una recta en el espacio tridimensional, que a su vez define unos ángulos con el sistema de coordenadas actual, de los cuales son útiles dos. Se llama λ al azimut, calculado como el ángulo de la proyección de esta recta sobre el plano XY con respecto al eje X (rotación en Z). Sean los puntos que definen la recta $X_1 = (X_1, Y_1, Z_1)$ y $X_2 = (X_2, Y_2, Z_2)$, se puede calcular el ángulo λ como

$$\lambda = \arctan(\frac{Y_2 - Y_1}{X_2 - X_1}) \tag{5.16}$$

que se usa para definir una matriz de rotación con el objeto de colocar los puntos que definen la recta en el plano XZ. La matriz de rotación es

$$M_\lambda = \begin{bmatrix} \cos(-\lambda) & -\sin(-\lambda) & 0 & 0 \\ \sin(-\lambda) & \cos(-\lambda) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.17}$$

Luego se busca calcular la elevación, determinada por el ángulo ϕ que forma la recta con respecto al eje Z, calculado después de aplicar la rotación anterior a la recta. El ángulo

será entonces

$$\phi = \arctan(\frac{Z'_2 - Z'_1}{X'_2 - X'_1}) \tag{5.18}$$

con $X'_1 = (X'_1, Y'_1, Z'_1)$, $X'_2 = (X'_2, Y'_2, Z'_2)$ las coordenadas de los puntos que forman la recta después de la rotación con λ. A partir de ϕ se define otra matriz de rotación,

$$M_\phi = \begin{bmatrix} \cos(-\frac{\pi}{2} - \phi) & 0 & -\sin(-\frac{\pi}{2} - \phi) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(-\frac{\pi}{2} - \phi) & 0 & \cos(-\frac{\pi}{2} - \phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.19}$$

que aplicada a los puntos que definen la recta, hacen que ésta quede perpendicular al plano XY.

Por último, es necesario calcular la traslación a aplicar para que la recta esté en el eje Z. Para ello se toma el segundo punto que define la recta después de las dos rotaciones, $X''_2 = (X''_2, Y''_2, Z''_2)$. Al marcar esta recta, se recomienda que el segundo punto marcado sea el más próximo al suelo. En cualquier caso, si no se tiene en cuenta esta condición y el resultado es el escenario transformado inverso, se puede rotar el escenario 180 con respecto al eje Z con otra función de la interfaz. Más adelante se comenta un poco más. Si X'''_2 es este mismo punto tras la trasformación de traslación, éste debe tener coordenadas $X'''_2 = (0, 0, Z''_2)$. Por tanto, la matriz de traslación es

$$M_{tras} = \begin{bmatrix} 1 & 0 & 0 & -X''_2 \\ 0 & 1 & 0 & -Y''_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.20}$$

Para pasar todo el escenario a este nuevo sistema de coordenadas definido por la recta marcada, es necesario aplicar las matrices anteriores a cada uno de los puntos y a cada una de las cámaras. Para ello se usa la matriz M tal que

$$M = M_{tras} M_\phi M_\lambda \tag{5.21}$$

El cambio de coordenadas a los puntos es simple. Si X_i es un punto en el espacio tridimensional, al transformarlo queda

$$Xnuevo_i = MX_i; \tag{5.22}$$

y para las matrices de las cámaras, el cambio consiste en usar la inversa de M aplicada a la matriz P_j de cada cámara j,

$$Pnueva_j = P_j M^{-1} \tag{5.23}$$

A partir de $Pnueva_j$ se calculan las matrices de rotación $Rnueva_j$, el vector de traslación $tnueva_j$ y los centros de cámara $Cnueva_j$ referidos al nuevo sistema de coordenadas.

También se ha añadido a la interfaz funcionalidad para girar el escenario 180º en XY y en XZ. Para ello se usan transformaciones similares a las rotaciones aplicadas a la arista que define el eje Z. La única diferencia es que el ángulo empleado es de 180º. Con ésto se evita que el escenario quede del revés al definir el eje Z, debido a la ambigüedad de los ángulos calculados en la transformación anterior. El usuario será el encargado de elegir

si acepta el escenario transformado con la rotación automática o lo invierte para corregir el sentido del giro. Esta tarea es fácil de aplicar al visualizar el escenario. Si aparece del revés con respecto al eje Z, se debe aplicar un giro en el plano XY, y si aparece del revés con respecto al eje Y se debe aplicar un giro en el plano XZ.

Aunque no se comenta antes, se debe mantener el orden de la ecuación 5.21, ya que el resultado no es invariante al orden de multiplicación de las matrices. Operando en el orden de esta ecuación el resultado es

$$M = \begin{bmatrix} \cos(-\frac{\pi}{2}-\phi)\cos(-\lambda) & -\cos(-\frac{\pi}{2}-\phi)\sin(-\lambda) & \sin(-\frac{\pi}{2}-\phi) & -X_2'' \\ \sin(-\lambda) & \cos(-\lambda) & 0 & -Y_2'' \\ \sin(-\frac{\pi}{2}-\phi)\cos(-\lambda) & -\sin(-\frac{\pi}{2}-\phi)\sin(-\lambda) & \cos(-\frac{\pi}{2}-\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.24)$$

Si se opera con otro orden se puede comprobar que el resultado no es el mismo.

Marcado del punto XY

El marcado del punto XY permite cambiar de nuevo el escenario de forma que el punto triangulado se encuentre en el plano XY. Esta función sirve para elegir un punto de referencia en altura, es decir $X_{xy} = (X, Y, 0)$. Todas las alturas del escenario irán referidas a este punto. En un inicio, la implementación se realizó de forma que el origen de coordenadas, también en la componente Z, estuviese definido por el eje Z, sin embargo tras experimentar, con varios escenarios se vio que en algunos de éstos convenía usar aristas bien definidas pero que no llegaban hasta el suelo del escenario. En este caso, los puntos por debajo de la altura definida por el eje Z quedaban por debajo del plano suelo. Para evitarlo, se introdujo la funcionalidad aquí comentada. Con respecto a la transformación aplicada, es similar a la traslación del cambio de eje Z, usando las coordenadas del punto XY marcado en vez de las de la recta. El cambio no se aplica directamente tras la selección del punto, sino durante la trasformación del escenario para registrarlo con respecto a la ortoimagen que se comenta a continuación.

Marcado de rectas de referencia y transformación del escenario

Una vez seleccionado el eje Z y marcados los cuadriláteros y el punto XY es necesario definir dos rectas, una sobre la planta del escenario reconstruido y otra sobre la ortoimagen. Ésta tarea sólo podrá realizarse en escenarios donde sea fácil reconocer objetos en planta, como escenas urbanas, estructuras o edificaciones en general. Se presenta un ejemplo en las figuras 5.4, 5.5 y 5.6. En éstas se puede ver lo fácil que es reconocer objetos en la vista en planta del escenario reconstruido y compararlos con una ortoimagen (figura 5.7).

Las rectas vienen definidas por cuatro puntos, dos cada una. Se deben marcar rectas iguales en el escenario, es decir los puntos marcados deben corresponderse entre sí. Las transformaciones de rotación, traslación y escala a aplicar al escenario para que la nube de puntos y cámaras se puede posicionar sobre la ortoimagen se calculan mediante los ángulos que forman las rectas, sus módulos y sus posiciones con respecto al origen de coordenadas. Para evitar trabajar con las coordenadas imagen de Matlab, se transforman las coordenadas de la recta marcada en la ortoimagen al sistema de coordenadas de Matlab para la representación de los puntos. Éste otro sistema de coordenadas toma la esquina inferior izquierda como origen del sistema. La transformación es sencilla, y consiste en invertir la componente y, y aplicar un cambio de escala de forma que las rectas tengan las

Figura 5.4: Fotograma del escenario ALCALA_3_MOD_2
Vuelo realizado en Alcalá de Henares, con el Rectorado de la Universidad de Alcalá en la parte central. Fuente: documentales *Madrid desde el aire*.

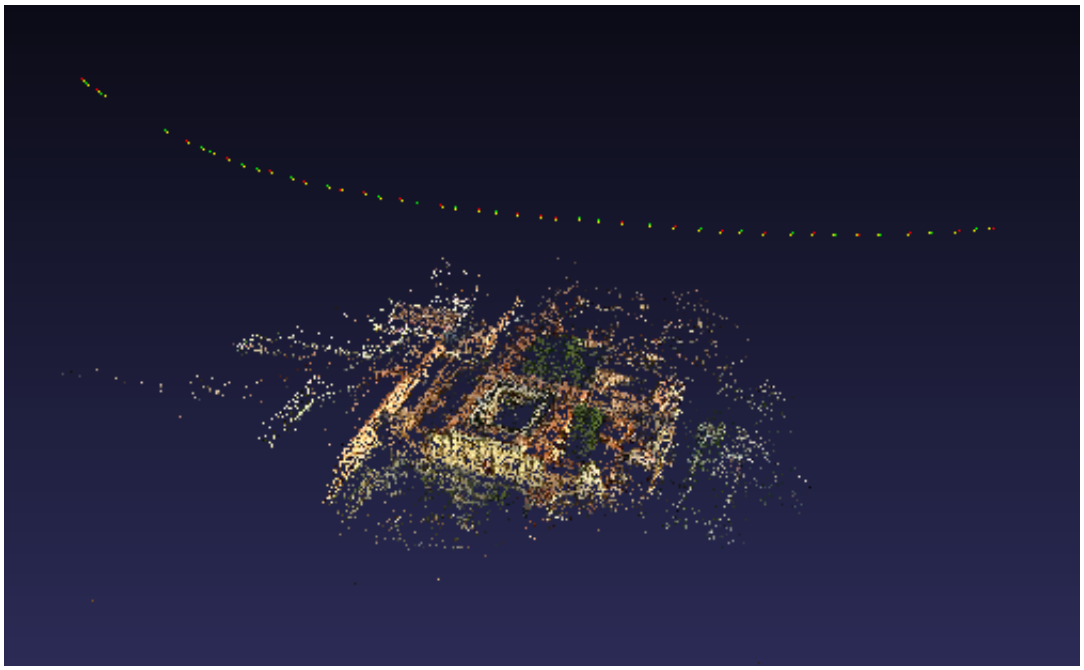

Figura 5.5: Reconstrucción del escenario ALCALA_3_MOD_2
Vista de la reconstrucción con la trayectoria del vehículo aéreo en la parte superior y la nube de puntos del escenario en la parte central. Fuente: e.p.

mismas dimensiones después de aplicar las transformaciones. A continuación se define un ángulo que defina la rotación entre ellas y por último la distancia entre ellas para definir la traslación a aplicar a la recta en la reconstrucción. Con estos valores de escala, rotación y traslación, es posible aplicar la misma transformación al escenario completo para poder posicionarlo sobre la ortoimagen.

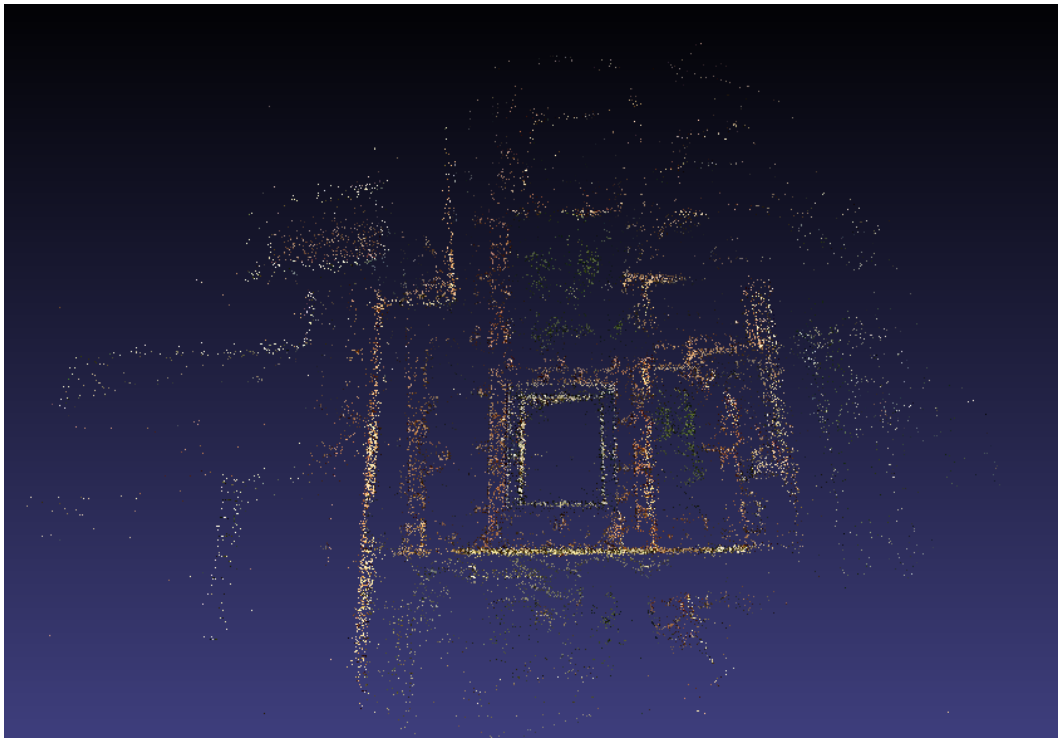

Figura 5.6: Planta de la reconstrucción del escenario ALCALA_3_MOD_2

Vista en planta de la nube de puntos del escenario. Fuente: e.p.

Figura 5.7: Ortoimagen del escenario ALCALA_3_MOD_2

Ortoimagen de la zona de vuelo para este escenario. La planta de los edificios se puede identificar fácilmente en esta figura y en la anterior para realizar la georreferenciación. Fuente: [45]

Georreferenciación

Cuando el escenario ha sido posicionado sobre la ortoimagen, habrá una correspondencia directa entre las coordenadas de la ortoimagen y las coordenadas de los puntos y cámaras. De la ortoimagen se toman las coordenadas en X e Y, y para la componente Z se asume que la ortoimagen está en el plano XY, o lo que es lo mismo, en $Z = 0$. El punto XY marcado con anterioridad deberá estar en la ortoimagen, y como se ha dicho antes, es el que define la altura del escenario. Todas las componentes Z de los puntos irán referidas a éste. Para las coordenadas X e Y se toma como origen la esquina inferior izquierda de la ortoimagen como origen del sistema de referencia. Todas las coordenadas se darán relativas a este punto en metros, evitando trabajar con las componentes UTM completas, que son valores muy grandes que dan problemas al trabajar con las matrices de las cámaras en esas magnitudes. En este punto se aplica la última transformación, que es simplemente una transformación de escala. El cambio de escala se calcula a partir de la relación de las coordenadas del escenario con las coordenadas métricas de la recta en la ortoimagen. Para ello se usa una medida de referencia en la ortoimagen que indica la relación metros/píxel en ésta. Esta relación se calcula mediante unos puntos de referencia de los cuales se conocen sus coordenadas en píxel y en metros. Los puntos pueden introducirse manualmente o de forma automática, siendo preferible la segunda. La forma automática usa un fichero denominado *ortoimagen.txt* dentro de la carpeta *orto3D* del escenario. Esta carpeta se encuentra en el directorio de la reconstrucción y también contiene la ortoimagen en *ortoimagen.jpg*. La descripción del fichero se detalla en el manual de Bundler adjunto (ver B). Para poder calcular la relación, es necesario al menos dos puntos. El cálculo es muy simple, se toman dos puntos con sus coordenadas en píxel y en metros, luego se calculan los módulos que forman las rectas definidas en ambos sistemas de coordenadas, y por último, se calcula la relación entre sus módulos, que es el resultado final de metros por píxel.

Cálculo de las medidas en metros

En la funcionalidad de cargar un escenario se presenta la opción de medir una distancia en metros. En realidad se trata del marcado de una arista modificado. La modificación consiste en que después del marcado manual y de la triangulación se calcula también su módulo. Como el módulo es la distancia en metros entre estos dos puntos, se tiene una herramienta sencilla de medida dentro del escenario. Las medidas obtenidas no serán precisas, ya que existen varias fuentes de error que harán que esta medida sea aproximada. Por un lado, los parámetros de cámara tendrán errores al no disponer de valores iniciales fiables, por otro, la georreferenciación también introduce errores, y por último, el marcado manual también es poco preciso. La validez de la medida dependerá de cada escenario en concreto. Al no disponer de medidas reales para los vuelos, no se han podido validar los valores obtenidos. Éste puede ser un posible trabajo de continuación.

5.2.2. Resultados de Orto3D

Orto3D da como resultados varios ficheros, de los cuales, los más interesantes son *orto3D.out*, *orto3D.ply* y *orto3D_plano.ply*. En el apéndice B se comentan con más detalle.

5.3. Resultados, problemas y soluciones

En esta sección se realiza un comentario general de los resultados. A lo largo del trabajo se han realizado varias pruebas con Bundler, obteniendo resultados muy diferentes. Como el objetivo final del trabajo es operar con fotogramas tomados desde UAV, el trabajo se comenzó con una muestra de vídeos reales tomados por el SIVA. Los resultados, que se comentan en 5.3.1 , fueron poco satisfactorios. La configuración del escenario, el desconocimiento de los parámetros de las cámaras y la pequeña resolución de las imágenes no permitieron obtener resultados adecuados. Un problema identificado prácticamente al inicio fue el debido a los caracteres sobreimpresos en la imagen. Para evitar el efecto negativo producido, se implementó un filtro sencillo, que se comenta en 5.3.2. Para trabajar con mejores escenarios y poder desarrollar el prototipo, se usaron fotografías de un edificio de planta cuadrada tomadas con una cámara digital compacta. Una vez desarrollado el prototipo, se volvió a trabajar con escenarios aéreos, esta vez con imágenes de buena resolución tomadas de secuencias de vídeo para documentales. Existe una gran cantidad de imágenes de este tipo a un coste muy bajo, de ahí que se tomase esta fuente. Los resultados obtenidos se comentan en 5.3.3.

5.3.1. Resultados iniciales con el SIVA

Como el objetivo de este trabajo es usar escenarios grabados con UAV, las primeras pruebas se aplicaron a imágenes tomadas por el SIVA. Éste se trata de un sistema con capacidad para transmitir vídeo en el espectro visible y en el infrarrojo (ver sección 3.3). Los fotogramas empleados como primera prueba corresponden a un vuelo en Alconada de Maderuelo, en las proximidades de Corral de Ayllón (Segovia). En estas imágenes se ve la iglesia del pueblo con algunas de las casas. En ésta secuencia se pueden apreciar bien los objetos y el vehículo realiza un movimiento paralelo al pueblo. Sin embargo, dado el rango de distancias del escenario, será difícil estimar la profundidad. Por esta y otras razones, los resultados no fueron satisfactorios. Bundler no pudo recuperar las focales de algunas cámaras, y para otras obtuvo estimaciones muy dispares entre sí a pesar de usar la misma cámara. Estos resultados eran totalmente incoherentes con los fotogramas y las cámaras empleadas, por ejemplo con valores superiores al millón de píxeles trabajando con imágenes de poco más 300 píxeles de ancho. En las figuras 5.8 y 5.9 se pueden ver los valores de las focales para el escenario *ALCONADA_1* del SIVA usando 100 cámaras. Como no se dispone de ningún parámetro inicial sobre las cámaras y el escenario usado es muy complejo, se plantea forzar la inicializacion de la focal de los fotogramas a un valor de $f = 500$ (ver sección 5.3.4 sobre estimación de focales). El valor se toma asumiendo dos condiciones. La primera es que entre los fotogramas empleados no hay modificaciones de zoom. La segunda es que las focales de las cámaras tienen una valor cercano al ancho de la imagen. Como el ancho de estos fotogramas es de 352 píxeles, se ha tomado un valor aleatorio cercano, en este caso 500. Los valores de las focales recuperados son más estables y similares, sin haber focales negativas ni una gran disparidad, aunque sigue sin ser posible reconocer la nube de puntos. En parte se debe a la pequeña resolución pixélica de las imágenes, que hace que el número de puntos característicos detectados sea muy bajo y que por tanto también lo sea el número de puntos correspondientes. En las figuras 5.10 y 5.11 se pueden ver los valores de focal estimados para $f = 500$ píxeles. Como las nubes de puntos son totalmente irreconocibles, se vió necesario disponer de una herramienta que ayudase a visualizar los resultados obtenidos así como con métodos para validar el escenario reconstruido. Las herramientas de visualización empleadas no disponen de esta funcionalidad, por lo que ya desde los inicios del trabajo se vió la importancia de desar-

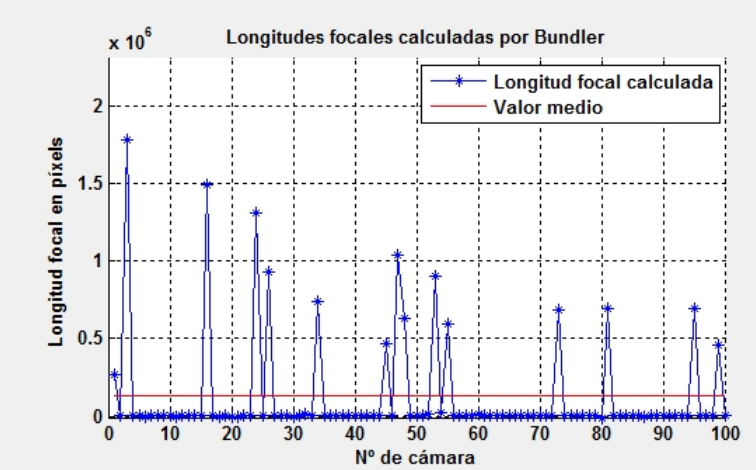

Figura 5.8: Focales estimadas por Bundler en ALCONADA_1

Gráfica de las focales. En la imagen se puede ver que los valores estimados son muy dispares entre sí. Fuente: e.p.

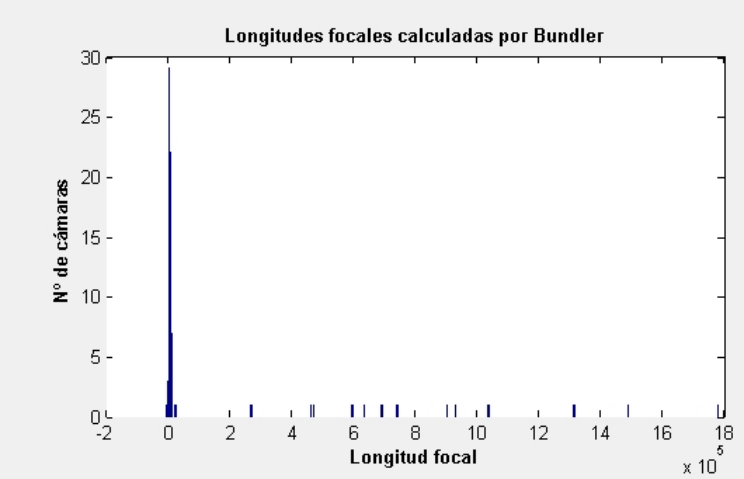

Figura 5.9: Distribución de las focales en ALCONADA_1

Gráfica con la distribución de las focales estimadas. Los resultados ocupan un rango muy grande de valores, mostrando la disparidad de los datos. Fuente: e.p.

rollar una herramienta propia con las características requeridas, y es aquí donde comienza el desarrollo de Orto3D, el prototipo en Matlab comentado con anterioridad.

Una de las primeras funcionalidades requeridas fue la visualizacion de puntos correspondientes en las imágenes. Una de las causas de que las reconstrucciones sean erróneas o no sean reconocibles puede ser que los puntos correspondientes empleados no sean correctos. Gracias a esta funcionalidad se pudieron identificar problemas con los caracteres impresos sobre la imagen. Estos caracteres corresponden a información sobre el vuelo, muy común en imágenes tomadas con UAV. SIFT detecta puntos característicos en éstos que Bundler luego no es capaz de eliminar. Si además el número de puntos obtenidos es pequeño debido a la baja resolución de la imagen (352×258 píxeles), y de este número, los puntos erróneos en los artefactos son un porcentaje alto, Bundler creerá que son correctos, devolviendo una reconstrucción errónea. Para evitar este problema se propuso la implementación de un filtro externo a Bundler implementado en C y basado en la distancia entre píxeles correspondientes asumiendo que el vídeo va a tomarse siempre en movimiento. El

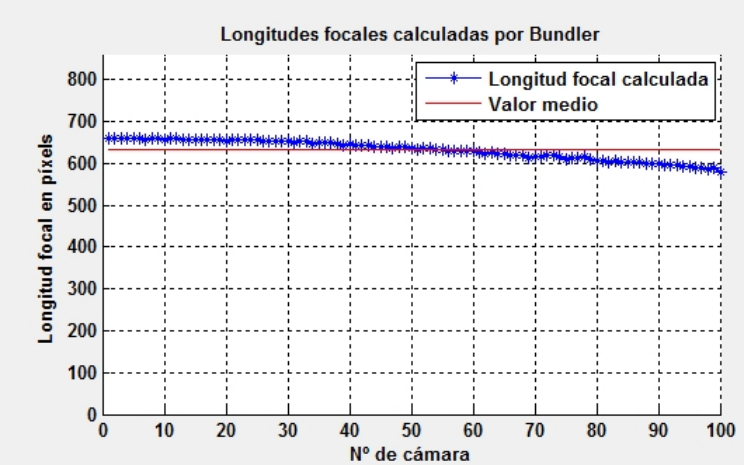

Figura 5.10: Focales estimadas por Bundler en ALCONADA_1_MOD_1

Gráfica de las focales. En la imagen se puede ver que al usar el forzado de focal con $f = 500$ píxeles los valores estimados son cercanos entre sí. Fuente: e.p.

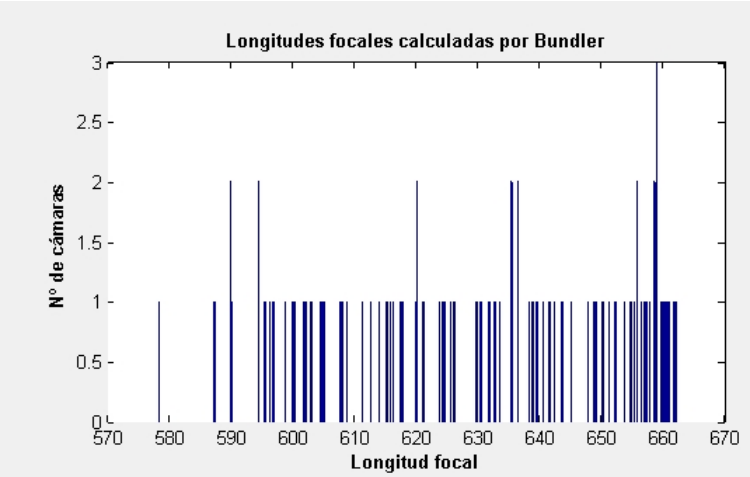

Figura 5.11: Distribución de las focales en ALCONADA_1_MOD_1

Gráfica con la distribución de las focales estimadas. Los resultados ocupan un rango pequeño de valores. Fuente: e.p.

filtro y los resultados obtenidos con él se comentan en el siguiente apartado.

Tras estos resultados, la primera conclusión es que para mejorar el aspecto de las reconstrucciones es necesario usar imágenes con mayor resolución. El problema es disponer de vídeos mejores. Otra conclusión es que las focales de inicialización son determinantes en la reconstrucción. En la sección 5.6 se comentan con más detalle. Otro problema es que el rango del escenario es muy grande con respecto a la separación entre fotogramas (baseline) y la distancia entre objetos y cámaras. Para obtener una buena estimación de la profundidad es necesario usar un vídeo con una separación mayor entre fotogramas, usar un vídeo de mayor duración y además un vídeo que visualice el escenario desde varios ángulos. Como resumen, los problemas identificados inicialmente son los siguientes:

- Presencia de artefactos sobreimpresos en la imagen, tales como caracteres de datos del vuelo. Estos puntos detectados pueden provocar errores en la reconstrucción al usar correspondencias erróneas. Para evitarlo se puede implementar un filtro que

las elimine. Todos los puntos detectados que cumplan esta condición van a estar situados en los mismos puntos de la imagen, justo en los caracteres, de ahí que sea sencillo detectarlos y rechazarlos. La solución propuesta consiste en implementar un filtro que los elimine cuyo criterio de selección sea la distancia entre los puntos detectados en varias imágenes suponiendo que las cámaras están en movimiento. Si la distancia es inferior a un umbral definido, el punto es rechazado. En caso contrario se acepta. Al estar en movimiento, salvo que el movimiento sea circular, es imposible que se detecten dos puntos correspondientes con las mismas coordenadas en ambas imágenes, de ahí que se puedan considerar como erróneos. Si no hubiese movimiento de cámara, las imágenes serían idénticas y tampoco serían válidas para reconstruir y si el movimiento fuese circular, podría haber puntos correctos en el centro de la imagen que fuesen eliminados, donde los objetos tenderían a estar en las mismas posiciones por las características del vuelo, aunque es un hecho poco probable.

- Resolución de imagen baja. Las imágenes empleadas son de pequeño tamaño de 352×258, muy inferior a las imágenes empleadas en la documentación ofrecida por Bundler. A esto hay que sumarle que las imágenes están comprimidas, lo cual reduce la calidad de las mismas. La compresión es necesaria para la transmisión de vídeo en el canal aéreo, caracterizado por una ancho de banda limitado. Como las imágenes son pequeñas, SIFT detecta una pequeña cantidad de correspondencias, que resultará en una pequeña cantidad de puntos reconstruidos. Por esta razón es difícil reconocer cualquier tipo de objeto en la nube de puntos. Como la visualización de los resultados no es suficiente para validar la reconstrucción, es necesario crear una herramienta de validación. Es en este momento cuando se plantea el desarrollo de Orto3D, con funcionalidad para marcado manual de puntos correspondientes. Éstos se triangularán a partir de los parámetros de cámara proporcionados por Bundler y se añadirán a la reconstrucción . También debiera disponer de una visualización con más funcionalidad que las aplicaciones existentes. Éstas, aunque potentes, son insuficientes para validar visualmente los resultados obtenidos cuando éstos son de difícil reconocimiento. Además, es imprescidible desarrollar una aplicación de este tipo si no se quiere limitar la validación a un reconocimiento visual de los resultados. Para mejorar el problema de la poca cantidad de puntos, la única solución es usar vídeos de mayor resolución.

- La distancia entre el UAV y los objetos grabados es muy grande para este escenario, siendo superiores a los $1000m$. Hay que decir que los UAV pueden volar a grandes altitudes para tomar las imágenes. La distancia entre cámara y objeto influye en la calidad de la reconstrucción. Por un lado, la característica de la imagen se acerca más a propiedades afines que a propiedades del espacio proyectivo. Por otro lado, al tener distancias grandes, un pequeño error (por ejemplo de un píxel) suponde una distancia enorme en la reconstrucción. Esto afectaría en gran medida a la triangulación, y por ende, a la reconstrucción y obtención de los parámetros de calibración de las cámaras.

- Una combinación de los problemas anteriores puede causar errores en cascada y que el resultado final sea una reconstrucción deficiente.

Aunque aquí se comentan únicamente los resultados del escenario *ALCONADA*, se probaron también otros escenarios del SIVA con mejores resoluciones (720×576). Para éstos los resultados fueron incluso peores, al no disponer de un número suficiente de fotogramas para reconstruir edificios o escenarios. Como procedimiento adicional para mejorar

los resultados se añade el filtro a la secuencia de ejecución de Bundler. Los resultados se comentan en el siguiente apartado. Al final se verá que las imagenes disponibles del SIVA son insuficientes para dar unos resultados correctos. Esto no quiere decir que sea imposible reconstruir con los vídeos tomados con el SIVA. Tan sólo que con los fotogramas usados los resultados no son buenos. Pudiera ser que con otros escenarios, con mayores resoluciones y determinadas características de vuelo sea posible reconstruir, pero al no disponer de éstos no se ha podido comprobar.

5.3.2. Resultados con filtro

Como se comenta en el apartado anterior, la presencia de caracteres de información sobreimpresos en la imagen tiene un efecto negativo en la reconstrucción. En las figuras 5.12 y 5.13 se presentan dos ejemplos de puntos correspondientes erróneos detectados sobre caracteres. Como posible solución se plantea la aplicación de un filtro que discrimine aquellos puntos correspondientes que puedan ser erróneos. Este filtro se añade en la secuencia de Bundler, justo a la salida de la etapa de búsqueda de correspondencias y antes de comenzar la secuencia de reconstrucción. Es simplemente un pequeño programa en código C que abre un fichero con las correspondencias encontradas para las imágenes, guardadas en *matches.init.txt* y que apuntan a los puntos característicos descritos en los ficheros listados en *list_keys.txt* para cada una de las imágenes. Sea por ejemplo una correspondencia entre los puntos característicos x_i y x_j encontrados en las imágenes i y j, y sean id_i, id_j los identificadores de estos puntos (sus posiciones en el fichero de puntos característicos), el fichero *matches.init.txt* tendrá una entrada donde la primera línea serán los identificadores del par de imágenes, es decir i j, seguido de una línea con el número de correspondencias encontradas, y de los identificadores para cada una de éstas, en este caso id_i id_j.

Ahora tomando el fichero de característicos para la imagen i, listado en *list_keys.txt*, comprimido[5] y con el mismo nombre que la imagen pero con extensión *.key*, se busca el punto característico en la posición dada por el identificador id_i. Cada punto característico contiene su posición en la imagen, escala, rotación y un vector descriptor. El filtro toma esas posiciones y calcula la distancia entre ellas. Si la distancia es mayor que un umbral acepta la correspondencia, si es inferior la elimina. El umbral se define al llamar al filtro, siendo el valor por defecto un píxel. Las correspondencias de salida se guardan en el fichero $matches.filtered.txt$ que luego es leído por la etapa de reconstrucción. El fichero de salida del filtro $matches.filtered.txt$ tiene la misma estructura que *matches.init.txt*.

En las pruebas aquí presentadas se usa un umbral de un píxel, ya que se ha observado que los puntos correspondientes sobre los caracteres tienden a tener la misma posición en varias imágenes. Esta suposición es válida siempre que los fotogramas se tomen en movimiento. Si no lo hubiera, se descartarían todos los puntos. El peor escenario sería un vuelo circular, donde los puntos verdaderos en el centro de la imagen podrían ser detectados como erróneos. Dado que los fotogramas usados siempre tendrán movimiento y estarán suficientemente separados, y que el umbral usado es muy bajo, es poco probable que haya puntos correspondientes reales que sean eliminados erróneamente. Si los fotogramas empleados carecen de caracteres o de información sobreimpresa, no es necesario aplicar este filtro, es más, aún siendo aplicado, no tendrá ningún efecto sobre la salida.

[5] El fichero comprimido tendrá extensión *.key.gz*.

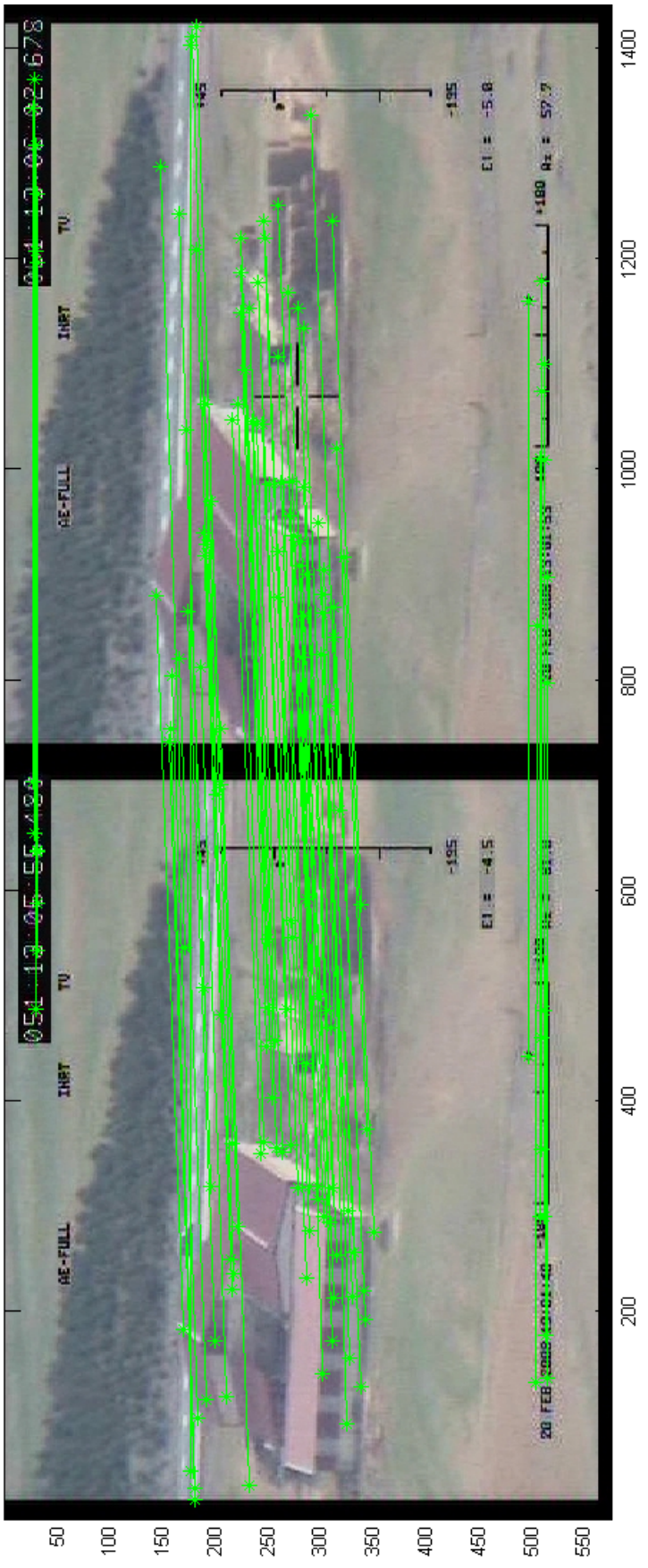

Figura 5.12: Correspondencias erróneas en GRANJA_1

Puntos correspondientes entre dos fotogramas (35,44) del escenario GRANJA_1. Algunas correspondencias tomadas como válidas son puntos detectados en los caracteres sobreimpresos en el vídeo. El resultado es una mala estimación de la relación entre cámaras. Fuente: e.p.

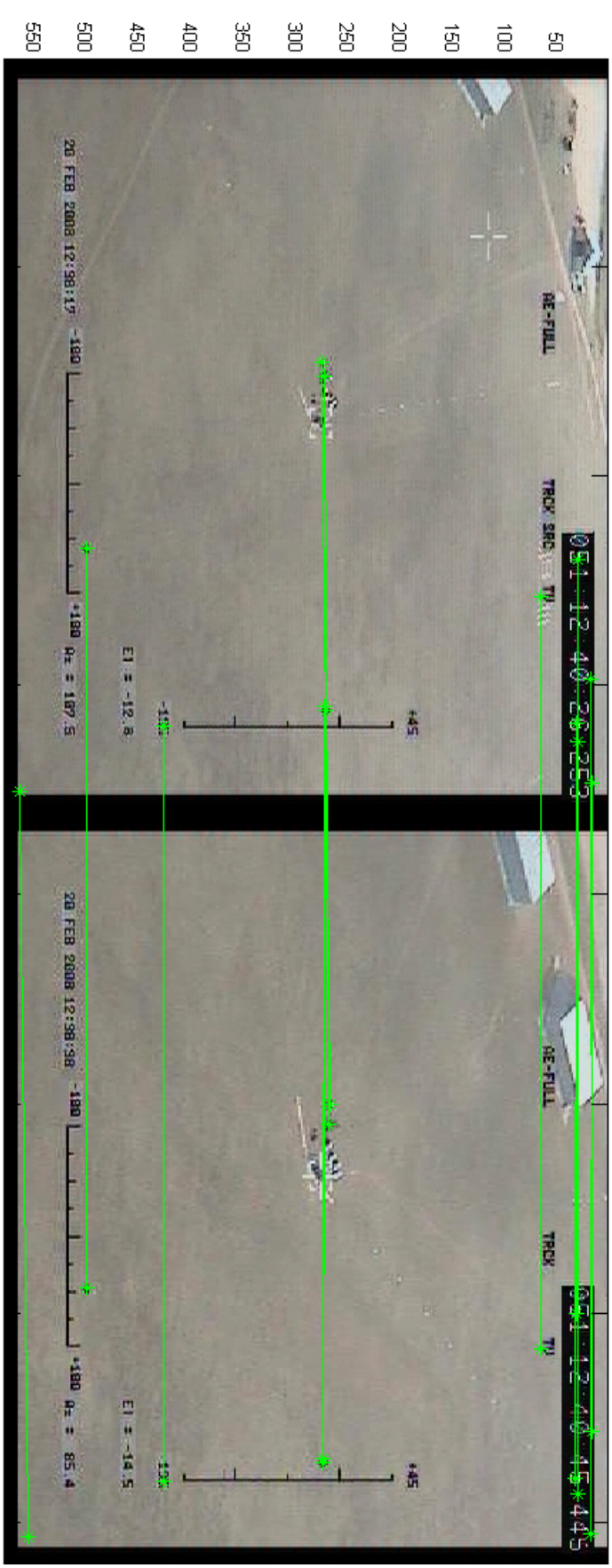

Figura 5.13: Correspondencias erróneas en AERODROMO_5

Como en la figura anterior, se presentan correspondencias erróneas en dos fotogramas (1,25), aunque este caso, el escenario AERODROMO_5 es más relevante todavía, al ser mayor el número de correspondencias falsas que el de verdaderas. Fuente: e.p.

Generalmente los vídeos tomados desde plataformas aéreas, salvo que tengan como objetivo la filmación de documentales, programas de televisión o similares, siempre tendrán información sobreimpresa referente a localización del vuelo, zoom, dirección del aparato u otros tipos de datos referentes a la operación del UAV. Un filtro de este tipo tiene gran interés ya que el objetivo final de este trabajo es precisamente emplear este programa para este tipo de vídeos.

Para utilizarlo en Bundler es necesario realizar algunas modificaciones al fichero *RunBundler.sh*:

1. Añadir después de
   TO_SIFT =$\$BASE\_PATH/bin/ToSift.sh$
   la línea
   $FILTER$ =$\$BASE\_PATH/bin/filter\_v0.1.exe$

2. Después de
   $\$MATCHKEYS$ *list_keys.txt matches.init.txt*
   añadir
   $\$FILTER$ *matches.init.txt matches.filtered.txt* 1

3. Por último, cambiar
   echo "$--$ *match_table matches.init.txt*">>*options.txt*
   por
   echo "$--$ *match_table matches.filtered.txt*">>*options.txt*

Incluso con el filtro, los resultados a la salida de los vídeos del SIVA no son mucho mejores, con la salvedad de que se han eliminado las correspondencias falsas. Los puntos de la nube son tan escasos que no se puede reconocer ningún elemento del escenario. Además, al triangular en Orto3D con puntos seleccionados maualmente no se obtienen buenos resultados. Las razones pueden ser varias, las identificadas son la línea base, muy pequeña con respecto a la profundidad del escenario, y la baja resolución de las imágenes. A modo de ejemplo de los resultados obtenidos con el filtro, se presentan en el apartado 5.4.1 los resultados obtenidos con el filtro y forzando la focal a un valor de $f = 1{,}2ancho = 422{,}4$ con $ancho = 352$ el ancho de la imagen.

5.3.3. Resultados con otros escenarios

Debido a la problemática con las imágenes del SIVA, fue necesario cambiar la planificación inicial del trabajo para usar imágenes diferentes a las disponibles del SIVA. El primer cambio consistió en dejar de trabajar con imágenes tomadas desde avión para trabajar con imágenes tomadas con cámaras fotográficas de edificios o similares. Por los datos publicados de Bundler y Photo Tourism se sabe que funciona correctamente con fotografías y con imágenes en las que las cámaras no estuviesen muy distantes de los objetos del escenario. Gracias a trabajar con esta configuración se pudieron obtener mejores resultados de reconstrucción y se pudo desarrollar correctamente la aplicación Orto3D. Con los vídeos iniciales, al no disponer de un escenario correcto, era imposible desarrollar correctamente Orto3D, ni siquiera fue posible reconocer objeto alguno en la reconstrucción. Sin embargo en las reconstrucciones con fotografías, los resultados fueron mucho mejores, incluso visualmente. El segundo cambio consistió en utilizar una segunda fuente de imágenes aéreas diferente del SIVA. El objetivo era usar cámaras de mayor calidad y con objetos más cercanos a las cámaras. La solución fue trabajar con vídeos aéreos tomados para televisión, que tienen generalmente mayor calidad que los tomados con UAV,

mayor estabilidad, mejor tipo de cámara, etc. Los resultados obtenidos con estas nuevas imágenes también fueron malos, al analizarlos se pudo comprobar que las estimaciones de las focales eran erróneas. Para evitarlo se siguió la misma aproximación que en el SIVA, y se forzaron sus valores durante la inicialización. Forzando las focales, los resultados fueron mejores, aunque todavía queda pendiente trabajar con calibraciones de cámara conocidas y posiciones GPS del vehículo (y por ende de la cámara) para analizar el error cometido, y la precisión de la reconstrucción con respecto al escenario real.

El prototipo Orto3D implementado se ha aplicado a algunos de los escenarios de pruebas, entre ellos las fotografías tomadas con una cámara digital (ver en sección 5.7.1). La ventaja de usar esta cámara con un escenario conocido con un edificio de planta fácilmente reconocible y en un rango de distancias pequeños, es que se puede probar a grandes rasgos la georreferenciación de la posición y la orientación de las cámaras sobre el mapa, descartando valores incoherentes. Como resumen, los resultados con fotografías de alta resolución (2272×1704) son satisfactorios. En ellos se pueden reconocer los objetos perfectamente y es muy sencillo trabajar con ortoimágenes para georreferenciar los escenarios si las imágenes tomadas son de estructuras o edificaciones. Esto es muy diferente de la visualización de los resultados del SIVA en Orto3D donde es imposible ver alguna estructura reconocible.

Volviendo a los vídeos aéreos para documentales, los fotogramas tienen una buena resolución y un rango muy inferior de distancias entre cámaras y objetos. Pese a que los vídeos son de mejor resolución(768×576) que los del escenario *ALCONADA*, también se ha encontrado el problema de la estimación de la focal. Las focales estimadas (sin ser forzadas) van desde valores negativos hasta valores muy dispares entre cámaras, usando siempre la misma cámara. El resultado son escenarios mal reconstruidos a causa de una mala inicialización, como se comenta en el apartado 5.3.6. El movimiento del vehículo y el rango de distancias también influye, ya que se han observado mejores resultados con grabaciones en vuelo cercano alrededor de un edificio y peores en movimiento *forward*. Los objetos reconstruidos no siempre son visualmente reconocibles, aunque algunos resultados son interesantes.

Siguiendo el mismo procedimiento aplicado a las imágenes del SIVA, se forzaron las focales de las imágenes. Tras esta modificación, los resultados fueron absolutamente mejores, pese a trabajar siempre con estimaciones de la longitud focal. Algunos de los valores de focal forzados se tomaron de la ejecución de Bundler ya realizada. De todos los valores encontrados se tomaron los más coherentes (no negativos, no nulos y no excesivamente mayores que el ancho de la imagen) y se inicializó la focal con un valor cercano a los valores más repetidos. Los resultados fueron buenos, aunque al desconocer los valores reales de la focal no se ha podido verificar su validez. La otra opción usada en la mayoría de los escenarios de documentales ha sido forzar el valor a 1,2 veces el ancho de la imagen. Igualmente será un valor estimado que puede no estar cerca de la solución real. En el siguiente apartado (5.3.4), se presentan los resultados con más detalle.

Estos resultados no fueron exactamente los esperados. Según la documentación de Bundler, el código permite registrar cámaras de forma fiable, y se esperaba obtener una estimación fiable de la longitud focal de las cámaras sin ninguna información a priori sobre el escenario. Sin embargo, ésto no se cumple en los resultados obtenidos. Para buscar una justificación, se revisó toda la documentación y se estudió la teoría de los algoritmos implementados en el código fuente. Así se llegó a la conclusión de que Bundler no permite obtener buenos

resultados si no se dispone de una estimación sobre las focales al menos del primer par de imágenes reconstruido. Toda la reconstrucción depende del primer par, de ahí que éste necesite ser el más robusto. Además, el algoritmo usado para recuperar los parámetros de las cámaras del primer par asume que las cámaras están calibradas, condición que no se cumple cuando no se dispone de éstas en la cabecera EXIF. El problema es menos importante para el resto de cámaras, ya que para ellas sí que se emplea un método que calcula la calibración. Otro asunto que no queda bien especificado en la documentación es que Bundler carece de un método de autocalibración. Revisando el código fuente se encontró que Bundler, cuando no dispone de cabecera EXIF, usa el valor $f = 532$ el cual puede no estar cerca del valor real. Al tener una mala inicialización genera un error en cascada que acaba provocando que la nube de puntos sea irreconocible y que la reconstrucción sea errónea. Para confirmar esto se tomaron las fotografías de POLITECNICA, se eliminaron sus cabeceras EXIF y se inicializó la reconstrucción a $f = 500$ píxeles para todas las cámaras, un valor lejano a la solución real. Luego se usó Bundler con ellas y se visualizaron los resultados. Éstos fueron peores que en el caso con EXIF, pero aún así el edificio era reconocible. Lo destacable es que al menos para este escenario con resolución (2272×1704), Bundler fue capaz de acabar la optimización de la focal en un valor relativamente cercano al valor real en comparación con los resultados con vídeos aéreos. El valor real incluido en la cabecera es de $f = 2287{,}77778$, el valor medio de las estimaciones de Bundler con la cabecera EXIF es de $f = 2367$ y el valor medio estimado por Bundler al forzar a $f = 500$ píxeles es de $f = 1978$, con un error de un 13,54 % sobre el valor calculado a partir de la cabecera. Pero claro, en un contexto muy diferente al de las imágenes aéreas. En el siguiente apartado se presentan gráficas con los valores de focal recuperados.

Uno de los problemas de SfM es el efecto debido a las derivas en la reconstrucción. Éste se comenta en [68] y se presenta en detalle en [12]. Los errores acumulados durante la reconstrucción pueden causar pequeños desplazamientos en las coordenadas estimadas de los puntos y de las cámaras. Pese a no tener un efecto importante en la visualización, sí es problematico cuando se busca un modelo preciso del escenario. Bundler ha sido usado para escenarios con muchas más cámaras, como se puede ver en [3][4], aunque todos los escenarios presentados en este trabajo se han reconstruido con pocas imágenes en comparación con los anteriores, nunca con un número superior a 100 . Aunque este aspecto no se ha comprobado en este trabajo, puede ser interesante dedicarle un tiempo a ello como trabajo de continuación.

Por último, comentar que casi todos los escenarios usados pertenecen a zonas urbanas con edificios y estructuras. Tan sólo se han usado cuatro escenarios sin ningún tipo de estructura, simplemente para visualizar el resultado de Bundler en este contexto concreto, aunque debido a la dificultad, no se han georreferenciado con Orto3D. En la figura 5.14 se presenta un fotograma del vuelo y en la 5.15 una imagen de la reconstrucción del escenario CARCAVAS_MOD_1. Como se puede apreciar, es interesante cómo Bundler ha sido capaz de reconstruir la forma de los cerros. Aunque no se ha probado, podría estudiarse el uso de Bundler combinado con modelos de terreno en tres dimensiones. En la documentación de Bundler se hace referencia al alineamiento de las nubes de puntos a DEM [70].

Figura 5.14: Fotograma del río Henares

Vuelo siguiendo el curso del río Henares, grabación de los cerros del margen izquierdo. Fuente: documentales *Madrid desde el aire*.

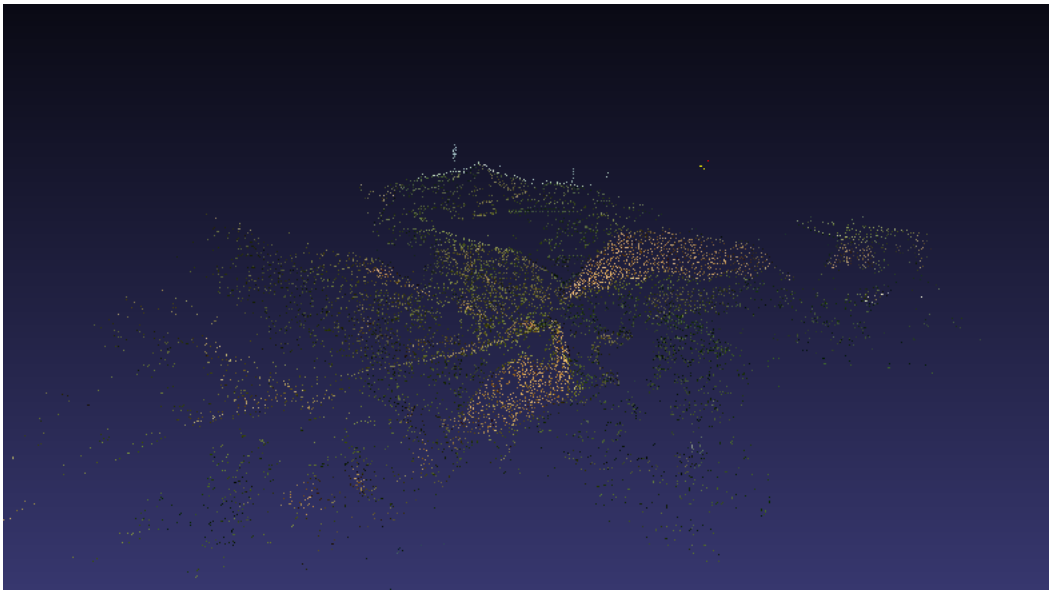

Figura 5.15: Reconstrucción de los cerros

Vista frontal de los cerros reconstruidos. Fuente: e.p.

5.3.4. Estimación de focales

Para inicializar las cámaras es necesario disponer de una estimación de la longitud focal. Ésta es indispensable para el primer par de cámaras reconstruidas, aunque también es conveniente para el resto. De hecho Bundler siempre usa alguna estimación, ya sea el de la cabecera EXIF de las imágenes o un valor por defecto. El primer par se calcula a partir del método de Níster, que recupera la pose de dos cámara asumiendo que están calibradas. Por esta razón es necesario conocer la focal para el primer par. Los parámetros del resto de cámaras se calculan mediante la DLT. De su resultado se puede obtener la matriz de calibración y de ahí la longitud focal. Aunque la focal se puede calcular directamente por este método, Bundler se apoya igualmente en un valor inicial para obtener una buena estimación. Estos valores iniciales se toman de las cabeceras EXIF de las cámaras siempre y cuando éstos están disponibles.

La extracción de la focal de la cabecera EXIF se realiza mediante el script $extract\_focal.pl$. Se trata de un script en Perl que contiene una lista de cámaras digitales compactas y réflex de público general con las anchuras del sensor CCD de cada una. Hay un gran número de cámaras en la lista, pero seguro faltan muchas otras ya que las contenidas son una selección de las más comunes. Completar la lista con nuevas cámaras es un proceso sencillo consistente en añadir una línea con el modelo de cámara tal como aparece en la cabecera EXIF y su ancho de CCD. Cuando el script toma una imagen, extrae su cabecera EXIF con *jhead.exe* y de ésta lee el modelo de cámara y la focal en mm. El nombre del modelo lo compara con su lista de cámaras y si la encuentra, toma el ancho de CCD de ésta guardado en la lista. La focal en píxeles se puede tomar de la siguiente ecuación:

$$f_{pixels} = res_x(\frac{f_{mm}}{ancho\_CCD_{mm}}) \tag{5.25}$$

donde f_{pix} es la focal en píxeles, res_x la resolución pixélica de la imagen en la horizontal (ancho de la imagen)[6], f_{mm} la focal en mm y $ancho\_CCD_{mm}$ el ancho del CCD en mm. La extracción de los datos de cabecera la realiza el ejecutable $jhead.exe$ ($jhead$ en Linux) que se incluye en la carpeta *bin* de Bundler. Cuando no encuentra el modelo de cámara en la lista, intenta cargar la contenida en la cabecera EXIF, y si no encuentra ninguna, entonces asigna a esta focal un valor nulo. Luego internamente en Bundler se verá se inicializa a otro valor no nulo por defecto.

Para aclarar la extracción de la focal se va a realizar el cálculo para dos imágenes, presentadas en las figuras 5.16 y 5.17 y pertenecientes al escenario HANGAR (del antiguo hangar militar del Campus Externo de la UAH). Las dos imágenes se tomaron con la cámara usada para pruebas desde la misma posición y usando la focal mínima y máxima de la cámara (zoom mínimo y máximo). De las especificaciones se sabe que el valor mínimo corresponde a 5,8mm y el valor máximo a 23,2mm. La resolución pixélica empleada es de 2272×1704 píxeles, donde el ancho de imagen es 2272 píxeles. En las tablas 5.3 y 5.4 se muestra la información obtenida al extraer las cabeceras EXIF de ambas imágenes con $jheah.exe$.

[6]También es el ancho en píxeles del CCD.

Figura 5.16: Hangar de la UAH con zoom mínimo

Imagen tomada con la cámara de pruebas configurada con el zoom mínimo, correspondiente a una longitud focal de 5,8mm. Fuente: e.p.

Figura 5.17: Hangar de la UAH con zoom máximo

Imagen tomada con la cámara de pruebas configurada con el zoom máximo, correspondiente a una longitud focal de 23,2mm. Fuente: e.p.

File name	IMG_0478.jpg
File size	938032 bytes
File date	2009:09:27 17:48:33
Camera make	Canon
Camera model	Canon PowerShot A520
Date/Time	2009:09:27 17:48:33
Resolution	2272 x 1704
Flash used	No (auto)
Focal lenght	5.8mm (35mm equivalent: 37mm)
CCD width	5.69mm
Exposure time	0.017s (1/60)
Aperture	f/3.5
Focus dist.	9.06m
Whitebalance	Auto
Metering Mode	matrix

Tabla 5.3: Cabecera EXIF de imagen con zoom mínimo

File name	IMG_0485.jpg
File size	1203562 bytes
File date	2009:09:27 17:49:22
Camera make	Canon
Camera model	Canon PowerShot A520
Date/Time	2009:09:27 17:49:21
Resolution	2272 x 1704
Flash used	Yes (auto, red eye reduction mode)
Focal lenght	23.2mm (35mm equivalent: 147mm)
CCD width	5.69mm
Exposure time	0.017s (1/60)
Aperture	f/5.5
Focus dist.	8.21m
Whitebalance	Auto
Metering Mode	matrix

Tabla 5.4: Cabecera EXIF de imagen con zoom máximo

Para la primera imagen conocemos que $f_{mm} = 5{,}8mm$, y que $ancho\_CCD_{mm} = 5{,}76mm$. Como valor de ancho de CCD se ha tomado el asignado por el extractor de focal de Bundler, pero como se puede ver en la tabla 5.4, el valor de éste es diferente en la cabecera. La diferencia es de $0{,}07mm$. A pesar de que para el siguiente cálculo se tome el valor anterior para el ancho de CCD en vez de el de la cabecera EXIF, más adelante se comentará algo más sobre este asunto. Entonces tomando este valor la focal en píxeles será

$$f_{pix} = 1{,}006944445 res_x \tag{5.26}$$

y tendrá los siguientes valores para cada una de las resoluciones de la cámara empleada (tiene tres opciones de resolución):

- para $res_x = 640$ píxeles, $f_{pix} = 644{,}44444$
- para $res_x = 1024$ píxeles, $f_{pix} = 1031{,}11111$
- para $res_x = 2272$ píxeles, $f_{pix} = 2287{,}77778$

Ahora tomando la segunda imagen, si la focal máxima es $f_{mm} = 23{,}2mm$ la focal en píxeles vendrá determinada por,

$$f_{pix} = 4{,}027777778 res_x \tag{5.27}$$

y tendrá los siguientes valores para la cámara:

- para $res_x = 640$ píxeles, $f_{pix} = 2577{,}77778$
- para $res_x = 1024$ píxeles, $f_{pix} = 4124{,}44445$
- para $res_x = 2272$ píxeles, $f_{pix} = 9151{,}11111$

Para las imágenes mostradas de ejemplo las focales serán el tercer caso. Si se abre el fichero *list.txt* se pueden comprobar que estas ambas se han inicializado así.

Volviendo a Bundler, una forma de inicializar los parámetros de las cámaras según el autor, es usar un valor de focal tal que

$$f_{pix} = 1{,}2 res_x \tag{5.28}$$

Esta inicialización estará dentro del rango de focales de la mayoría de las cámaras. Por ejemplo para la cámara de pruebas, el valor mínimo y máximo de la focal viene determinado por el rango de zoom, que para la cámara de pruebas es $(1{,}006944445 res_x, 4{,}027777778 res_x)$. Como se puede comprobar, el valor anterior está dentro del rango por lo que es una posible inicialización. Para este valor, si el ancho es 2272 píxele, la focal será $f_{pix} = 2726{,}4$ píxeles y su valor en mm será

$$f_{mm} = 1{,}2 * 5{,}76 = 6{,}912mm \tag{5.29}$$

Volviendo al tema de la diferencia entre el ancho de CCD de la cabecera EXIF y del script de Bundler, la diferencia entre usar un valor u otro vendrá dada por

$$error\_f_{pix} = res_x f_{mm}\left(\frac{1}{ancho\_EXIF_{mm}} - \frac{1}{ancho\_CCD_{mm}}\right) \approx 0{,}002136 res_x f_{mm} \tag{5.30}$$

Para el caso de $f_{mm} = 5{,}8mm$, $error\_f_{pix} = 28{,}147$ píxeles, y para $f_{mm} = 23{,}2mm$, $error\_f_{pix} = 112{,}589$ píxeles que son un 1,24 % y 4,96 % del ancho de la imagen respectivamente. Habría que evaluar el efecto que puede tener en la reconstrucción, y ver de dónde viene esta diferencia. Por no ser el caso central del trabajo no se ha continuado con esta comprobación. Pudiera ser que los anchos de CCD guardados sean erróneos para alguna de las cámaras de la lista, o que en ésta se empleen valores teóricos en base al tipo de CCD usado (determinado por las pulgadas).

Como se ha comentado previamente, los valores de las focales para cada cámara se inicializan a partir de los valores almacenados en la etiqueta EXIF. Cuando no se dispone de la cabecera EXIF, este valor se inicializa a 0, que luego se iguala a $f = 532$ internamente en el código de Bundler. Por tanto, cuando no se dispone de información de la focal, la cámara se fuerza a un valor arbitrario que no tiene por qué estar cerca de la solución final

y que puede caer en una solución no óptima durante la optimización. Este es uno de los problemas caracterizados con los escenarios de vídeo. Los fotogramas, al no disponer de cabeceras EXIF, se inicializan a $f = 532$. Tras una primera iteración en algunos escenarios, éstos valores caen en valores negativos, como se puede apreciar en las figuras 5.18, 5.22 y 5.20. Para evitar este problema, se fuerzan las focales a un valor concreto. En este caso los resultados son significativamente mejores, como se pueden ver en las figuras 5.19, 5.21 y 5.23. Esto se hace asumiendo que a lo largo de toda la secuencia se emplea la misma cámara, y que esta cámara tiene configuraión constante. En otras palabras, que no usa el zoom en esta secuencia. En las figuras 5.18, 5.22 y 5.20 se puede ver el valor medio de la focal. Éste valor se calcula usando un rango de valores válidos para el tamaño de la imagen, tomando únicamente valores absolutos y descartando valores nulos y excesivamente altos (ver sección B.6 en los apéndices). Para el caso de la figura 5.20 del escenario TORRE_PICASSO_AZCA_MOD_1, una de las focales estimadas tiene un valor muy dispar en comparaciçon con el resto. Este valor es del orden de 4×10^6 lo cual indica que es una estimación errónea. Cuando se ejecuta el mismo escenario pero esta vez con una focal inicial forzada, los valores quedan más próximos entre sí. Lo mismo ocurre en el escenario de MECO_3_MOD_1.

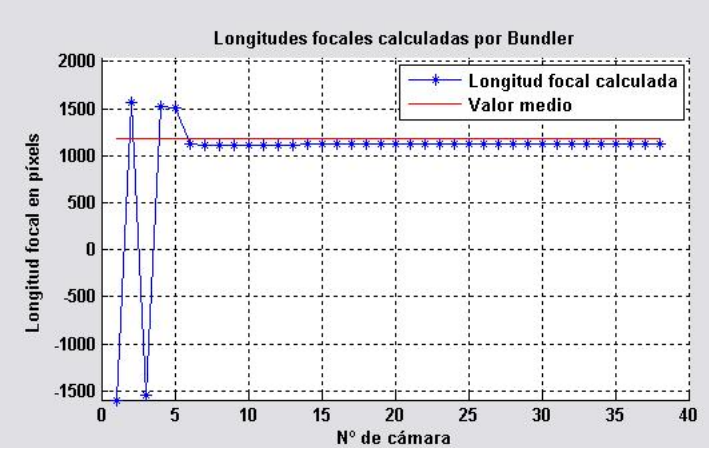

Figura 5.18: Estimación de focal en ALCALA_2

Estimación de la distancia focal realizada por Bundler, sin tener información de cabeceras EXIF. Las focales estimadas son erróneas, con valores muy dispares y algunos negativos. Fuente: e.p.

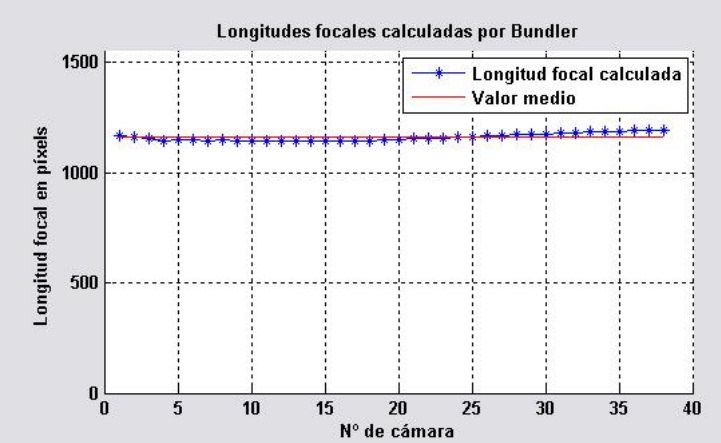

Figura 5.19: Estimación de focal en ALCALA_2 con valor forzado

Estimación de la distancia focal forzando el valor inicial. Las focales tienen valores coherentes y cercanos entre sí. Fuente: e.p.

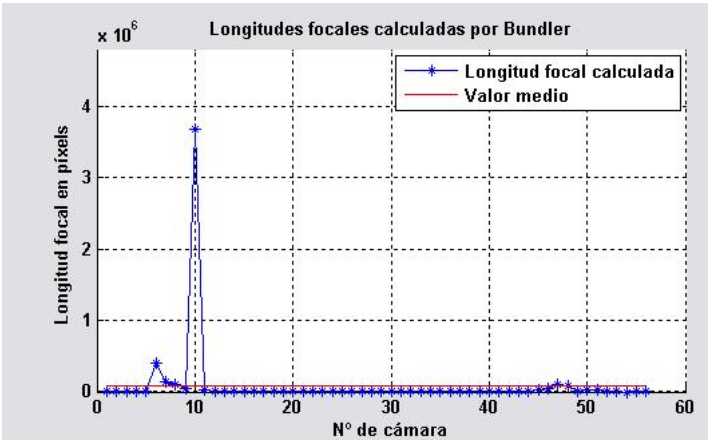

Figura 5.20: Estimación de focal en TORRE_PICASSO_AZCA

Estimación de la distancia focal sin información previa sobre la cámara. Los valores obtenidos son erróneos. Fuente: e.p.

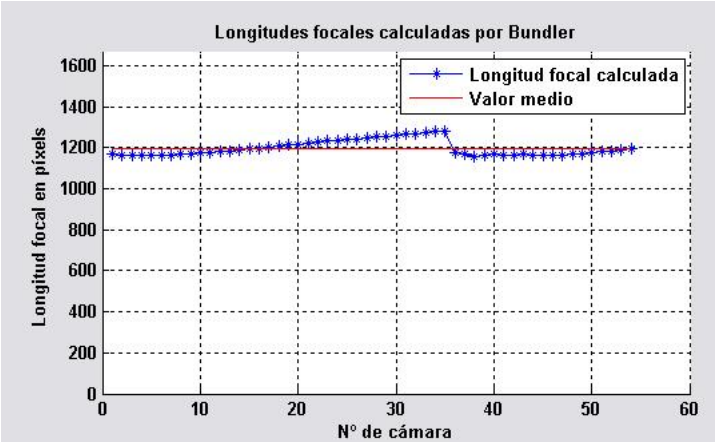

Figura 5.21: Estimación de focal en TORRE_PICASSO_AZCA con valor forzado

Valor de la focal forzado a una estimación. Los valores obtenidos son coherentes y cercanos entre sí. Fuente: e.p.

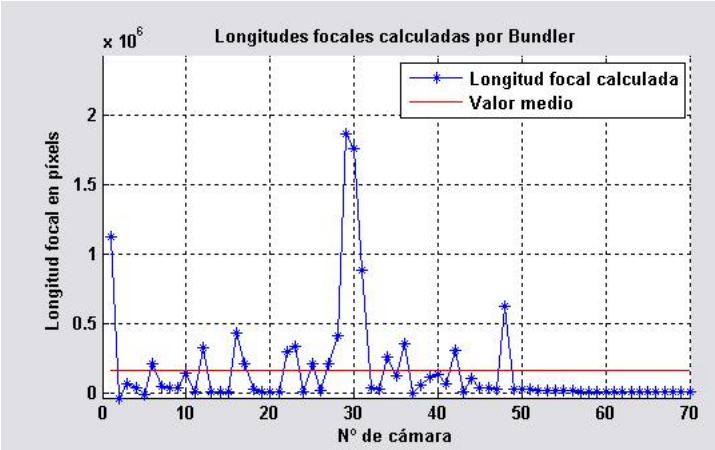

Figura 5.22: Estimación de focal en MECO_3

Estimación de la focal sin disponer de un valor inicial. Resultados incoherentes con el escenario. Fuente: e.p.

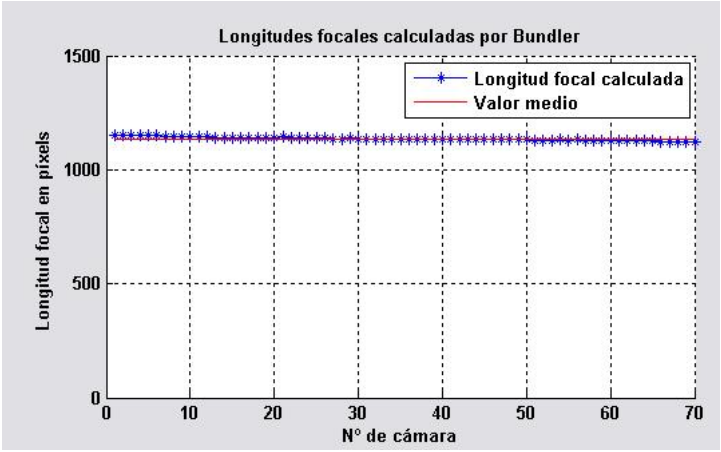

Figura 5.23: Estimación de focal en MECO_3 con forzado

Estimación de la focal partiendo de un valor forzado. Los resultados son cercanos entre sí y coherentes. Fuente: e.p.

Para dar una idea de la evolución de la focal durante la ejecución y comprobar cómo esta cae en valores negativos, se muestran una serie de figuras a continuación. En estas figuras se representa únicamente la evolución del primer par[7]. Para cada figura se añade un comentario sobre la misma. A grandes rasgos, se puede ver que los resultados para el primer par son mucho mejores cuando se dispone de valores inciales para la focal.

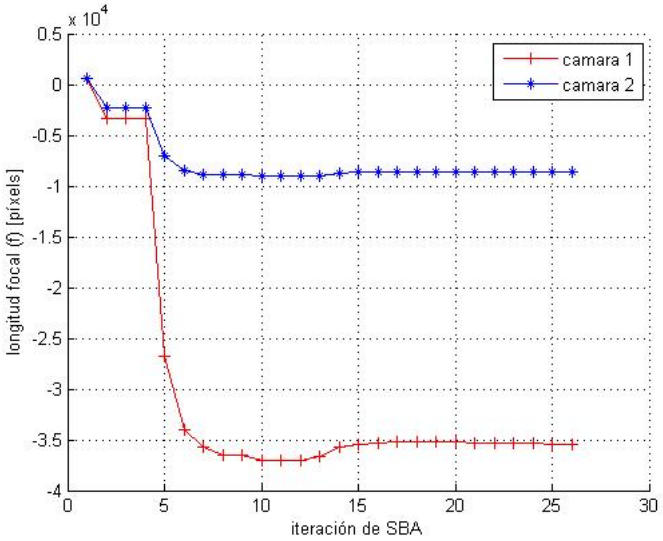

Figura 5.24: Evolución del par inicial en MECO_3

Valores asignados a la focal del par inicial a lo largo de la reconstrucción. La optimización con SBA comienza con $f = 532$, pero desde la primera iteración cae en valores negativos. Fuente: e.p.

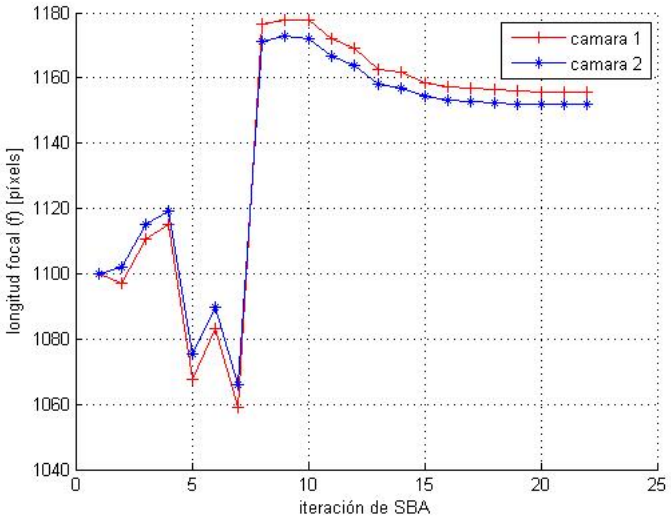

Figura 5.25: Evolución del par inicia en MECO_3_MOD_1

La optimización comienza forzando el valor inicial a f=1100, que en las sucesivas iteraciones de SBA se va modificando en un rango entre [1160,1180], y que en las últimas iteraciones se estabiliza hacia el valor final. Fuente: e.p.

[7]Esta funcionalidad de representación no está incluida en la aplicación Orto3D aunque puede añadirse en nuevas versiones.

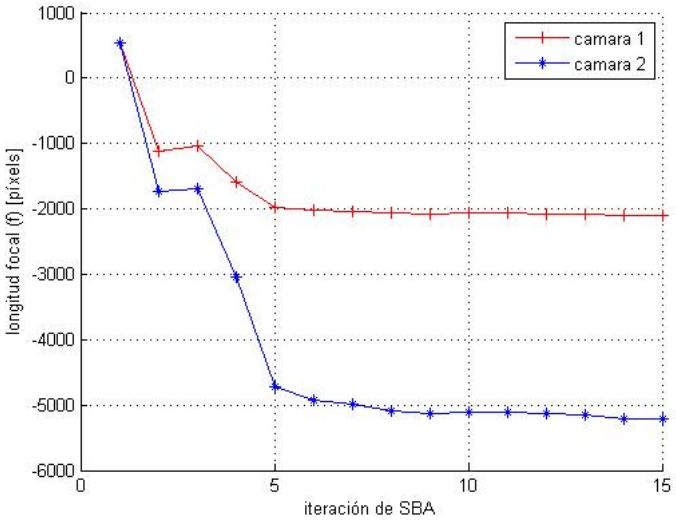

Figura 5.26: Evolución del par inicial en TORRE_PICASSO_AZCA

Valores asignados a la focal del par inicial a lo largo de la reconstrucción. La optimización comienza con f=532, pero desde la primera iteración cae en valores negativos. Fuente: e.p.

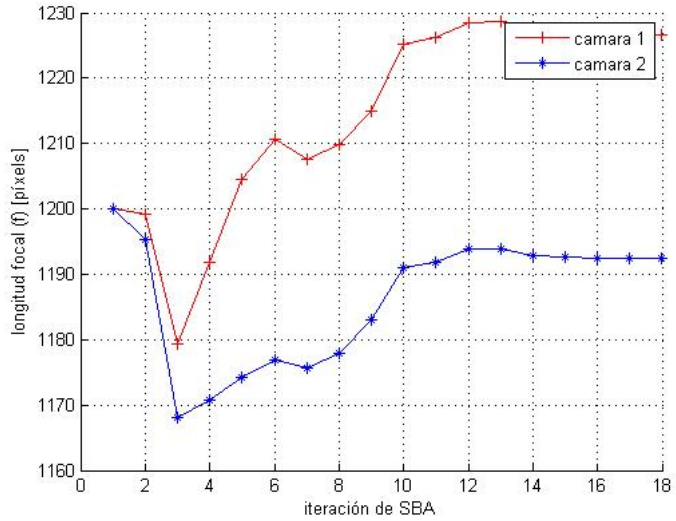

Figura 5.27: Evolución del par inicial en TORRE_PICASSO_AZCA_MOD_1

La optimización comienza forzando el valor inicial a f=1200, que en las sucesivas iteraciones de SBA se va modificando en un rango entre [1165,1230], y que en las últimas iteraciones se estabiliza hacia el valor final. Fuente: e.p.

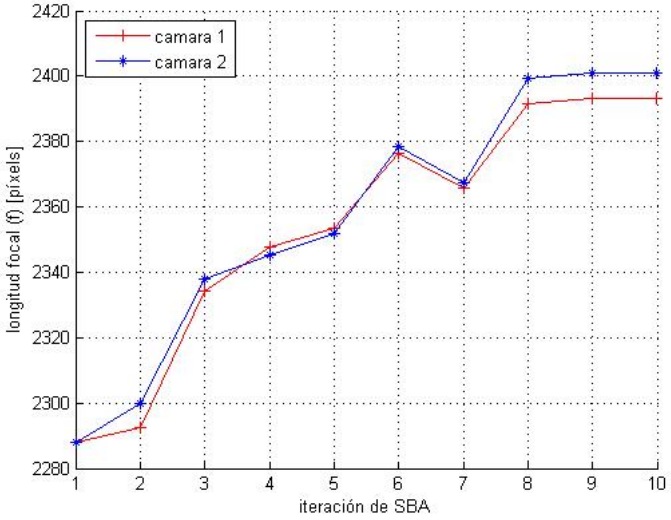

Figura 5.28: Evolución del par inicial en POLITECNICA con EXIF

Las imágenes se tomaron con una cámara conocida cuyos datos se almacenan en la etiqueta EXIF asociada a la imagen. La reconstrucción comienza con el valor de longitud focal f=2287.78, calculado a partir de los datos de la cámara. Fuente: e.p.

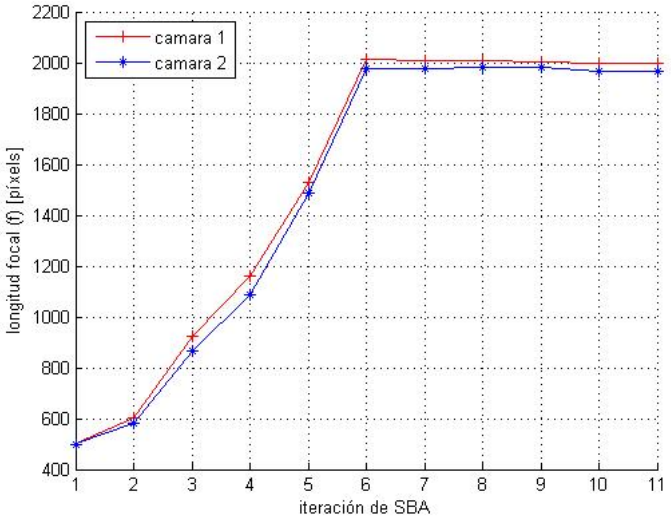

Figura 5.29: Evolución del par inicial en POLITECNICA sin EXIF

En las mismas imágenes usadas para la anterior gráfica se eliminan las etiquetas EXIF y se ejecuta Bundler. En la primera iteración la focal es f=532 (valor por defecto) y tras cada iteración los valores optimizados se acercan al valor teórico (f=2287.78) pero quedan por debajo de éste valor alrededor de f=2000. A diferencia de los fotogramas de vídeo, el carecer de datos EXIF no provoca que las focales sel primer par caigan en valores negativos. Fuente: e.p.

5.3.5. Distorsión radial en Bundler

Bundler estima los parámetros de distorsión radial de las imágenes durante la etapa de reconstrucción de estructura. Se puede indicar que estime estos valores o que no lo haga, aunque en los resultados aquí presentados se ordena a Bundler que siempre las estime y sin restricciones. A pesar de que se usan fotogramas tomados con la misma cámara, los resultados son muy variables. La distorsión radial es un efecto que es idéntico para una cámara y que tiene que ver con la lente y no con el fotograma ni el escenario en sí.

En las figuras 5.30, 5.31 y 5.32 se pueden ver tres formas de representar la problemática de la distorsión radial en un fotograma. En el primero se muestran en rojo las posiciones de unos puntos sobre la imagen, y en amarillo esos mismos puntos pero corregidos a partir de los parámetros de distorsión estimados por Bundler. En la segunda figura se hace una representación similar, siendo el vector azul sobreimpreso en la imagen la distancia entre el punto con distorsión y sin distorsión (corregido). Como se puede ver, los puntos en los extremos de la imagen se verán más afectados que los puntos de la zona central. Por último, en la tercera figura se presentan las imágenes originales y corregidas en dos canales (rojo y verde). Las dos primeras figuras han sido generadas en Matlab, mientras que la tercera se ha creado usando ImageJ. La versión $v0.3$ incluye el ejecutable *RadialUndistort.exe*, que permite obtener las imágenes corregidas a partir de los parámetros de distorsión estimados por Bundler.

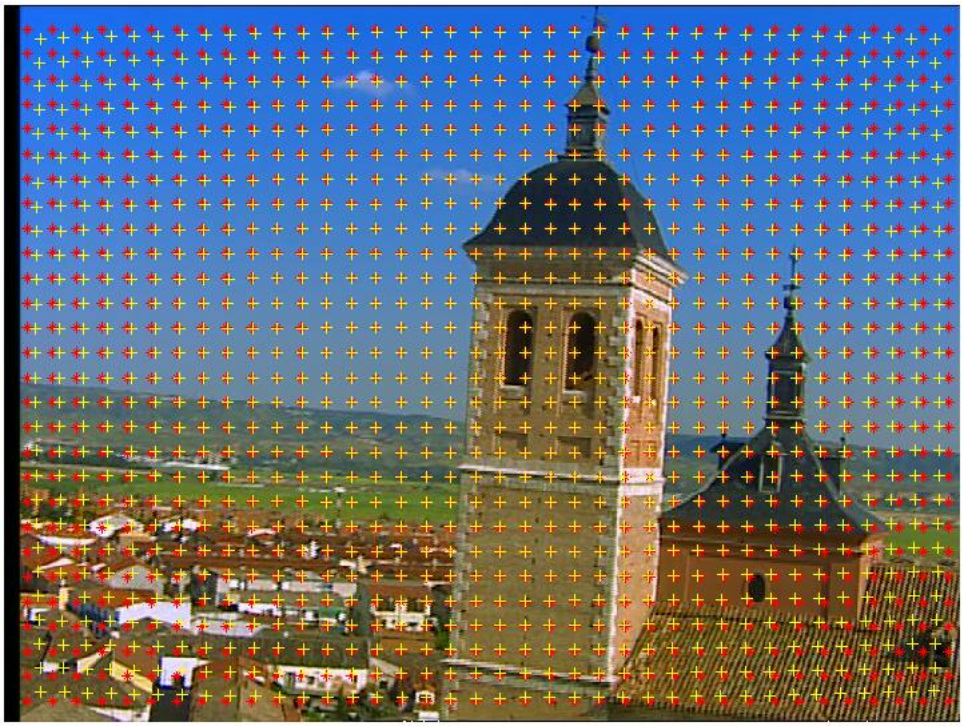

Figura 5.30: Distorsión radial (1)

Muestra de la posición de los píxeles en la imagen en función de la distorsión radial. En rojo una muestra de píxeles en la imagen original, y en amarillo esos mismos píxeles corregidos. Fuente: e.p.

Figura 5.31: Distorsión radial (2)
Muestra del desplazamiento de los píxeles provocado por la distorsión radial. Fuente: e.p.

Figura 5.32: Distorsión radial (3)
Superposición de la imagen con distorsión radial y la imagen corregida. Se ha empleado el canal rojo para representar la imagen con distorsión radial y el canal verde para representar la imagen corregida con los parámetros estimados por Bundler. Fuente: e.p.

En el resto de figuras del apartado se presentan las estimaciones de distorsión radial realizadas por Bundler para tres escenarios en dos casos, sin usar un valor inicial de focal y usándolo. Las tres primeras figuras, 5.33, 5.34 y 5.35 corresponden al escenario MECO_3 de los vídeos documentales, en los que no se ha empleado una estimación de la longitud focal. Para inicializar la reconstrucción, Bundler ha tomado el valor por defecto. Éste, usado cuando no se dispone de información EXIF, es $f = 532$. Comentando más en detalle las tres figuras, la primera (figura 5.33) muestra los valores estimados para cada cámara, con los identificadores de cámara en el eje de abscisas y los valores en el de ordenadas. Se representan los parámetros k_1 y k_2 así como sus valores medios. Como se puede apreciar, los valores de k_2 varían poco, siendo la mayoría nulos. En la figura 5.34 se puede ver otra forma de representar los mismos valores, esta vez se presentan en dos histogramas, uno para los parámetros k_1 y otro para los k_2. Como se puede apreciar, una gran cantidad de las cámaras tienden a tener valores nulos o cercanos, aunque también hay valores estimados muy dispares. Lo segundo no es coherente con el hecho de que se esté usando la misma cámara para la secuencia usada en el escenario. En la figura 5.35 se presenta la evolución de los valores de los parámetros para las cámaras del primer par. Como ya se ha comentado en anteriores secciones, el primer par es el más relevante de la reconstrucción. En el eje de abscisas se representa la iteración de SBA mientras que en el de ordenadas se presenta el valor de los parámetros. Como se puede apreciar, en la primera iteración los parámetros k_1 y k_2 se inician a valores nulos, que terminan convergiendo tras siete iteraciones.

Las tres figuras siguientes son las correspondientes a las imágenes del mismo escenario[8] pero forzando la inicialización de la focal a una estimación diferente al valor por defecto de Bundler. En este caso se ha tomado $f = 1100$ píxeles. Más en detalle, la figura 5.36 presenta los valores de los parámetros de distorsión estimados para las cámaras. Como se puede ver, la variación de las estimaciones es mayor que en el ejemplo anterior, pero como se aprecia en la figura 5.37, el rango de los valores de k_1 es menor que en el caso anterior. Sin embargo el rango de variación de los valores de k_2 es mayor. En cuanto a la evolución de los parámetros del primer par en la optimización SBA, se puede ver en la figura 5.38 que éstos convergen a un valor estable tras ocho iteraciones.

Las siguientes seis figuras presentan los resultados con otro de los escenarios tomados de los documentales, de nombre TORRE_PICASSO_AZCA. Las tres primeras corresponden a la reconstrucción sin forzado de focal y las tres siguientes a la reconstrucción con un forzado de focal de $f = 1200$ píxeles. En estos casos, la evolución de los parámetros también converge en la etapa de optimización, como se aprecia en las figuras 5.41 y 5.44. De las figuras 5.39 y 5.42 se puede ver que en la primera los valores de k_1 varían de positivos a negativos para las cámaras, y en la segunda son valores negativos más estables excepto para las cámaras 36, 37 y 38 que se desmarcan del resto. Los valores de k_2 vuelven a ser más estables en el primer caso y menos en el segundo, aunque el valor de este parámetro es muy pequeño. Lo interesante de estos resultados es que para ninguno de los dos escenarios con sus dos configuraciones (sin y con focal) se obtienen estimaciones de los parámetros similares para todas las cámaras. En algún caso como el del escenario TORRE_PICASSO_MOD_1 (con forzado), presentado en la figura 5.43, la mayoría de los

[8]Para no confundir el escenario original con aquél cuya focal se ha forzado a un valor, se añade el sufijo _MOD_n al nombre del escenario, de ahí que el escenario MECO_3 con focal forzada pase a llamarse MECO_3_MOD_1. Esta notación se sigue para el resto de escenarios incluidos en el DVD adjunto, con n un identificador de la modificación. Generalmente $n = 1$, pero en algunos escenarios donde se ha probado con más de un valor de focal diferente pudiera ser otro número.

parámetros k_1 son muy similares, pero algunos concretos se desmarcan del resto. Teniendo en cuenta que la cámara es la misma para toda la secuencia de fotogramas del escenario, sin apreciarse un cambio de zoom evidente, estos resultados indican que Bundler no es capaz de recuperar el valor de distorsión de la cámara de forma fiable, siendo peor el caso en el que no se dispone de una focal de inicialización.

Para probar el mismo problema con otro escenario un poco más sencillo que los anteriores de los documentales, se ha tomado el escenario de fotografías POLITECNICA. Este escenario ha sido tomado con una cámara digital compacta concreta descrita en el apartado 5.7.1. Para este caso los valores estimados de los parámetros de cámara deben ser muy similares. Las tres primeras figuras presentan los resultados de las estimaciones usando las focales grabadas en las cabeceras EXIF de las imágenes. Con el objeto de conocer el efecto que tiene la ausencia de este dato en las reconstrucciones, se han tomado las mismas imágenes y se ha eliminado de sus cabeceras EXIF la información relativa al fabricante, el modelo de cámara, el ancho del chip CCD y la focal en mm. En las figuras 5.45 y 5.46 se puede ver que para el primer caso, los valores estimados son bastante estables y similares para todas las cámaras. Hay una cierta disparidad, pero no tan acusada como en los escenarios anteriores de los documentales. La evolución del primer par converge rápidamente a un valor estable tras cuatro iteraciones como se puede comprobar en 5.47 y en esta evolución, los valores de k_1 para ambas cámaras (rojo y azul en la figura) son muy similares, y tienden a estabilizarse a valores próximos. Lo mismo ocurre con los valores de k_2 (verde y negro). Comparando con los resultados anteriores de los documentales, tan solo los escenarios con focales forzadas tienen una caracerística similar en la evolución de los parámetros del primer par. En los escenarios sin forzado los valores de k_1 para cada cámara son muy diferentes, especialmente para TORRE_PICASSO_AZCA (figura 5.41). Esto reafirma lo comentado anteriormente sobre el problema de no disponer de una focal de inicialización. Cuando no se dispone de esta focal y se usa el valor por defecto, Bundler da resultados erróneos e inestables para el primer par, que se arrastran a toda la reconstrucción. Como se ha comentado, las tres últimas figuras corresponden al escenario POLITECNICA sin la cabecera EXIF (figuras 5.48, 5.49, y 5.50), donde se puede ver de nuevo este problema. Los resultados son significativamente peores en comparación con los del mismo escenario con la cabecera EXIF. Resumiendo, con los resultados aquí mostrados, se reafirma que la ausencia de una estimación de focal fiable influye en la estimación de los parámetros de las cámaras, confirmando los malos resultados de las reconstrucciones cuando no se usa una focal de inicialización diferente de la por defecto. Los parámetros de distorsión, aunque sean de pequeña magnitud influyen en la triangulación de los puntos y por ende en los resultados de la reconstrucción, siendo el parámetro $k1$ el más relevante al tener mayor magnitud que $k2$.

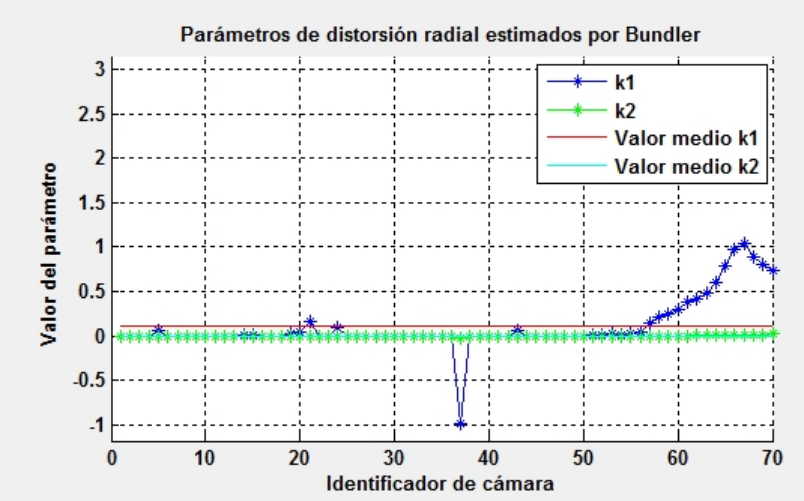

Figura 5.33: Distorsión radial en MECO_3

Fuente: e.p.

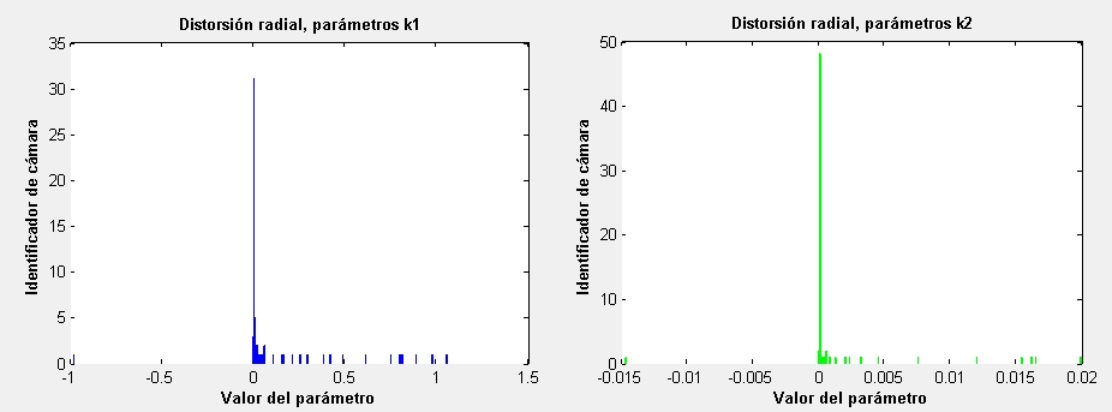

Figura 5.34: Distribución de la distorsión radial en MECO_3

Fuente: e.p.

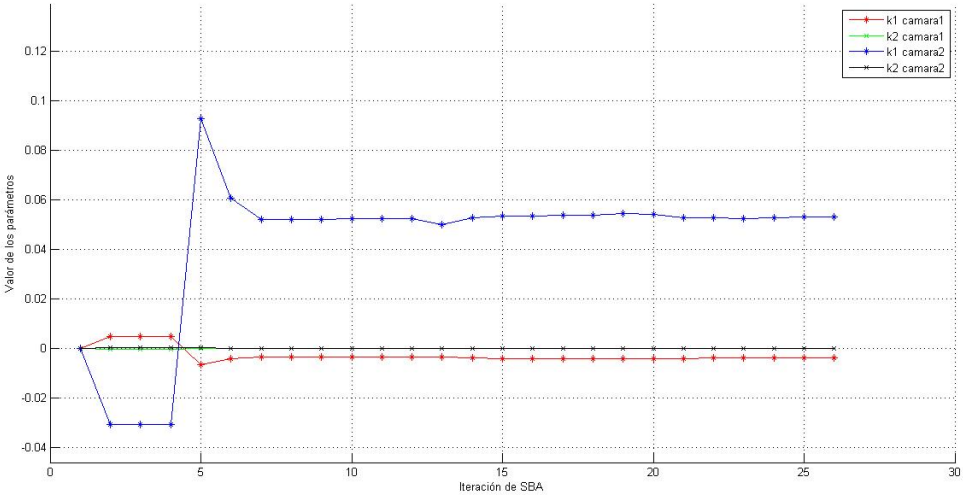

Figura 5.35: Evolución de la distorsión en el primer par de MECO_3

Valores de los parámetros de distorsión radial calculados en cada iteración SBA. Fuente: e.p.

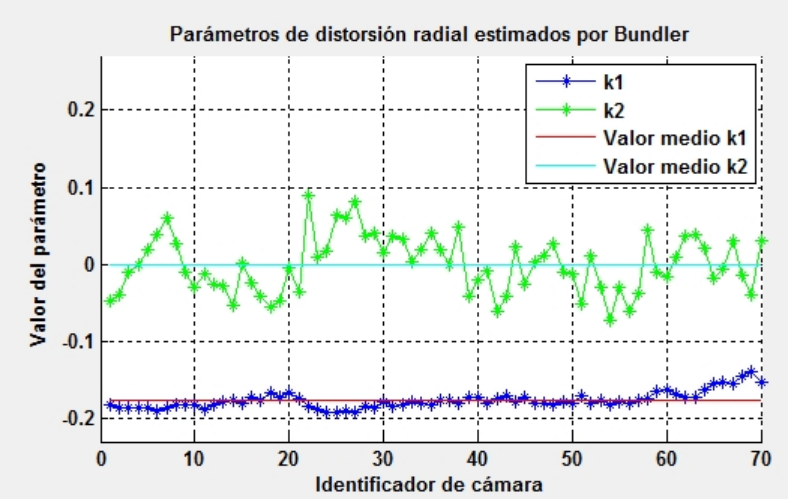

Figura 5.36: Distorsión radial en MECO_3_MOD_1

Fuente: e.p.

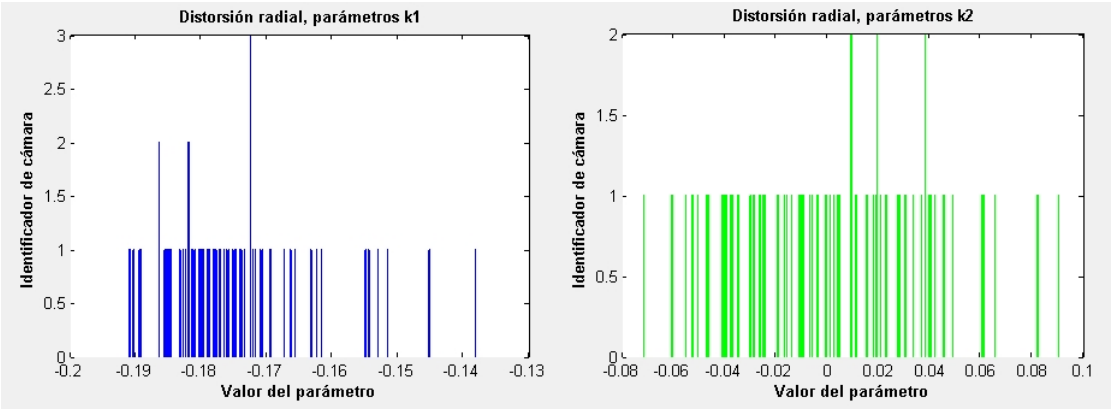

Figura 5.37: Distribución de la distorsión radial en MECO_3_MOD_1

Fuente: e.p.

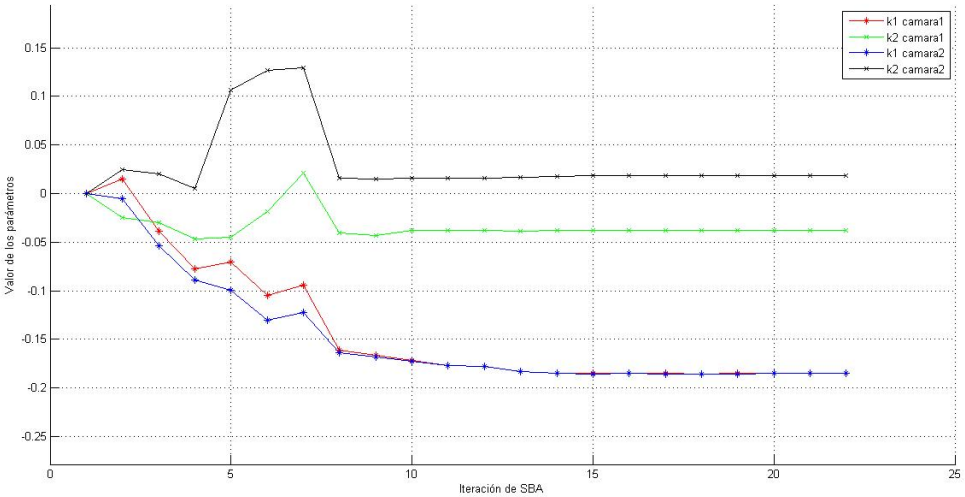

Figura 5.38: Evolución de la distorsión en MECO_3_MOD_1

Valores de los parámetros de distorsión radial calculados en cada iteración SBA. Fuente: e.p.

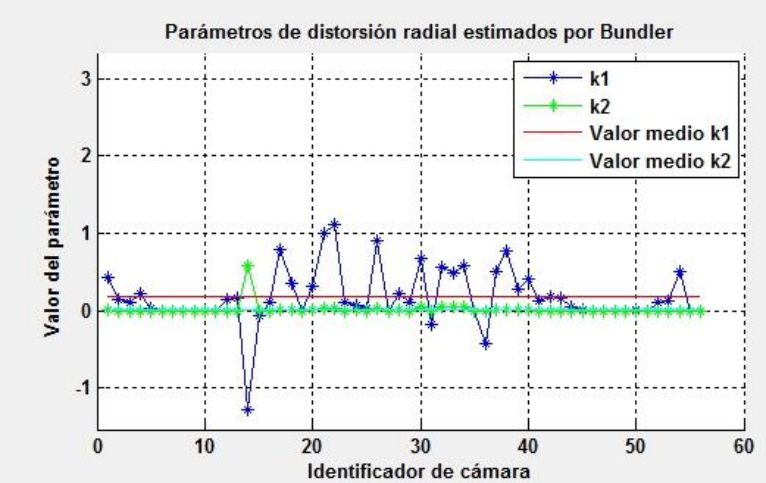

Figura 5.39: Distorsión radial en TORRE_PICASSO_AZCA

Fuente: e.p.

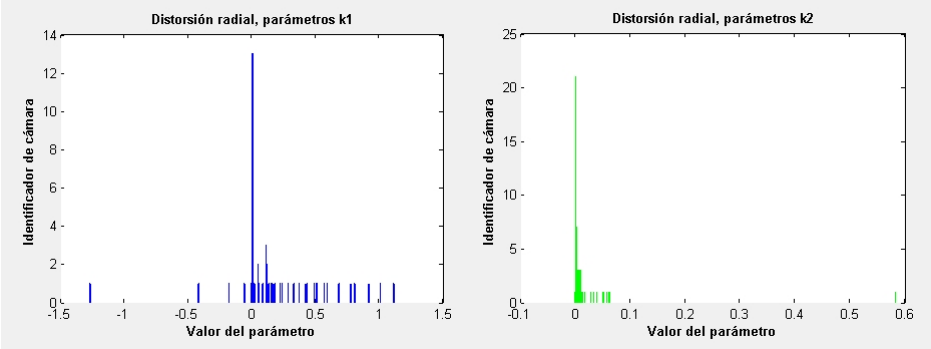

Figura 5.40: Distribución de la distorsión en TORRE_PICASSO_AZCA

Fuente: e.p.

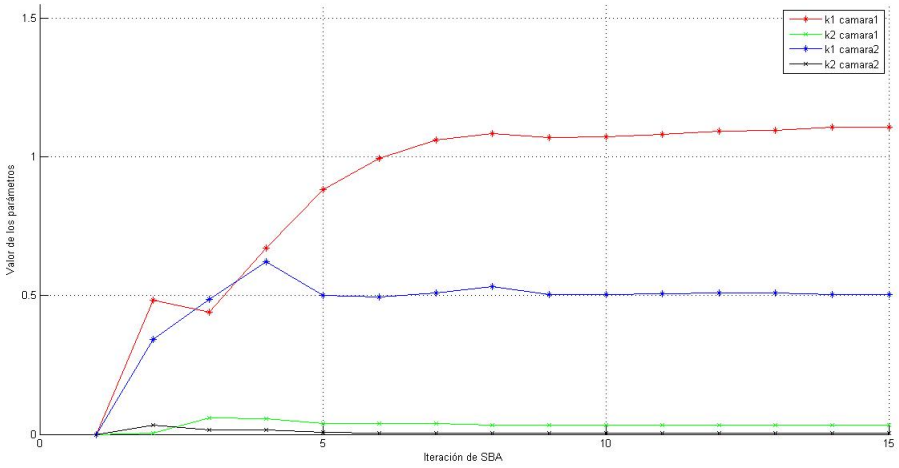

Figura 5.41: Evolución de la distorsión en TORRE_PICASSO_AZCA

Valores de los parámetros de distorsión radial calculados en cada iteración SBA. Fuente: e.p.

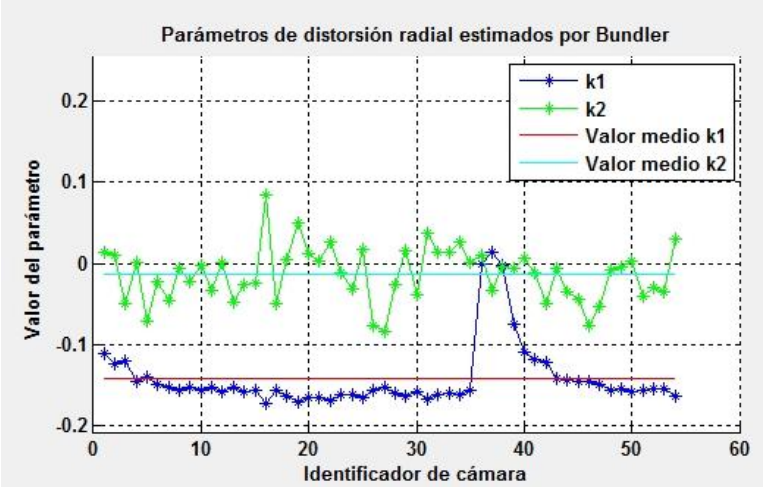

Figura 5.42: Distorsión radial en TORRE_PICASSO_AZCA_MOD_1

Fuente: e.p.

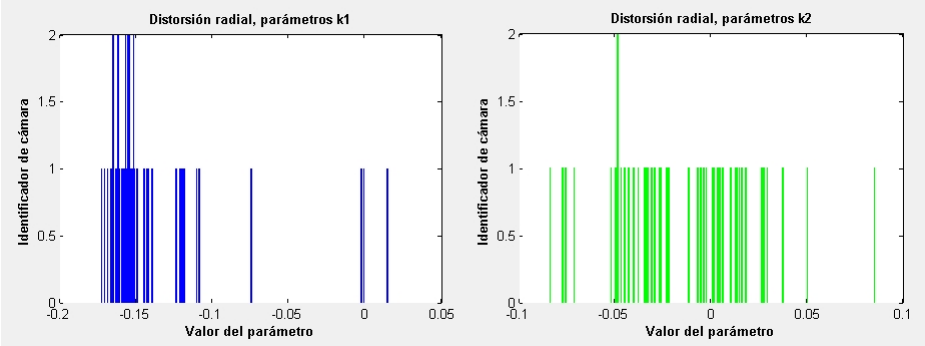

Figura 5.43: Distribución de la distorsión en TORRE_PICASSO_AZCA_MOD_1

Fuente: e.p.

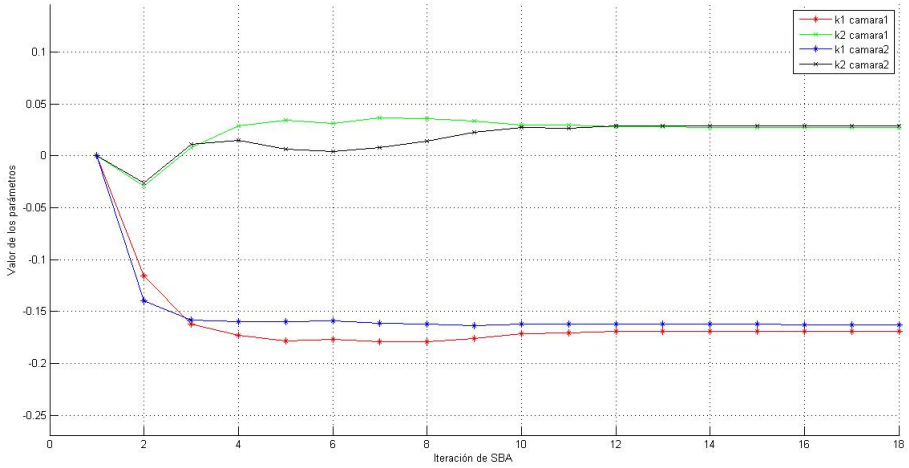

Figura 5.44: Evolución de la distorsión en TORRE_PICASSO_AZCA_MOD_1

Valores de los parámetros de distorsión radial calculados en cada iteración SBA. Fuente: e.p.

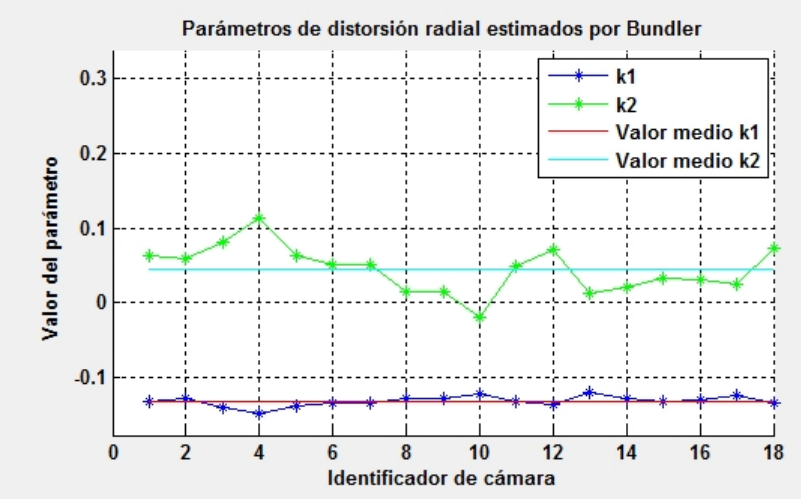

Figura 5.45: Distorsión radial en POLITECNICA con EXIF

Fuente: e.p.

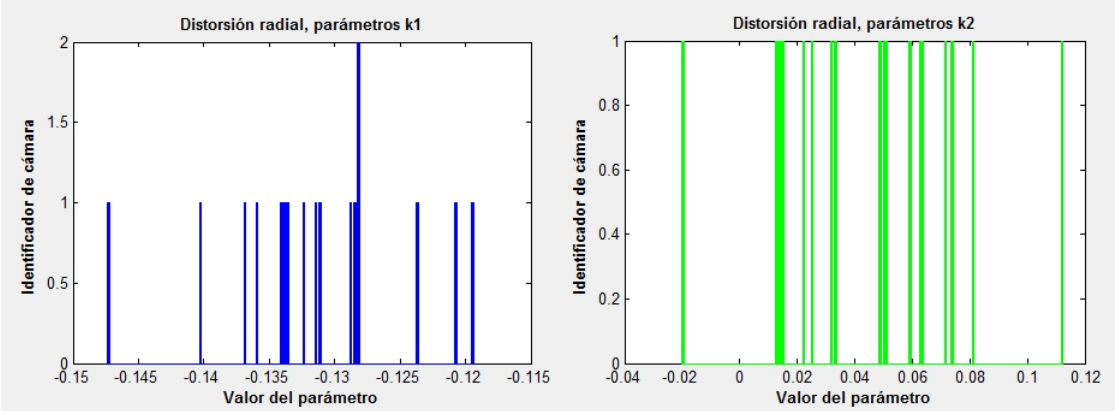

Figura 5.46: Distribución de la distorsión en POLITECNICA con EXIF

Fuente: e.p.

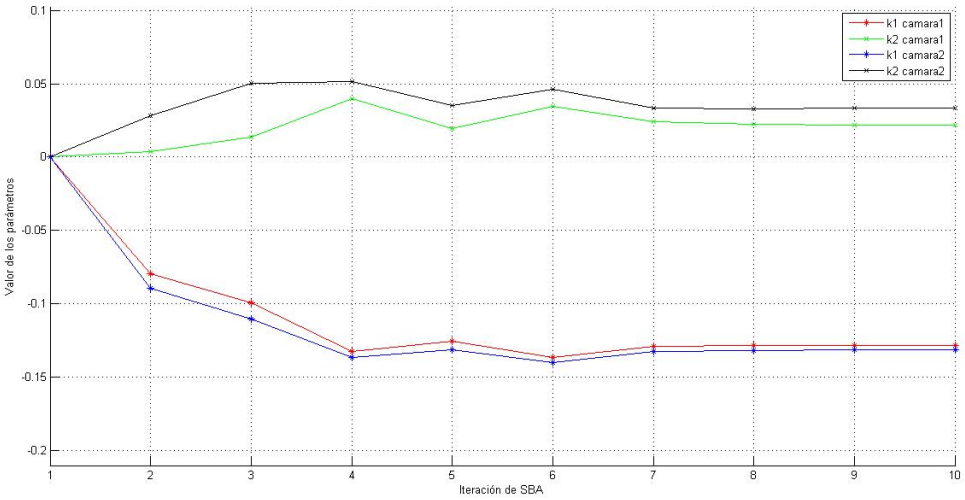

Figura 5.47: Evolución de la distorsión en POLITECNICA con EXIF

Fuente: e.p.

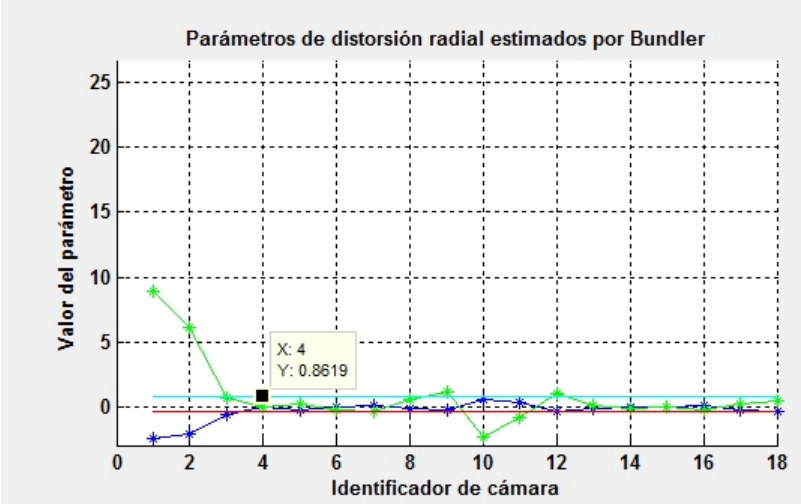

Figura 5.48: Distorsión radial en POLITECNICA sin EXIF

Fuente: e.p.

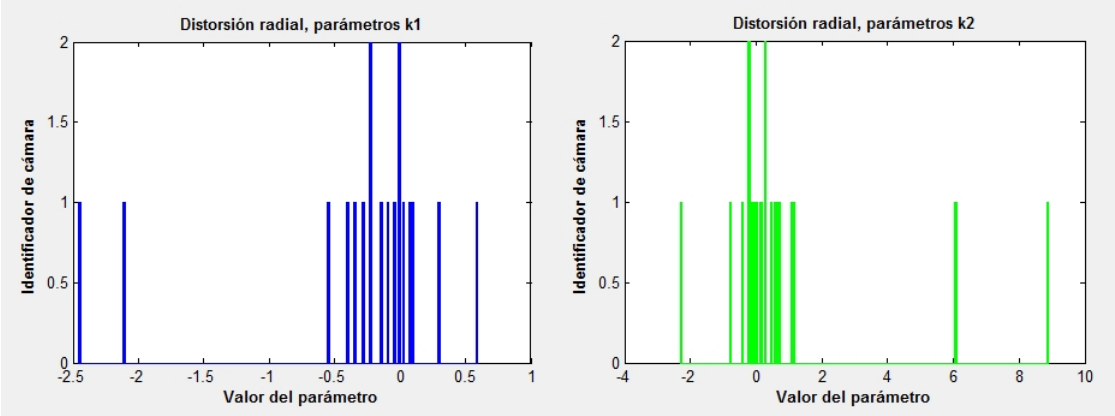

Figura 5.49: Distribución de la distorsión en POLITECNICA sin EXIF

Fuente: e.p.

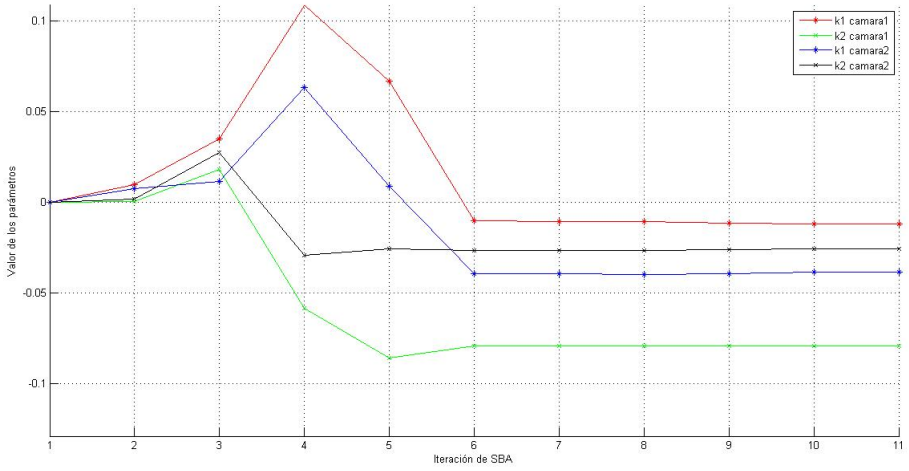

Figura 5.50: Evolución de la distorsión en POLITECNICA sin EXIF

Fuente: e.p.

5.3.6. Errores de SfM

Además de los problemas encontrados debido al tipo de imágenes empleadas y que se describen en la sección anterior, existen unos errores propios del método SfM de Bundler, observados ya por el autor del Software. Según se expone en [68], hay cuatro modos de fallo bien identificados, y que se han presentado también en las pruebas de reconstrucción realizadas a partir de las imágenes aéreas. Los modos de fallo son los siguientes:

1. Falta de solape: cuando las imágenes de entrada carecen de un solape adecuado entre sí. Que dos imágenes (o más) carezcan de solape significa que los objetos presentes en una de ellas no lo están en la otra, ya sea porque el ángulo con el que se toma la imagen es muy diferente, porque las cámaras tienen un desplazamiento grande entre sí, o porque ocurra el fenomeno conocido como oclusión de objetos (de la bibliografía *object occlusion*). Los tres casos tienen como consecuencia que al no disponer de objetos comunes, el algoritmo de "matching"no será capaz encontrar las suficientes correspondencias como para que el algoritmo de reconstrucción de estructura pueda obtener datos de su posición y del escenario. Según [68], para que haya suficiente solape entre imágenes es necesario que un punto sea visible por al menos tres cámaras.

2. Texturas ambiguas o repetitivas: cuando los objetos de la escena son muy similares, con texturas parecidas donde la etapa de matching no es capaz de diferenciar los puntos de un objeto de los del otro, y los trata como si perteneciesen a un único objeto. Este problema se presenta en gran medida en escenas urbanas con edificios de apariencia regular, o edificios cuya fachada se repita.

3. Mala inicialización: ocurre cuando los parámetros estimados para el par inicial son erróneos. Toda la reconstrucción, tanto puntos como cámaras, depende de los valores estimados para el par inicial. Si estos valores no se estiman de manera correcta y fiable, el resultado caerá en una solución errónea de la que probablemente no se pueda recuperar. Este problema se ha observado en todas las pruebas con fotogramas de vídeo sin cabecera EXIF, tanto con los escenarios del SIVA como con los de los documentales. Carecer de una focal en la cabecera EXIF hace que el sistema tenga que operar con estimaciones iniciales lejanas al valor real. Esto provoca que durante la optimización, el resultado puede caer en mínimos locales. La consecuencia final es que las focales recuperadas tengan valores muy dispares entre fotogramas o incluso valores negativos como se puede ver en los resultados presentados. Hay dos causas por las que la inicialización puede fallar:

   - Inverso de Necker (Necker reversal): se conoce como cubo de Necker a una ilusión óptica que hace que un cubo dibujado solo por sus aristas tenga dos posibles interpretaciones. En la reconstrucción de estructura también puede presentarse esta ambigüedad, ya que las dos interpretaciones posibles del primer par son estables. Estas dos interpretaciones vendrán determinadas por un mínimo global y un mínimo local de la función de coste de SfM. Si la pareja inicial de imágenes presenta el inverso de Necker, es muy probable que la solucion final también presente este problema. Para evitarlo la solución pasa por tomar un par inicial diferente. En la figura 5.51 se puede ver un ejemplo de este problema.
   - Paralaje insuficiente: cuando el par inicial tiene una línea base insuficiente para una reconstrucción precisa. Bundler estima el paralaje entre imágenes mediante una homografía obtenida a partir de las correspondencias. Para calcular la

validez de esta homografía se aplica un procedimiento RANSAC con votación entre puntos inliers y outliers a ésta. Si él número de outliers es alto, entonces las imágenes tienen poco paralaje y se pueden tomar como cámaras iniciales. Ahora bien si dos cámaras tienen una línea base pequeña pero además presentan una gran cantidad de correspondencias falsas que RANSAC toma como outliers, éste par puede ser aceptado erróneamente como un buen par inicial. Para resolver el problema conviene seleccionar manualmente un par de inicio con una línea base suficiente.

4. Errores en cascada: cuando alguno de los parámetros de cámara se estima erróneamente, el error se propagará en el resto del escenario. Los puntos añadidos dependerán de la cámara errónea dando lugar a triangulaciones erróneas. A su vez, la estimación de las nuevas cámaras depende de los puntos, que al ser erróneos darán lugar a cámaras erróneas y así sucesivamente. El resultado son estimaciones erróneas de grandes partes de la reconstruccion de las que será difícil recuperarse. Se identifican dos causas:

   - Mala inicialización de camara: cuando los puntos correspondientes de la imagen son pocos y un porcentaje alto de ellos son falsos, el algorimto RANSAC puede aceptar estimaciones falsas de la cámara. Una vez la cámara tenga estimaciones incorrectas, el error se propagará en el resto de la reconstrucción.
   - Incertidumbre elevada: cuando la posición de los puntos correspondientes en la imagen no permite obtener buenas inicializaciones de las cámaras. Por ejemplo, una imagen cuyos puntos correspondientes estén la mayoría próximos en una pequeña región de la imagen. Esta proximidad puede dar lugar a estimaciones poco precisas, especialmente cuando las distancias del escenario son grandes. Relativa a la distancia del edifico el error puede ser pequeño, pero relativa a objetos mucho más cercanos, la incertidumbre puede ser elevada. si la vista se usa para triangular puntos distantes 3 metros, los puntos pueden tener un error significativo aun teniendo un error de repreoyección bajo.

Estos problemas son fácilmente identificables visualmente. Una de las consecuencias de estos problemas es que el escenario se parta en dos piezas totalmente irreconciliables, una construida antes del error y otra construida después del error. Según se explica en [68], el orden en que se reconstruye el escenario, determinado por el par inicial elegido, influye absolutamente en la salida (cada nueva cámara se añade en función de ese par).

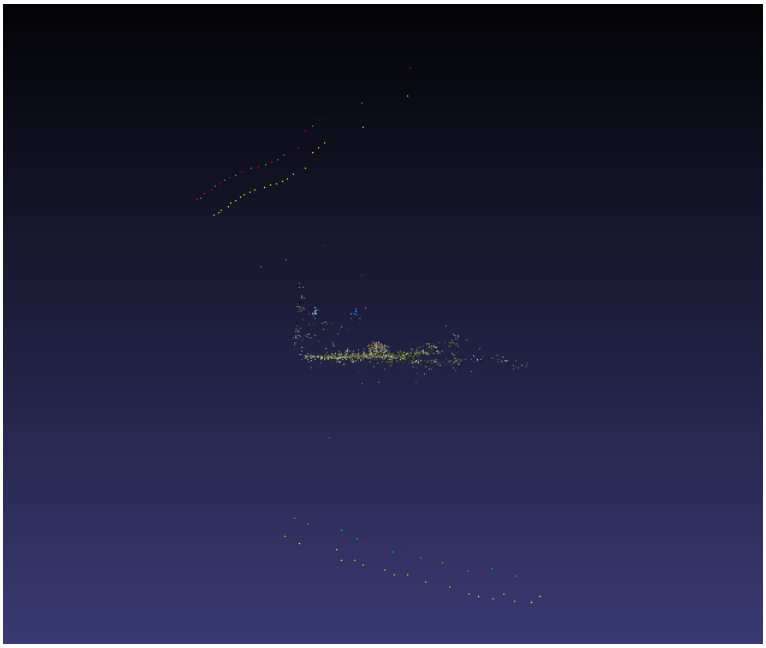

Figura 5.51: Reverso de Necker en el escenario ERMITA_0
El escenario corresponden a un vuelo alrededor de una ermita. Se puede apreciar cómo las posiciones estimadas de las cámara se agrupan en dos subconjuntos, uno sobrevolando el edificio y otro por debajo (obviamente esta posición es incoherente). Este problema se debe al reverso de Necker. Fuente: e.p.

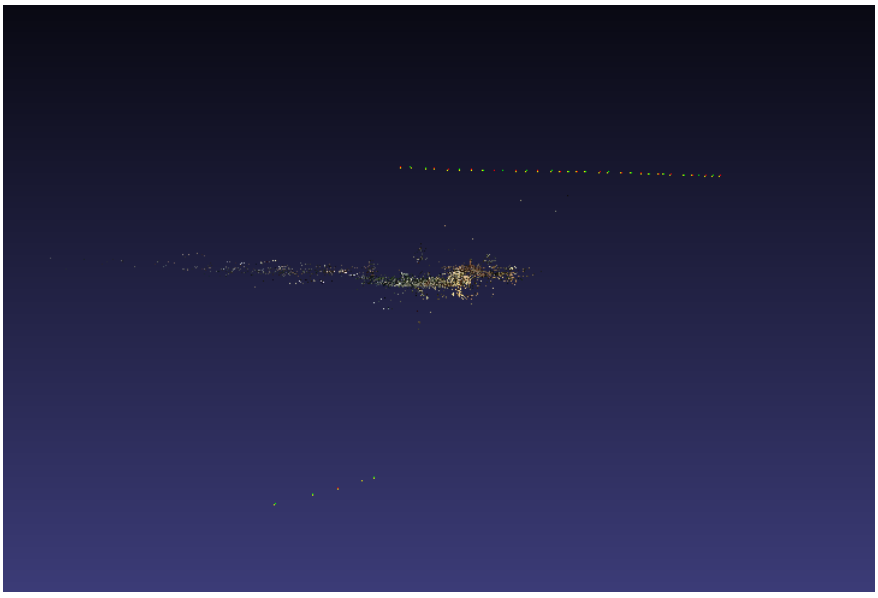

Figura 5.52: Reverso de Necker en el escenario ALCALA_2
El escenario corresponden a un vuelo sobre la Plaza Cervantes en Alcalá de Henares. El resultado se ve afectado por el reverso de Necker. Las cámaras están agrupadas en dos subconjuntos, y la nube de puntos está entre medias. Las cámaras de la parte inferior son claramente incoherentes. En primer lugar, el vuelo es una trayectoria aproximadamente recta y sus las posiciones de estas cámaras debieran estar próximas a las cámaras de la parte superior. En segundo lugar, si la posición fuese la parte inferior, significaría que la imagen se ha tomado de abajo hacia arriba, lo cual es imposible. Fuente: e.p.

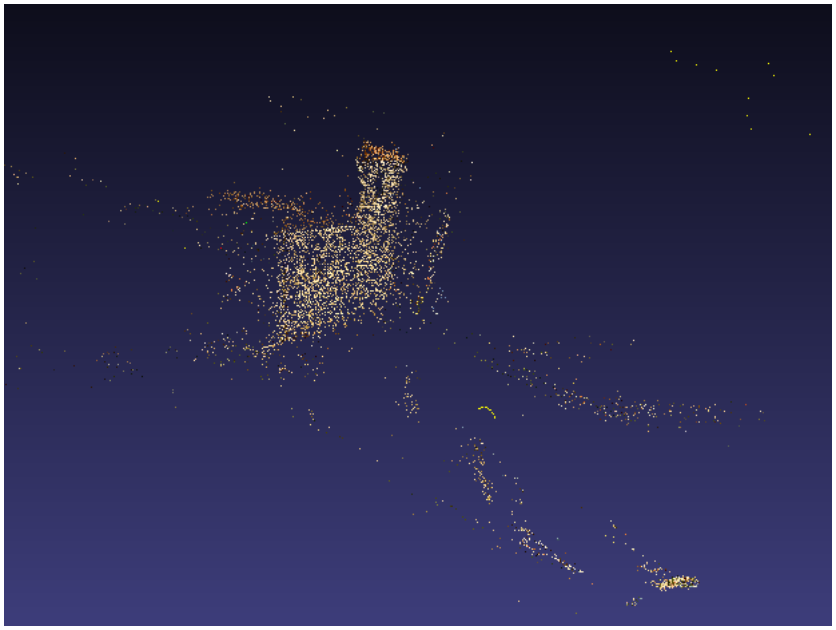

Figura 5.53: Errores en cascada en el escenario LOS_SANTOS_2

El resultado de la reconstrucción aparece partido. Este es un ejemplo del problema causado por una mala inicialización. El resultado final es un error en cascada que acaba partiendo los objetos. Fuente: e.p.

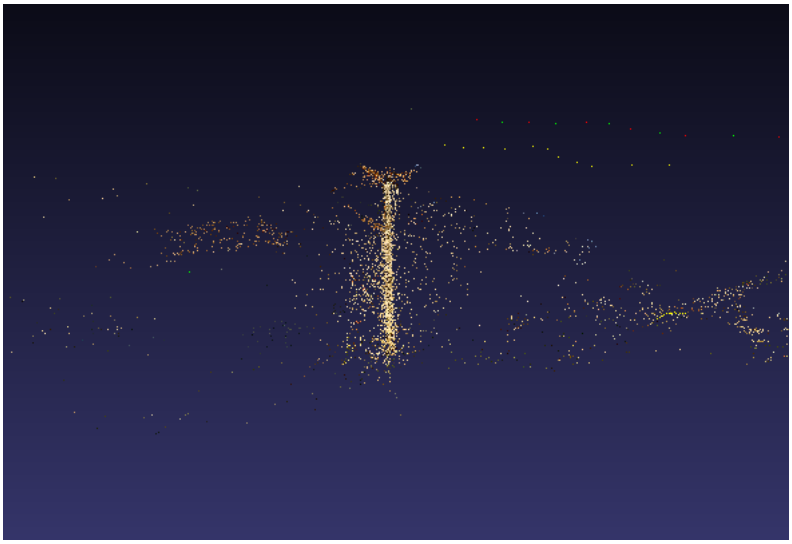

Figura 5.54: Errores en cascada en el escenario LOS_SANTOS_2

En otro detalle del mismo escenario se puede apreciar una vista lateral de la fachada de la iglesia. La reconstrucción del tejado se ha partido en dos partes reflejadas, como puede verse en la parte superior. Fuente: e.p.

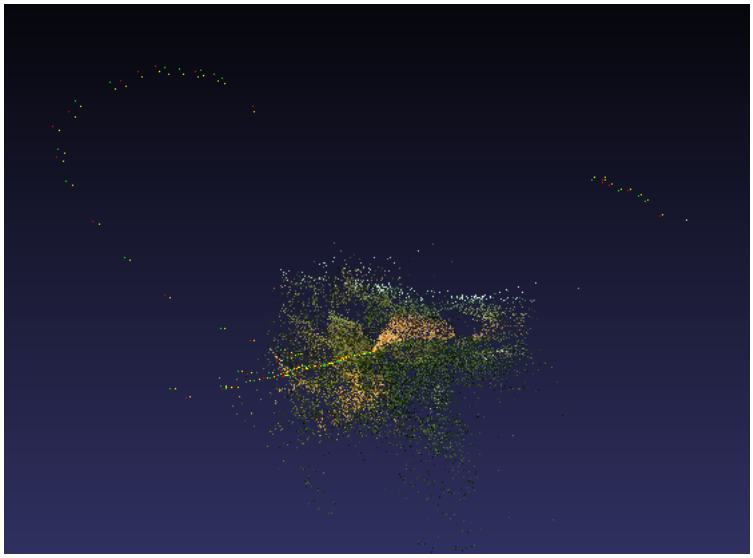

Figura 5.55: Errores en cascada debidos a una mala inicialización
Reconstrucción de los cerros en los márgenes del río Henares a su paso por Alcalá de Henares. Debido a que se desconoce la focal de los fotogramas, se produce una mala inicialización que se arrastra a lo largo de toda la reconstrucción. Fuente: e.p.

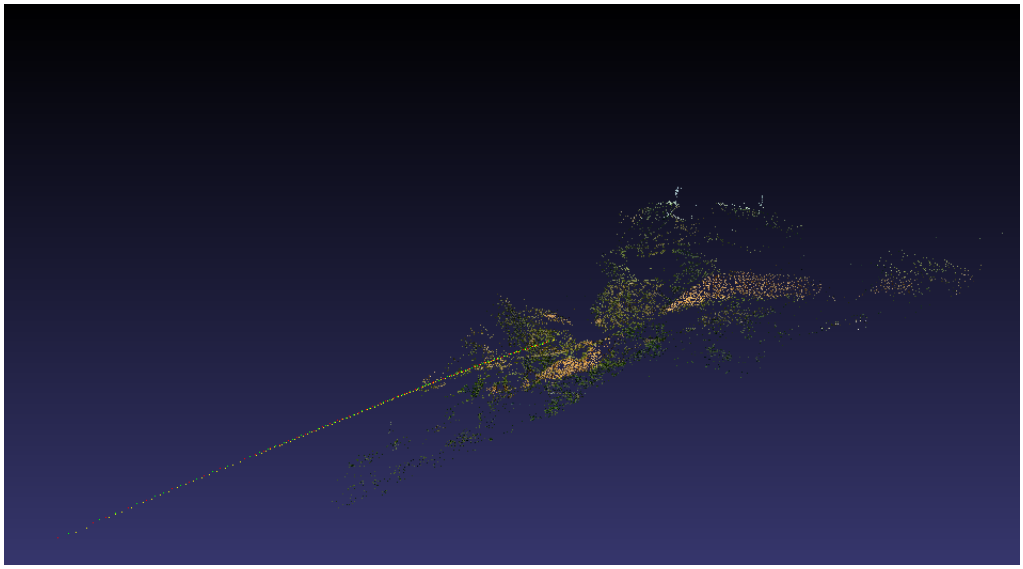

Figura 5.56: Errores en cascada debidos a una mala inicialización
El mismo escenario de la figura anterior pero a diferencia, en este caso se usa una estimación de la focal para inicializar las cámaras. Como se puede pareciar el resultado es mucho mejor. Fuente: e.p.

5.4. Escenarios de ejemplo

En el DVD adjunto a esta memoria (ver E para más información) se incluyen varios escenarios, de los cuales, en esta sección se comentan un escenario del SIVA, otro de las fotografías y dos de los documentales. Éstos dos últimos corresponden a los más representativos del conjunto de escenarios de los documentales, y además, se han georreferenciado con Orto3D. Los nombres de los escenarios son ALCONADA_2_MOD_2, POLITECNICA, TORRE_PICASSO_MOD_1 y MECO_3_MOD_1. En los apartados a continuación se proporciona más información sobre cada escenario concreto.

5.4.1. ALCONADA_2_MOD_2

Este escenario pertenece a un vuelo realizado con un UAV real, el SIVA, en los alrededores de Corral de Ayllón (Segovia). En concreto la secuencia de fotogramas es una grabación de Alconada de Maderuelo, un pueblo en las cercanías de la zona de vuelo. En los fotogramas se muestra la iglesia y algunas casas del pueblo. La distancia entre el vehículo y el pueblo es grande, superior al kilómetro. Los fotogramas son de baja resolución pixélica, 352×288 píxeles, han sido comprimidos para la transmisión vía enlace radio y presentan caracteres sobreimpresos. En las figuras 5.58 y 5.59 se presentan dos de los fotogramas del escenario. Como la resolución es pequeña, SIFT no es capaz de recuperar una gran cantidad de puntos, siendo detectados algunos de ellos en los caracteres. Al tener pocos puntos característicos, se tendrán también pocas correspondencias y la reconstrucción dispondrá de poca información en la que apoyarse.

En la figura 5.57 se presenta el resultado de la reconstrucción y las cámaras. Como

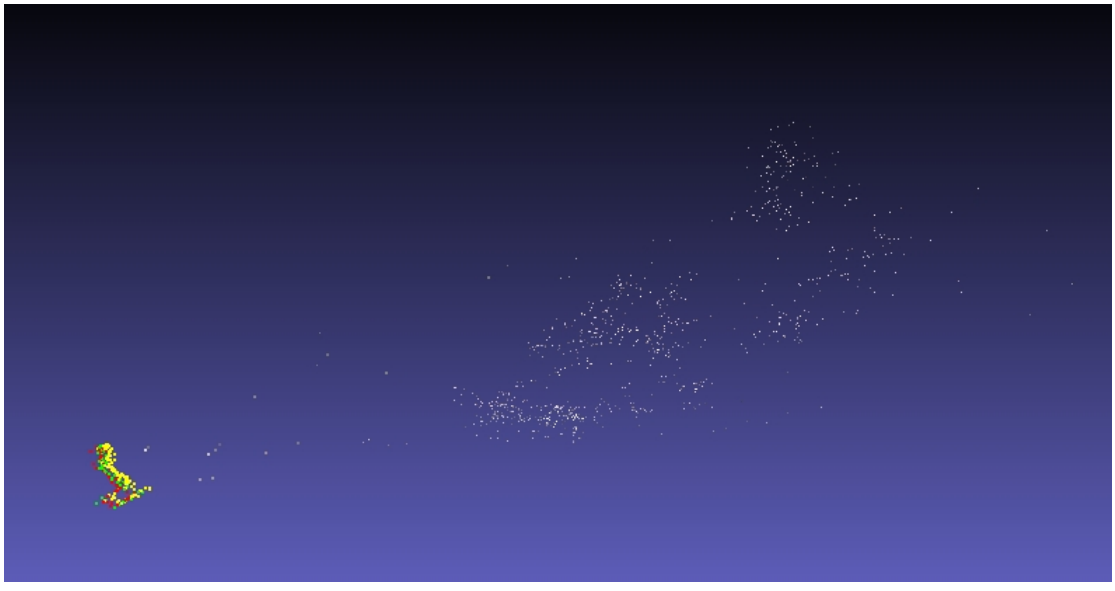

Figura 5.57: Reconstrucción de ALCONADA_2_MOD_2

Resultado devuelto por Bundler para este escenario. Fuente: e.p.

se puede ver, es imposible reconocer algún edificio en la nube de puntos. Es más, las posiciones de las cámaras en la esquina inferior izquierda, no parecen muy coherentes con el movimiento del UAV. Al tratarse de un escenario tomado de fotogramas de vídeo, no se dispone de cabecera EXIF. Para usar un valor de inicialización de focal, se ha tomado

$f = 1{,}2res_x = 422{,}4$ con $res_x = 352$. En las figuras 5.60 y 5.61 se presentan los resultados de estimación de la focal. En la primera figura se puede apreciar que Bundler recupera valores de focales en un rango no muy grande en comparación con los resultados del mismo escenario sin el forzado 5.3.1. A diferencia de los resultados anteriores, el caso aquí presentado usa también el filtro de correspondencias.

Figura 5.58: Fotograma SIVA (1)

Ejemplo de fotograma tomado de un vídeo del SIVA. Fuente: INTA

Figura 5.59: Fotograma SIVA (2)

Ejemplo de fotograma tomado de un vídeo del SIVA. Fuente: INTA

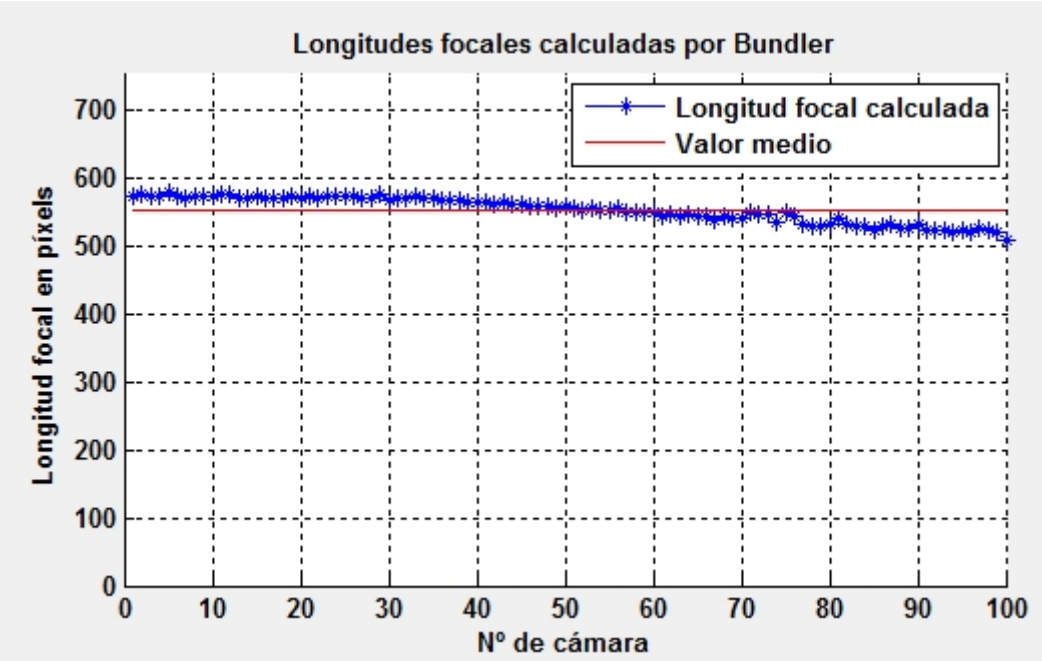

Figura 5.60: Focales en ALCONADA_2_MOD_2

Focales estimadas cuando se inicia a $f = 422{,}4$ y usando el filtro. Fuente: e.p.

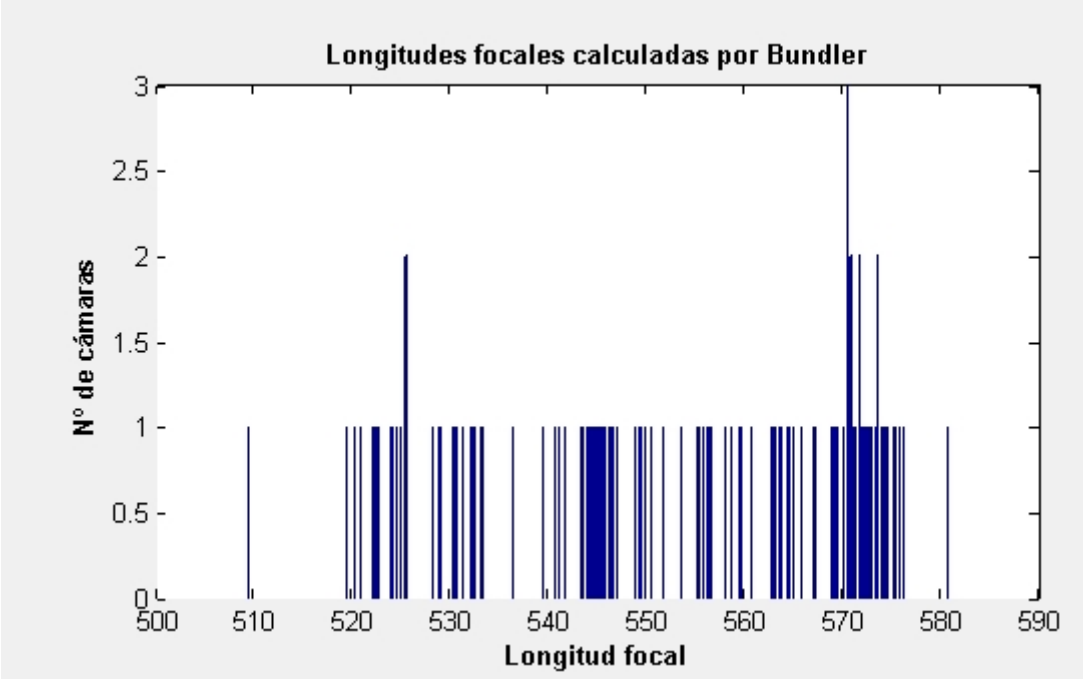

Figura 5.61: Distribución de focales en ALCONADA_2_MOD_2

Histograma de las focales estimadas cuando se inicia a $f = 422{,}4$ y usando el filtro. Fuente: e.p.

5.4.2. POLITECNICA

Este escenario ha sido reconstruido a partir de fotografías tomadas con una cámara convencional[9]. El edificio mostrado es la Escuela Politécnica de la Universidad de Alcalá, en concreto la esquina Oeste del edificio (laboratorios de pesados). Esta parte del edificio tiene una planta cuadrada, lo que permite identificar fácilmente la planta en una nube de reconstrucción. Además, como se conocen las posiciones de las cámaras de una forma aproximada, se puede comprobar que las estimaciones de localización de las cámaras devueltas por Bundler son coherentes. Con todo esto, se dispone de una fuente de información adecuada para el desarrollo del prototipo. En inicio se utilizaron los vídeos del SIVA para implementar las ventanas y funcionalidad de Orto3D, sin embargo la mala calidad de la nube y de la reconstrucción hacía difícil la validación de los resultados obtenidos. Este escenario de fotografías, al ser conocido, tener buena resolución de imagen y usar las focales de las cámaras, ha simplificado el desarrollo de Orto3D sin dar lugar a ambigüedades en los resultados.

El escenario trabaja con ocho cámaras de las cuales Bundler ha sido capaz de reconstruir seis. En las figuras 5.62 y 5.63 se presentan dos de las imágenes usadas en el escenario. En la figura 5.64 se puede apreciar la facilidad a la hora de georreferenciar. La planta del edificio reconstruido por Bundler y la de la ortoimagen son perfectamente identificables.

El escenario reconstruido se puede ver en las imágenes tomadas de Meshlab en las figuras 5.65, 5.66 y 5.67. Se muestran tres imágenes en las que se pueden ver dos vistas laterales y una vista de la planta de la reconstrucción.

Figura 5.62: Fotografía de POLITECNICA (1)

Fuente: e.p.

[9]Canon Powershot A520

Figura 5.63: Fotografía de POLITECNICA (2)

Fuente: e.p.

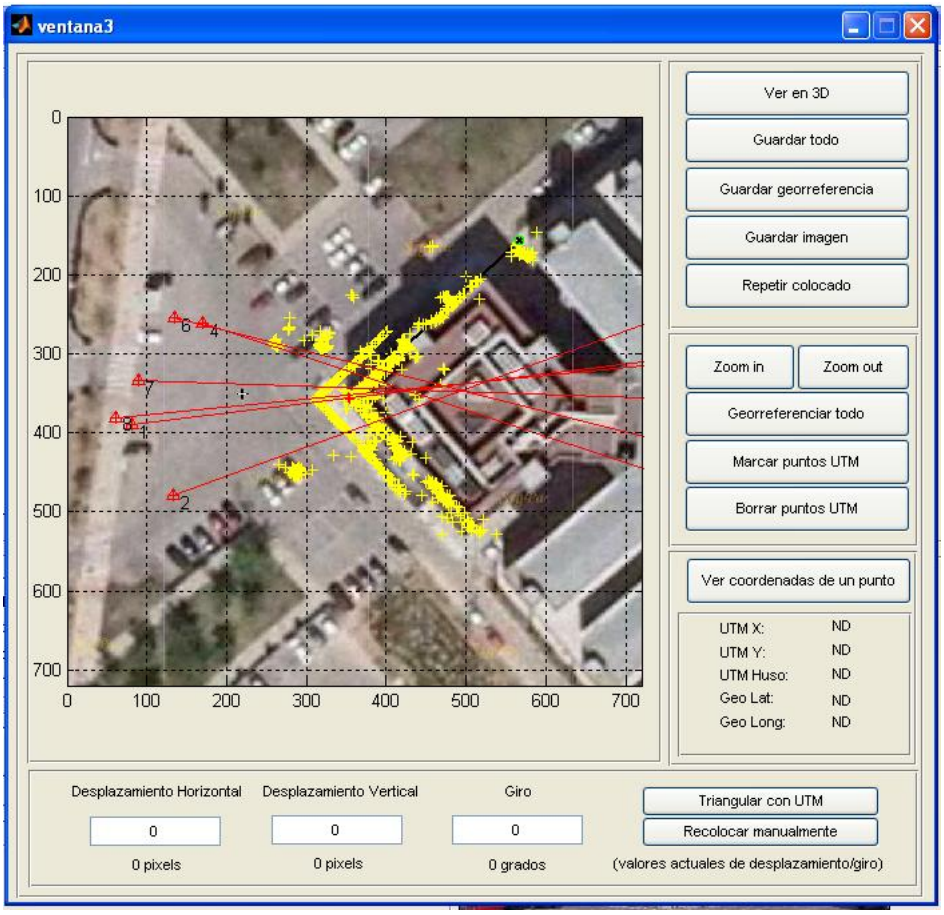

Figura 5.64: Detalle de la georreferenciación del escenario POLITECNICA

En la imagen se puede ver la planta georreferenciada con una ortoimagen tomada de Sigpac. Se puede ver como la planta de la reconstrucción sigue la planta del edificio (puntos amarillos). Fuente: e.p.

Figura 5.65: Reconstrucción de POLITECNICA (1)
Nube de puntos de la reconstrucción. Fuente: e.p.

Figura 5.66: Reconstrucción de POLITECNICA (2)
Vista de la nube de puntos desde otro ángulo. Fuente: e.p.

Figura 5.67: Reconstrucción de POLITECNICA (3)
Vista en planta de la nube de puntos. Se puede reconocer fácilmente la planta del edificio. Fuente: e.p.

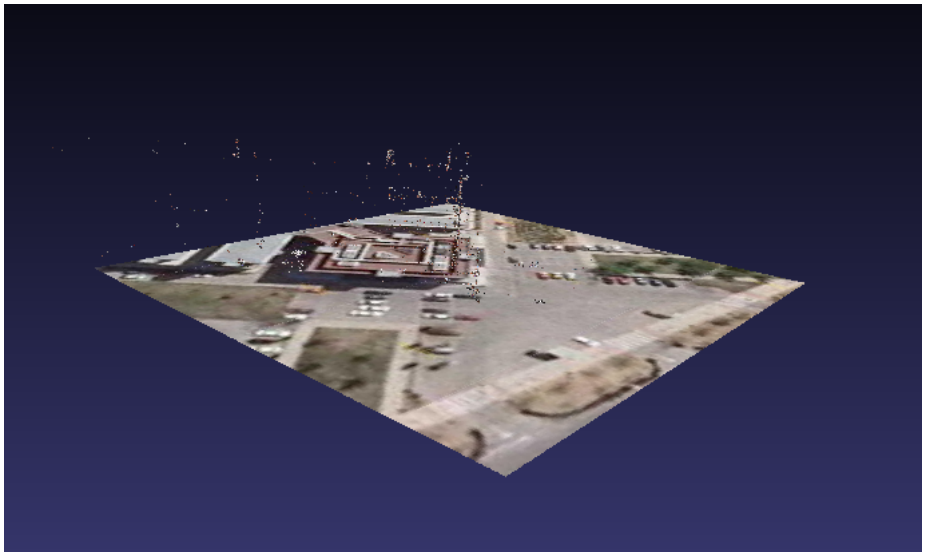

Figura 5.68: Reconstrucción de POLITECNICA sobre ortoimagen
Reconstrucción 3D de Politécnica. Fuente: e.p.

5.4.3. TORRE_PICASSO_AZCA_MOD_1

Este escenario pertenece al conjunto de escenarios tomados de los documentales de televisión. Se trata de una grabación alrededor de la zona comercial Azca en Madrid, con la Torre Picasso como edificio más notable de la imagen. En las figuras 5.69 y 5.70 se muestras dos fotogramas del escenario.

En las siguientes figuras 5.71, 5.72 y 5.73 se presentan capturas de pantalla de la re-

Figura 5.69: Fotograma de TORRE_PICASSO_AZCA_MOD_1 (1)
Fuente: documentales *Madrid desde el aire*.

Figura 5.70: Fotograma de TORRE_PICASSO_AZCA_MOD_1 (2)
Fuente: documentales *Madrid desde el aire*.

construcción, mostrando únicamente la nube de puntos y las cámaras. Por último, en la figura 5.74 se puede ver la misma nube de puntos georreferenciada y superpuesta a la ortoimagen usada para la georreferenciación y en la figura 5.75 esta misma visualización pero lateral, con la ortoimagen como base del escenario.

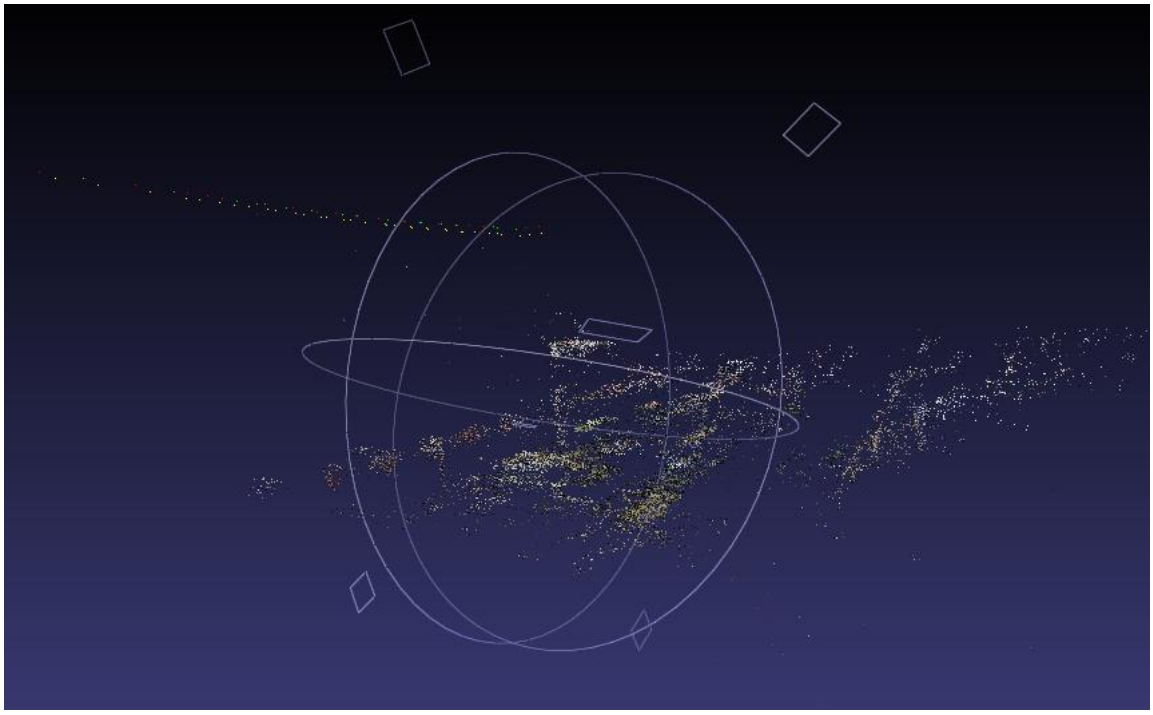

Figura 5.71: Reconstrucción de TORRE_PICASSO_AZCA_MOD_1 (1)
En la imagen se pueden apreciar los edificios y la trayectoria de las cámaras. Este escenario no es lo suficientemente denso como para visualizar los edificios en una vista rápida, pero se pueden reconocer algunas formas con ayuda de los fotogramas. Fuente: e.p.

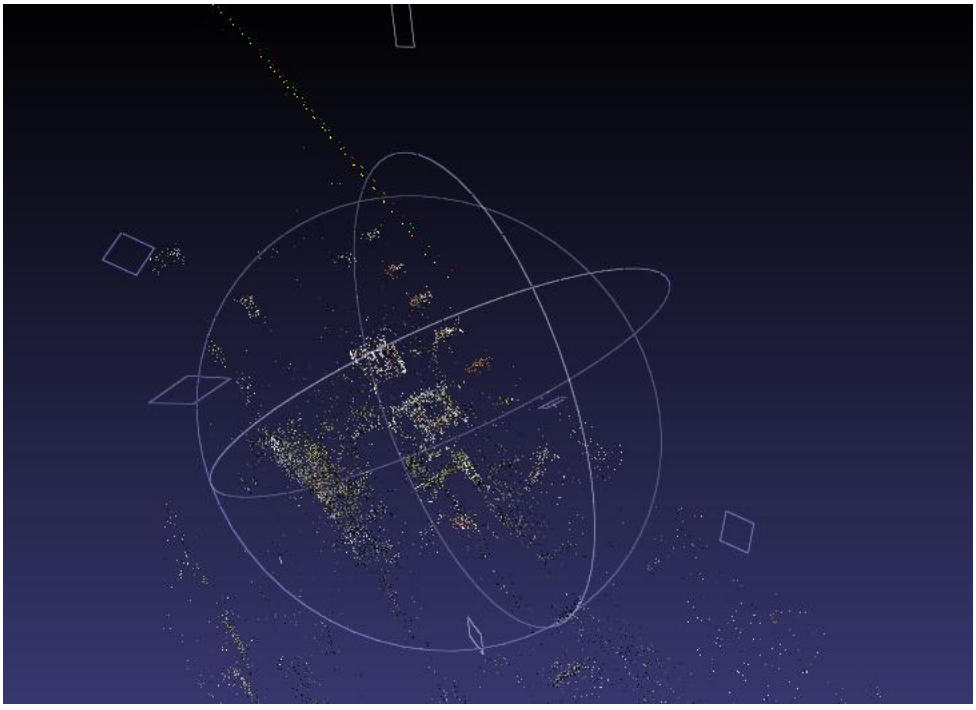

Figura 5.72: Reconstrucción de TORRE_PICASSO_AZCA_MOD_1 (2)
Otra vista del mismo escenario en la que se puede apreciar la trayectoria de las cámaras y la nube de puntos de los edificios. Fuente: e.p.

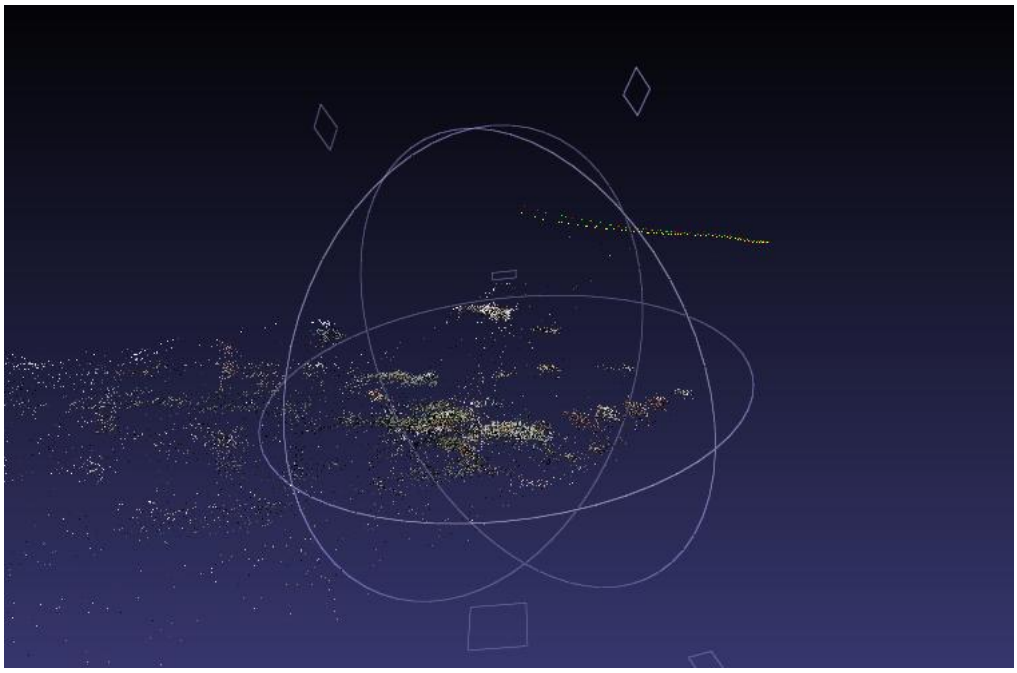

Figura 5.73: Reconstrucción de TORRE_PICASSO_AZCA_MOD_1 (3)
En la parte superior derecha se pueden apreciar las cámaras reconstruidas. Al ser de una misma secuencia de vídeo, la posición de las cámaras permite conocer aproximadamente la trayectoria del vehículo aéreo. Fuente: e.p.

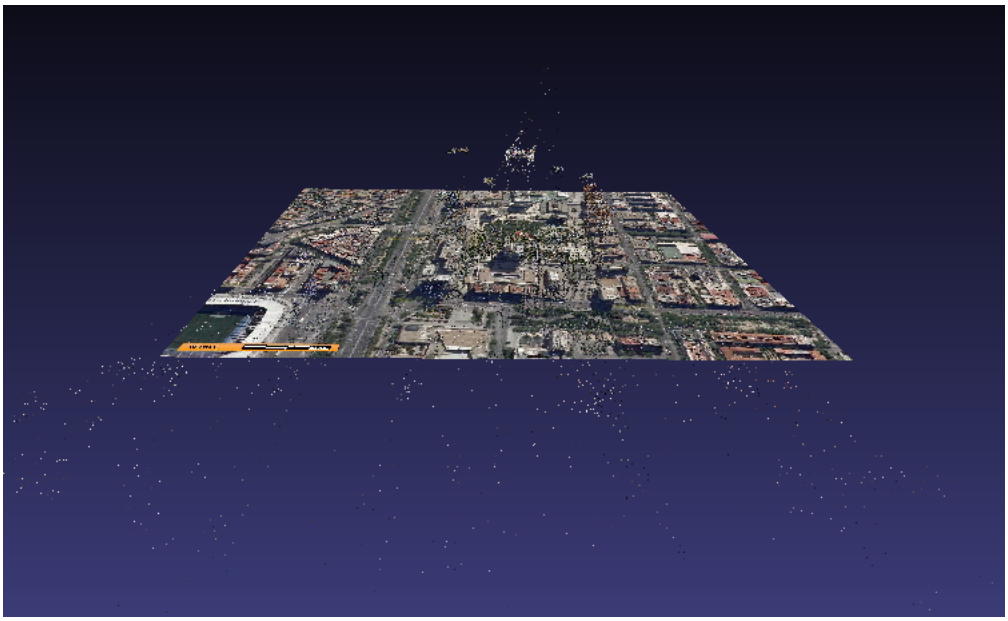

Figura 5.74: Escenario TORRE_PICASSO_AZCA_MOD_1 sobre ortoimagen (1)
Después de georreferenciar el escenario, la nube de puntos se representa sobre la ortoimagen. En esta figura se puede apreciar la Torre Picasso en la parte central. Fuente: e.p.

Figura 5.75: Escenario TORRE_PICASSO_AZCA_MOD_1 sobre ortoimagen (2)

Vista lateral de la reconstrucción georreferenciada. En la parte central de la imagen se puede apreciar la reconstrucción de la Torre Picasso y los edificios que le rodean. En la parte inferior de la nube de puntos se encuentra la ortoimagen que al ser una vista lateral del escenario aparece como una línea. Fuente: e.p.

5.4.4. MECO_MOD_1

El escenario de MECO corresponde también a una grabación de los documentales en este caso a un conjunto de fotogramas grabados alrededor del campanario de la Iglesia de Nuestra Señora de la Asunción en Meco (Madrid). A diferencia de los anteriores mostrados, es el que mejor densidad de puntos tiene. El vehículo realizó la grabación a una distancia muy próxima al edificio, de ahí que se disponga de una gran cantidad de puntos correspondientes para modelar la reconstrucción. En las figuras 5.76 y 5.77 se muestran dos de los fotogramas. En las figuras 5.78, 5.79 y 5.80 se presenta la reconstrucción desde diferentes ángulos, donde es fácil reconocer las posiciones de las cámaras. Por último, en la figura 5.81 se presenta la reconstrucción georreferenciada sobre la ortoimagen. Como se puede apreciar, se ha tomado un punto en el campanario como punto de referencia de altura. Lo correcto sería marcar un punto en el suelo y que la posición de éste en el eje Z de la reconstrucción se tome como origen de alturas del escenario, sin embargo, el tipo de fotogramas usados no ha permitido el marcado fiable de alguno. Para evitar ambigüedades, se ha tomado un punto de la parte inferior del campanario bien visualizado en la secuencia de fotogramas.

Figura 5.76: Fotograma de MECO_1_MOD_1 (1)

Fuente: documentales *Madrid desde el aire*.

Figura 5.77: Fotograma de MECO_1_MOD_1 (2)
Fuente: documentales *Madrid desde el aire*.

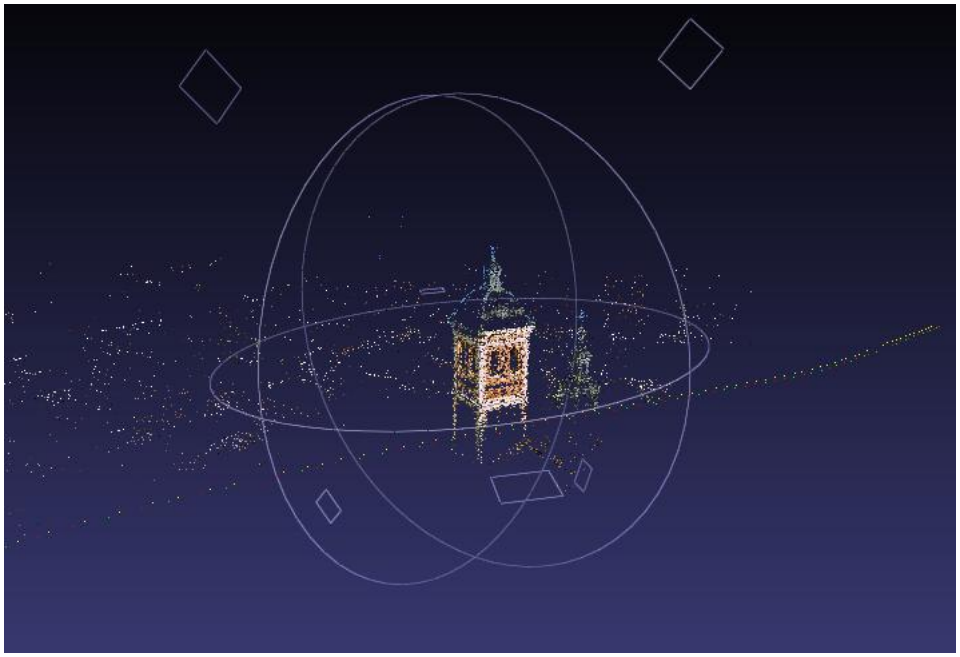

Figura 5.78: Reconstrucción de MECO_1_MOD_1 (1)
En la imagen se puede distinguir claramente la figura del campanario. Se han reconstruido algunos puntos del fondo, pero es imposible identificar los edificios que rodean el campanario. Fuente: e.p.

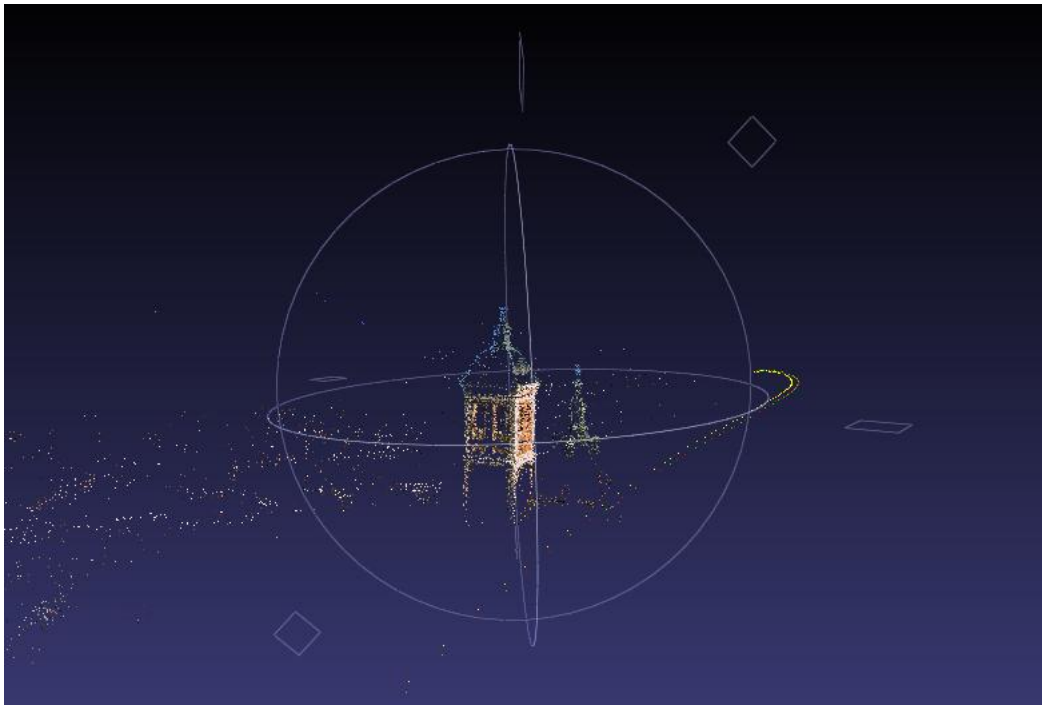

Figura 5.79: Reconstrucción de MECO_1_MOD_1 (2)
A la derecha se puede apreciar la trayectoria de las cámaras. Fuente: e.p.

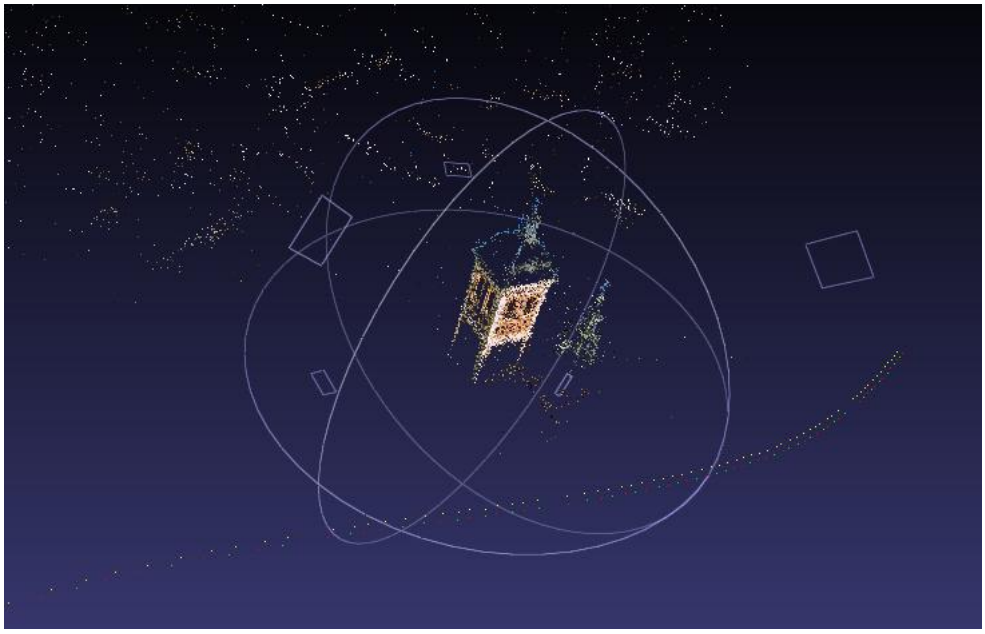

Figura 5.80: Reconstrucción de MECO_1_MOD_1 (3)
En la parte inferior se aprecia la trayectoria de las cámaras. El campanario es fácilmente reconocible. Fuente: e.p.

Figura 5.81: Reconstrucción de MECO_1_MOD_1 sobre ortoimagen

Como se puede apreciar por la posición de la ortoimagen respecto a la nube, se ha tomado un punto del campanario como altura de referencia. Fuente: e.p.

5.5. Proceso recomendado

En primer lugar, para poder obtener reconstrucciones con Bundler es indispensable trabajar con cámaras cuyas longitudes focales sean conocidas o estimaciones cercanas a los valores reales. En caso de que no se conozca el valor de la focal en píxeles ni se conozcan los parámetros de las cámaras, una de las opciones es inicializar el valor de la focal a 1,2 el ancho de la imagen. Éste es un valor empírico basado en los valores de longitud focal que tienen la mayoría de las cámaras digitales, aunque puede no estar próximo al valor real. Como ésta tampoco es una estimación válida, otra opción usada en este trabajo es ejecutar Bundler sin un valor de focal, comprobar los datos devueltos, y volver a ejecutar forzando las focales con un nuevo valor elegido entre las focales estimadas durante la primera ejecución. Este segundo método puede dar también buenos resultados, aunque es preferible usar el primer método. Las imágenes usadas deben tener una adecuada separación entre ellas. En el caso de secuencias de vídeo se debe usar un muestreo de fotogramas que permita tener una línea base suficiente. Por ejemplo, se pueden muestrear a 1:10 (tomar un fotograma de cada diez), 1:20, 1:25 o superiores. La frecuencia de muestreo dependerá de la velocidad del vehículo. Obviamente a menos velocidad, la separación métrica entre fotogramas será menor. Otro punto a tener en cuenta es el número de imágenes empleadas. A mayor número, mayor tiempo de ejecución y más probabilidad de que la reconstrucción caiga en algunos de los problemas comentados en el apartado 5.3.6.

5.6. Tareas realizadas

El trabajo se puede dividir en varias fases, a su vez divididas en varias tareas. Algunas de ellas se han llevado a cabo en paralelo. La primera fase ha consistido en un estudio de la teoría detras de la reconstrucción de estructura, la visión computacional y el tratamiento de imágenes. Esta tarea se ha continuado hasta el final del trabajo, debido a la gran cantidad de algoritmos e información existente sobre el campo. A la vez se inició la evaluación de herramientas de código abierto para elegir aquella más robusta. Finalmente la actividad se centró en usar Bundler por las propiedades descritas sobre esta herramienta en la documentación. La siguiente tarea vino marcada por el uso inicial de Bundler con vídeos tomados de UAV, fase que puede resumirse como pruebas iniciales de Bundler. La siguiente fase, viene determinada por la validación de los resultados y ante la ausencia de una herramienta que permitiese la comprobacion de manera correcta y flexible, se desarrolló una propia. Las herramientas de visualización disponibles no permiten validar los resultados de una forma cómoda, sencilla y fiable, de ahí que haya sido necesario desarrollar una aplicación para realizar esta tarea. Los escenarios se probaron a validar con una versión inicial del prototipo de visualización y validación. A continuación se realizan sucesivas modificaciones sobre el prototipo inicial. Con el prototipo final y la aplicación se han validado los resultados dados por Bundler. La última fase consiste en la redacción de la memoria y otros documentos. Esta memoria incluye un manual de usuario de la aplicación además de los resultados de las pruebas.

El plan de trabajo se ha ido ajustando a medida que las necesidades, problemas o conclusiones así lo han requerido. El presente trabajo partió en un inicio con un objetivo claro, la evaluacion de herramientas de reconstrucción de estructura aplicadas a vídeos tomados desde UAV, aunque sin una definición clara del procedimiento de trabajo a seguir. Se identificaron una serie de fases de trabajo, pero debido a los continuos problemas encontrados al trabajar con los vídeos y los escenarios reconstruidos, fue necesario añadir nuevas

tareas o modificar las identificadas inicialmente. Una de las modificaciones realizadas es la tarea de selección de una herramienta de reconstrucción. Dado el tiempo requerido para evaluar los resultados, la tarea se limitó a analizar únicamente una, elegida en base a los algoritmos usados por la aplicación, los resultados devueltos, la documentación disponible (aun siendo escasa) y el lenguaje de programación empleado, además de la disponilbidad del código fuente. Otra modificación de la actividad fue la implementacion del filtro de falsos puntos detectados para eliminar los efectos debido a los caracteres sobreimpresos en la imagen. En el siguiente apartado se describe con mayor detalle cada una de las fases.

5.6.1. Fases de trabajo

Cada una de las fases de trabajo se compone de varias tareas que se comentan en cada apartado.

5.6.1.1. Estudio de temas de visión computacional

Antes de comenzar a probar los vídeos y buscar diferentes aplicaciones de reconstrucción de estructura se procedió a estudiar la teoría básica de la visión computacional. Una vez conocidos los conceptos básicos se inicio la aplicación a los vídeos y el desarrollo de los prototipos. A lo largo de todo el trabajo se continuó el estudio de temas y algoritmos de visión computacional, en base al libro de referencia principal [25], apoyado por el tutorial de Pollefeys [54] y completado con la lectura de varios artículos que pueden verse en la bibliografía al final de la memoria.

5.6.1.2. Análisis de herramientas disponibles

En el momento de iniciar el trabajo se desconocían las aplicaciones de código abierto del estado del arte existentes para la reconstrucción de estructura, sin embargo se identificaron tres iniciales, Bundler [67], SfM de Philip Torr [74] y SfM de Vincent Rabaud [57]. De estas tres, las dos últimas están desarrolladas en Matlab, mientras que la primera lo está en C/C++. Las razones de usar Bundler frente a las otras dos se pueden resumir en que ha sido desarrollado en un lenguaje de programación eficiente, utiliza algoritmos del estado del arte en visión, y la implementación ha sido utilizada para desarrollar una aplicación comercial. Esto último da una idea de las buenas características de la aplicación o al menos de los algoritmos implementados. Uno de los objetivos iniciales era crear una herramienta propia a partir del código fuente de Bundler y adaptarlo para el trabajo con vídeos de UAV. Sin embargo dada la complejidad de los algortimos internos de Bundler, y los problemas encontrados en las reconstrucciones, esta idea se descartó para este trabajo, aunque pudiera retomarse a partir de los resultados presentados en esta memoria. Adicionalmente a los problemas de eficiencia de Matlab, hay que destacar los problemas con las versiones y la compatibilidad de cada versión con las anteriores. Otra razón para descartar las aplicaciones basadas en Matlab es la falta de soporte o de ejemplos de la fiabilidad. Las aplicaciones opensource no llevan ningun tipo de mantenimiento ni ayuda detrás, por lo que ante cualquier problema la única opción es escribir al autor del código y confiar en que disponga de tiempo para contestar las preguntas. Bundler, aunque también peca de este problema tiene la ventaja de que esta basado en tecnologia y desarrollado por el autor de un software probado y usado para herramientas comerciales. Además es sencillo contactar con el autor para responder cuestiones que surjan sobre el programa.

5.6.1.3. Instalación y configuración de Bundler

Una vez seleccionada la aplicación se procedió a descargarla de la web e instalarla en el ordenador. Para poder usarla fue necesario instalar también Cygwin. Por último se configuró para poder ejecutarla. Todo esto se describe en el manual de Bundler en el apéndice B.

5.6.1.4. Pruebas iniciales y filtro

Las pruebas iniciales con los vídeos del SIVA fueron poco satisfactorias. El problema identificado debido a los puntos tomados en los caracteres sobreimpresos provocó que se modificase el plan de trabajo para añadir un filtro que los eliminase. Gracias al filtro, se elimina totalmente la presencia de puntos correspondientes erróneos presentes en los caracteres sobreimpresos en los vídeos y se consigue reducir la incertidumbre en la caracterización de la nube de puntos. Sin embargo el escenario aun así no era visualmente reconocible. De las imágenes de vídeo se conoce que el zoom es constante sin embargo los valores de focales devueltos son negativos o muy dispares. Por ello se volvió a probar forzando la focal con mejores resultados pero sin mejoras significativas. En este momento se decide el desarrollo de la herramienta de validación Orto3D, para facilitar la tarea de trabajo con los resultados.

5.6.1.5. Implementación de Orto3D

En primer lugar se identificaron los requisitos, la funcionalidad necesaria (a implementar), y la apariencia que debiera tener. De esta primera identificacion se definieron las pautas para el diseño de la aplicación y su posterior implementación. Los requisitos iniciales es que sea una aplicacion desarrollada en Matlab, que además tenga una interfaz gráfica fácil de utilizar, y que contenga toda la funcionalidad necesaria para la validación y visualización del escenario. Entre la funcionalidad básica se incluye la presentación de puntos característicos en las imágenes, los puntos en el escenario 3D, la posición de las cámaras, así como los puntos añadidos de forma manual. Además se incluye funcionalidad para reducir la incertidumbre introducida en los puntos marcados de manera manual mediante la ayuda de las líneas epipolares y adicionalmente de la correlación. Los puntos seleccionados manualmente ayudan a reconocer los objetos además de permitir la validación de los escenarios reconstruidos. La triangulación será válida sólo si las cámaras reconstruidas son correctas. Con esta funcionalidad, es posible añadir aristas o cuadriláteros seleccionados por el usuario, facilitando el reconocimiento de objetos, por ejemplo tejados o paredes en el escenario. Una arista se usa para definir el eje z del sistema de coordenadas del escenario. Si se marca la arista de una pared perpendicular al suelo, entonces se tiene todo el escenario referido a esta arista. El prototipo incluye las siguientes funciones:

- Selección de cuadriláteros.
- Selección de aristas.
- Selección de eje Z.
- Georreferenciación.
- Selección de puntos manuales con epipolar.
- Selección de puntos con epipolar.

- Guarda de puntos y cámaras en coordenadas UTM.
- Guarda de escenario en formato PLY.
- Representación de las cámaras en el escenario 3D.
- Sobreimpresión de puntos característicos sobre la imagen.
- Interfaces de diálogo con avisos al usuario.
- Triangulación de puntos característicos.
- Presentación de los ejes principales de las cámaras.
- Marcado de un punto de referencia en el plano suelo (XY).
- Marcado en la imagen satélite de puntos UTM de referencia para luego usarlos como referencias en la georreferenciación.
- Lectura de ficheros de puntos de referencia ya guardados.
- Presentación de resultados en 3D.
- Zoom para ayudar en el marcado de puntos.
- Obtención de un fichero de datos de la reconstrucción.

5.6.1.6. Pruebas con la herramienta de validación

Se prueban los resultados obtenidos con diferentes escenarios, tanto con fotografías como con vídeos. Para cada uno de estos escenarios se verifica la calidad de la reconstrucción, y se recogen los resultados. Algunos de los resultados se presentan en la memoria. Los escenarios usados son fotografías y vídeos tomados de documentales, así como los escenarios que usan fotogramas del SIVA. Estos últimos son los mismos escenarios usados para las pruebas iniciales pero revisados tras las conclusiones obtenidos con las fotografías y los documentales, además de algún escenario nuevo.

5.6.1.7. Redacción de la memoria, manuales y ayuda

La última fase del trabajo se dedicó a redactar la memoria del trabajo así como los manuales de Bundler y Orto3D. El manual de Bundler se ha incluido para aportar algo más de informacion sobre la herramienta. Actualmente la documentacion esta distrbuida en la información de la web, varios artículos, una tesis doctoral, y los readme.txt de la distribución. Con este manual se pretende aglutinar toda la información en un unico documento. Se trata de una versión inicial que puede completarse y mejorarse en sucesivas actualizaciones para proporcionar un manual completo y fácil de seguir por cualquier tipo de usuario. Se incluye también un manual de usuario de Orto3D con todos los pasos necesarios para georreferenciar un escenario. Además se incluye una ayuda html para Orto3D y otra ayuda html generada con Doxygen para Bundler.

5.7. Herramientas empleadas

En esta sección se comenta brevemente cada una de las herramientas hardware y software empleadas, así como las fotografías y vídeos usados en las pruebas.

5.7.1. Hardware

Ordenador Sobremesa

PC Compaq, Intel Pentium 4, 2.40GHz, 512MB DDR RAM, disco duro 32GB con todo el software necesario instalado.

Ordenador Portátil 1

Portátil Toshiba Satellite A50-522, Intel Centrino Pentium M 715 1.50GHz, 512MB DDR RAM, disco duro 40GB con todo el software necesario instalado. Ordenador usado para la implementacion del código fuente, ejecutar resultados de Bundler, etc.

Ordenador Portátil 2

Portátil HP Pavillion dv6 2180es, Intel Core i7 1.60GHz, 4GB RAM, disco duro 500GB, tarjeta gráfica NVIDIA GeForce GT 230M, con todo el software necesario instalado. Ordenador usado para la ejecución de pruebas cn Bundler y Orto3D, visualización de resultados y redacción de la memoria. Dado que Bundler es una aplicación pesada y que Orto3D al estar implementada en Matlab también es muy pesada, ha sido necesario emplear un ordenador con mayor capacidad de procesamiento que 5.7.2 y 5.7.1.

Cámara de fotos digital

Para las pruebas con fotografías he usado una cámara compacta de uso convencional, con 4.0 Megapixels de resolución (hasta 2272×1704 píxels), longitud focal 5.8-23.2mm (equivalente 35-140mm), y zoom óptico 4x. Se han realizado fotografías para pruebas con Bundler usando imágenes. Para usar Bundler no es necesario tener imagenes con mucha resolución, con una cámara con esta resolución es suficiente. De hecho si las imágenes son muy grandes (en tamaño de píxel), la implementación SIFT de Lowe [35] tiene problemas para manejarlas y obtener los puntos característicos, llegando a abortar la aplicación. En concreto se han probado imágenes de otra cámara con tamaño de 2539×2324 píxels, que SIFT no ha sido capaz de ejecutar en los ordenadores 5.7.2 y 5.7.1, puede que debido a la poca memoria RAM de la que disponen.

Las características técnicas de la cámara son las siguientes[10]:

- Fabricante: Canon
- Modelo: PowerShot A520.
- Resolución: 4,0 Megapixels.
- Longitud focal f: $5{,}8 - 23{,}2mm$ (equivalente $35 - 140mm$).
- Luminosidad: $f/2{,}6$ a $f/5{,}5$
- Zoom óptico: $4x$.
- Tipo CCD: 1/2,5”

[10]Datos de la cámara tomados del manual. Datos del sensor CCD tomados de http://www.dpreview.com/learn/?/key=Sensor_Sizes , http://en.wikipedia.org/wiki/Digital_photography y http://en.wikipedia.org/wiki/Charge-coupled_device

- Diametro CCD: 10,160mm
- Ancho CCD: 5,760mm
- Alto CCD: 4,290mm
- Diagonal CCD: 7,180mm
- Tamaño CCD: 24,7mm^2

5.7.2. Software

A continuación se presenta una lista del software empleado en el trabajo:

1. Windows 7
2. Windows XP
3. Bundler
4. Cygwin
5. Meshlab
6. Scanalyze
7. Matlab
8. Virtual Dub
9. VLC
10. ImageJ
11. VGG MultiView Compute Library
12. Implementación SIFT de David Lowe
13. Sigpac
14. Google Earth
15. Microsoft Visual Studio Express 2008
16. Microsoft Visual Studio Express 2010
17. Microsoft Visio 2003
18. OpenOffice
19. LEd
20. MiKTeX
21. PrintScreen
22. VidShot Capturer

En los apartados siguientes se comenta brevemente cada una y para qué han sido empleadas en el trabajo.

Windows XP - Windows 7

Todo el trabajo se ha realizado sobre estos sistemas operativos aunque Bundler también puede usarse sobre Linux. Todas las herramientas anteriormente citadas han sido instaladas sobre estos sistemas sin problemas de incompatibilidad. Aunque Bundler se puede usar sobre Linux, puede que el resto de aplicaciones no. Tampoco se ha probado el prototipo sobre Linux.

Cygwin

Para poder utilizar Bundler en un ordenador con Windows es necesario disponer de una instalación de Cygwin. Se trata de un entorno emulador de Linux para Windows consistente en dos partes, una que actúa como un emulador de Linux, y otra parte que contiene una colección de herramientas con la misma funcionalidad que en Linux. Las herramientas se pueden instalar de forma independiente en función de las necesidades del usuario. Es un programa de software libre, y por tanto gratuito bajo las condiciones de la licencia GPL. En el manual de Bundler se incluye una pequeña guía para su instalación (sección A.1). La versión utilizada es la 1.7.5.

MeshLab

Es un programa con funcionalidad similar a Scanalyze (ver apartado 5.7.2) pero con una interfaz de usuario mucho más atractiva, más sencilla de usar, y más eficiente (menos pesada que Scanalyze). Tiene implementada una mayor funcionalidad, es de código abierto y la actualización de versiones es frecuente, incluyendo nueva funcionalidad en cada nueva versión publicada. La versión empleada es la $V1.2.3$

Scanalyze

Herramienta de visualización de archivos PLY desarrollada en la Universidad de Stanford, al igual que el formato PLY. Este formato es el usado por Bundler para obtener un resutlado visual de la reconstrucción. Su contenido se basa en una descripción de objetos mediante vértices y polígonos. Es una herramienta de visualización muy sencilla pero también lenta y de funcionalidad reducida. Empleada la versión 1.0.3.

Matlab

Es un programa de cálculo matemático con un lenguaje de alto nivel para cálculo técnico y un entorno interactivo que permite implementar tareas complejas. dada la gran cantidad de funciones matemáticas disponibles y la facilidad de programación, el tiempo de desarrollo de aplicaciones en Matlab es inferior al basado en lenguajes como C, C++ o Fortran, reduciendo esfuerzos de implementación de prototipos o aplicaciones de pruebas. Además de las funciones incluidas en el paquete original hay disponibles en internet una gran cantidad de funciones para visión desarrolladas por la comunidad. Las aplicaciones desarrolladas se han implementado en este lenguaje a excepción del filtro codificado en C y que se ha compilado con gcc. Otras opciones interesantes que permite es la implementación de aplicaciones con interfaz gráfica gracias al editor GUIDE y la documentación en HTML de las funciones implementadas de forma sencilla con la función *publish*. Empleada la versión $R2008b$.

Vitual Dub

Herramienta de procesado y captura de vídeo. Utilizada para extraer los fotogramas de la secuencia de vídeo. Es una herramienta gratuita y bajo licencia GNU GPL. Para obtener los fotogramas en VirtualDub, primero seleccionar un fotograma de inicio y otro de final, luego ir a *File>Saveimagesequence* y el programa devolverá toda la secuencia de fotogramas desde el fotograma indicado como inicio hasta el indicado como final. Es importante modificar las opciones de imagen de salida de forma que las imágenes sean tipo JPEG, con extensión $*.jpg$. Si no se modifica esta opción, VirtualDub obtiene los ficheros de imagen con extensión $*.jpeg$ por defecto, que el script *RunBundler.sh* no reconoce y no toma como imagen de entrada. Empleada la versión $v1.9.9$[11].

VLC

Es un reproductor de vídeo de software libre bajo licencia GPL que a parte de reproducir incluye funcionalidad para extracción de fotogramas, aunque ésta es menos potente que la de VirtualDub pudiéndose hacer únicamente imagen por imagen (en la versión usada en este trabajo). La ventaja frente a VirtualDub es que permite leer la mayoría de formatos de vídeos sin instalar ningún códec adicional, al contrario que con VirtualDub con el cual se encontraron problemas al trabajar con los vídeos documentales.

Para obtener fotogramas con VLC, es necesario ir a $Video>Capturadepantalla$. Antes hay que modificar los parámetros de salida de la imagen en $Herramientas>Preferencias>Video$ y ahí buscar el apartado *Capturadevideo*. Por defecto VLC obtiene los fotogramas en formato PNG. Se puede elegir el prefijo, al igual que la numeración de los fotogramas y el directorio donde se desean guardar. La versión utilizada es la 1.0.5[12].

ImageJ

Herramienta de tratamiento de imagen con un alto potencial y muy flexible ya que a la funcionalidad básica se le pueden añadir multitud de plugins desarrollados por la comunidad ImageJ (se trata de un programa de código abierto). La versión es la 1.43[13].

VGG MultiView Compute Library

Conjunto de funciones en Matlab de visión computacional desarrollada por el grupo de robótica de la Universidad de Oxford[52]. Las funciones han sido desarrolladas a partir del libro publicado por Hartley y Zisserman. Algunas de estas funciones han sido modificadas e integradas en el prototipo Orto3D.

SIFT de David Lowe

Implementación del algoritmo SIFT realizada por David Lowe y disonible de forma gratuita en [67] (versión 4). Es la implementación original del inventor de SIFT disponible en un ejecutable para Linux y otro para Windows y código Matlab para la evaluación de los resultados. El ejecutable ha sido usado en la pipeline de Bundler y el código Matlab se ha tomado y modificado para añadirlo al prototipo Orto3D. El código fuente no es abierto, por lo que no se puede acceder a la implementación interna. Si se requiriese

[11] http://www.virtualdub.org/
[12] http://www.videolan.org/vlc/
[13] http://rsbweb.nih.gov/ij/

código opensource, existen otras implementaciones de SIFT abierta desarrolladas por otros autores como Andrea Vedaldi[14].

Sigpac

Sistema de Información Geográfica de Parcelas Agrícolas, versiones empleadas 5.4.4 y 6.3.0 [45]. Es una aplicación online del Ministerio de Medio Ambiente, y Medio Rural y Marino, que permite obtener ortoimágenes de cualquier punto de España. En su inicio se concibió con el objetivo de facilitar el control administrativo de parcelas, aunque actualmente, por su calidad, disponibilidad y facilidad de manejo, se emplea para otros usos tales como geología, infraestructuras o urbanismo. La gran mayoría de las ortoimágenes empleadas durante la formación se han tomado de esta herramienta. Para las coordenadadas terrestres usa ED-50 en las primeras versiones, mientras que la última permite también trabajar con WGS84, ETRS89 y REGCAN95. La última versión está basada en Microsoft Silverlight por lo que es necesario instalar previamente esta aplicación.

Google Earth

Aplicación que permite visualizar imágenes satélite. Ha sido usada como complemento a Sigpac. Notar que hace uso del sistema WGS84. Se ha usado la versión 5[15].

Entornos Desarrollo

Para compilar y trabajar con el código fuente de Bundler se han usado los entornos de desarrollo (IDE) Microsoft Visual Studio 2008 Express y Microsoft Visual Studio 2010 Express. Son ediciones gratuitas de estos IDEs[16].

Otras herramientas

En este subapartado se comentan las aplicaciones de ofimática OpenOffice, Microsoft Visio, los editores de LaTeXy un par de aplicaciones para captura de pantalla y vídeo. OpenOffice es una herramienta de ofimática gratuita y opensource[17]. Se ha empleado para la realización de documentos, presentaciones y figuras. La versión empleada ha sido la 3.2. Microsoft Visio se ha empleado para la creación de figuras. La versión utilizada es la 2003 La memoria se ha editado totalmente en LaTeXcon las aplicaciones MiKTeX[18] (versión 2.8) y LEd[19] (versión 0.53). Por último, para las capturas de pantalla se ha empleado PrintScreen[20] versión 4.4 y para la captura vídeo de pantalla se ha empleado VidShot Capturer[21] versión 1.0. Estas herramientas son gratuitas pero no son opensource.

5.7.3. Fotografías y vídeos para las pruebas

Se han usado escenarios de tres tipos: fotogramas grabados por un UAV (SIVA), por otro, fotografías tomadas con una cámara de fotos digital compacta y fotogramas de doc-

[14] http://www.vlfeat.org/ vedaldi/code/siftpp.html
[15] http://earth.google.es/
[16] http://www.microsoft.com/express/
[17] http://www.openoffice.org/
[18] http://miktex.org/
[19] http://www.latexeditor.org/
[20] http://www.latexeditor.org/
[21] http://www.geovid.com/VidShot_Capturer/

umentales para televisión tomados desde helicópteros. A continuación se detalla cada tipo de imagen.

Vídeos SIVA

El trabajo se ha centrado en la aplicación a vídeos tomados desde plataformas UAV. La única fuente de imágenes real con secuencias tomadas de UAV han sido las tomadas con el SIVA. Debido a los problemas de estas imágenes, de baja resolución, tomadas desde distancias lejanas a los objetos grabados, con poca línea base entre cámaras, con secuencias cortas de grabación y caracteres sobreimpresos, no se han obtenido los resultados esperados. Además de la baja resolución, hay que tener en cuenta el efecto producido por la compresión aplicada al vídeo para su transmisión a la estación terrena de control. A esto se suma que el marcado manual de puntos en estos fotogramas es muy impreciso, ya que es dificíl discernir con precisión los contornos o esquinas de paredes o tejados en la imagen. En la figura 5.82 se muestra un fotograma tomado con el SIVA, donde se puede apreciar la información sobreimpresa. Las posibilidades de reconstrucción son muy limitadas principalmente por las características de las imágenes empleadas y el vuelo realizado por el vehículo (distancia a objetos). Los tamaños de los fotogramas usados son 352×288 y 720×576 píxeles. Aunque los segundos tienen el mismo tamaño que los de los documentales, los problemas anteriormente comentados hacen que no sean suficientes para reconstruir el escenario. Lamentablemente, los fotogramas de ALCONADA_1, que son la mejor secuencia disponible del SIVA, tienen el primer tamaño. Dado que no se ha podido tener acceso a secuencias de mayor calidad, se han buscado vídeos documentales tomados desde el aire con buena calida de imagen y que se describen en a continuación.

Figura 5.82: Fotograma tomado con el SIVA (1)

Fuente: INTA

Fotografías de escenarios

Ante los problemas con los vídeos tomados con el SIVA y sobre todo a la falta de secuencias de vídeo de escenarios útiles para la implementación del prototipo, se propuso

la utilización de fotografías realizadas con cámaras convencionales de escenarios perfectamente identificados. En otras palabras, dejando a un lado el trabajo con vídeos de UAV, se buscaron escenarios donde se pudieran obtener imágenes de alta resolución para la implementación del prototipo Orto3D. El escenario usado para el desarrollo del prototipo es POLITÉCNICA, mostrado en la figura 5.83, aunque también se han realizado pruebas con fotografías de otros escenarios. Este escenario es una de las esquinas del edificio de la

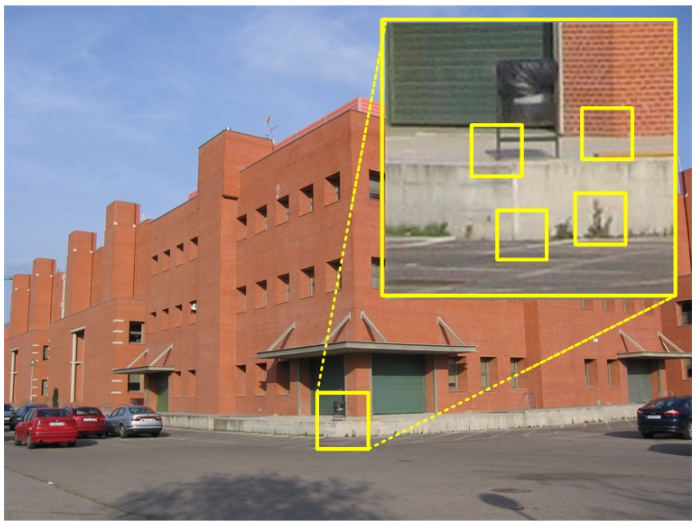

Figura 5.83: Fotografía del escenario POLITECNICA

En la imagen se puede ver que la resolución de la imagen permite visualizar con total nitidez los vértices de los objetos presentes en la imagen, facilitando el marcado de puntos característicos. En la ampliación de la imagen arriba a la derecha se puede ver con cierta nitidez la papelera, el muelle de carga, la pared o incluso una planta.

Fuente: e.p.

Escuela Politécnica de la UAH (Alcalá de Henares). Las razones que llevaron a elegir este edificio frente a otros fueron su forma, de planta cuadrada y la ausencia de árboles u otros elementos que ocultasen la vista del edificio principal. Disponer de un edificio de planta cuadrada es interesante por la facilidad para reconocer visualmente la planta del edificio en la estructura 3D y en la ortoimagen. Además, como se puede ver en la figura 5.83, la resolución de la imagen es lo suficientemente buena como para permitir marcar un punto con una incertidumbre pequeña.

Vídeos documentales

Como se ha comentado, para evitar los problemas de los vídeos tomados con el SIVA (ver 5.7.3) se han usado vídeos aéreos para televisión o documentales. Existen multitud de ejemplos donde se usan imágenes aéreas, por ejemplo en la difusión de eventos deportivos como la Vuelta Ciclista a España, o en la realización de documentales sobre ciudades. A efectos prácticos, que el vehículo aéreo usado en la grabación tenga piloto o no, es transparente a la reconstrucción de estructura, y sin embargo trabajar con este tipo de imágenes fácilmente accesibles es mucho más barato que realizar campañas de grabaciones con UAV reales. De todas las opciones posibles se han usado unos documentales de nombre *Madrid desde el aire* y realizados por Telemadrid. Las razones principales de esta elección son la calidad de los vídeos y el tipo de escenas recogidas. Las imágenes estan estabilizadas y practicamente la gran mayoría de secuencias grabadas corresponden a vuelos sobre ciudades y edificios, con secuencias de varios segundos de grabación y vuelos cercanos a los

edificios. Además, la identificación de los escenarios grabados es sencilla, al tratarse de grabaciones dentro la Comunidad de Madrid. Gracias a ello se ha podido identificar fácilmente las ortoimágenes necesarias para las georreferenciaciones. Aunque aquí solo se han empleado estos documentales, otros vídeos pudieran ser igualmente válidos. Los resultados obtenidos con estas imágenes son significativamente mejores que con los vídeos del SIVA, aunque se han observado algunos problemas como se ha comentado en este capítulo. En las figuras 5.84 y 5.85 se pueden ver un par de fotogramas empleados en las pruebas.

Figura 5.84: Imagen aérea de la Plaza Cervantes, Alcalá de Henares

Fotograma tomado de una grabación realizado sobre Alcalá de Henares, donde se puede apreciar la Plaza Cervantes y la Capilla del Oidor. El fotograma se ha empleado para el escenario ALCALA_2. Fuente: documental *Madrid desde el aire*.

Figura 5.85: Imagen aérea de una ermita en Fuente el Saz (1)

Fotograma tomado del mismo documental que la figura anterior y usado en el escenario ERMITA_0. Al ser únicamente un edificio, es fácil reconocer visualmente la estructura 3D devuelta por Bundler. Fuente: documental *Madrid desde el aire*.

Capítulo 6

Conclusiones y trabajos futuros

6.1. Conclusiones

Este trabajo se centra en el concepto de la reconstrucción de estructura aplicado a vídeos tomados desde plataformas aéreas, para a partir de ellos obtener escenarios tridimensionales del entorno observado. El trabajo se basa en el uso de Bundler, una herramienta muy potente de reconstrucción de estructura, fiable y robusta cuando se dispone de longitudes focales de inicialización para las cámaras y muy flexible en cuanto al número de opciones que permite para su ejecución. El hecho de que el código esté basado en algoritmos y técnicas del estado del arte hace que sea muy interesante su estudio y utilización. Además, gracias a que el código fuente sea abierto, se puede trabajar con la aplicación sin que ésta se convierta en una caja negra para el usuario. Incluso si fuera necesario, se podría modificar el código original para adaptarlo a las necesidades concretas identificadas. Debido a la dificultad asociada a la operación con imágenes, cada problema específico en visión requiere una solución propia. Las imágenes pueden ser de diferentes escenarios y tener diferentes iluminaciones, tipos de cámaras, poses, oclusiones e incluso en algunos casos información escrita sobre ellas. Todo esto genera una gran complejidad para su tratamiento. En este trabajo se ha podido comprobar que la presencia de caracteres sobreimpresos en la imagen, las grandes distancias de las cámaras a los objetos y las líneas de base pequeñas influyen negativamente en la reconstrucción. Para eliminar el efecto de los caracteres, la solución aquí propuesta añade una etapa adicional a la secuencia de trabajo de Bundler, mientras que para los otros dos problemas, en este trabajo se define un procedimiento recomendado de operación. La distancia a los objetos es un tema interesante con el objeto de conocer cual es la distancia máxima de operación para los UAVs con respecto a los objetos grabados que permite obtener reconstrucciones de calidad. Esto es una pregunta abierta que pudiera plantearse en un futuro trabajo de continuación.

También se ha podido comprobar que disponer o no de datos iniciales sobre las cámaras permite dar mejores o peores resultados y más o menos fiables. Bundler permite establecer varias restricciones para los parámetros iniciales en función de las opciones de entrada disponibles. De todas se ha verificado que el valor inicial de la focal tiene una relevancia absoluta en la solución final, especialmente si se trata del primer par. La elección del primer par es decisiva en cuanto a que de él dependerá el resto de la reconstrucción. Cada nueva cámara se añade en función de esta estimación inicial. Cualquier error de estimación que en éste se produzca, se arrastrará en cascada a lo largo de toda la reconstrucción, llevando a soluciones finales erróneas. Si se conoce con precisión el valor de la focal para todas las

cámaras o al menos para las dos imágenes iniciales, la probabilidad de que el escenario sea óptimo es mayor. Como se ha podido comprobar, también se pueden obtener resultados interesantes cuando los valores de inicialización son estimaciones cercanas al valor final. Al contrario, cuando no se dispone de ninguna información sobre el escenario, hay una gran probabilidad de que la reconstrucción sea errónea. Este hecho se ha podido comprobar a partir de las pruebas realizadas con los fotogramas de documentales. Aunque para alguno de estos escenarios la resolución y la distancia de la cámara a los objetos no debiera causar grandes problemas a priori, el no disponer de un valor de inicialización adecuado para la focal provoca que las estimaciones para algunas cámaras sean incorrectas. Al trabajar con el mismo escenario, pero forzando los valores de las focales a unas estimaciones, los resultados son significativamente mejores. Toda esta problemática se debe al método empleado en la estimación del par inicial. Éste es el algoritmo de Níster, que emplea cámaras calibradas. Bundler hace una serie de suposiciones sobre las cámaras, como que el punto principal de la imagen está en el centro de ésta, que el factor de cizalla (*skew parameter*) es nulo y que la relación de aspecto[1] es igual a la unidad. De la última suposición se puede decir que la longitud focal es igual en la horizontal que en la vertical y que además es conocida. El valor conocido de la focal se extrae de la cabecera EXIF y cuando la imagen no dispone de esta información, se inicializa a $f = 532$ por defecto o como se ha realizado en este trabajo, a un valor forzado manualmente. Esta inicialización tiene un impacto significativo en la robustez, fiabilidad y precisión de los datos, por lo que debiera buscarse algún método más robusto de estimación del par inicial a partir de cámaras no calibradas.

Bundler, aunque es una herramienta robusta, carece de método alguno de autocalibración. Toda la reconstrucción depende de valores iniciales para las focales, como se ha comentado en el párrafo anterior. Este defecto es importante de cara a trabajar con fotogramas no calibrados. Por esta causa, los resultados obtenidos en el trabajo no son precisos como se esperaba en un inicio. En cierto modo se esperaban resultados mejores en este sentido, dada la buena apariencia visual de los presentados en [3][4][67][69][70][71]. En estos escenarios no se usan cámaras calibradas, pero al usar un gran número de fotografías es muy probable que un buen porcentaje de ellas contengan en la cabecera EXIF un valor teórico de su focal, que se usa como inicialización. Aún siendo mucho mayor el subconjunto de cámaras sin focal, el primer par se estimaría a partir de un dato fiable y cercano al valor real, teniendo una inicialización suficientemente robusta. Si además el número de imágenes empleadas con focal EXIF es grande, la reconstrucción será más robusta todavía sin necesidad de emplear un método de autocalibración al final de la secuencia. Como en el contexto de este trabajo no existe la posibilidad de trabajar con imágenes tomadas a partir de cámaras calibradas, sería interesante plantear el uso de un método de autocalibración. Una posible continuación de este trabajo sería incluir una etapa más en Bundler. Esta nueva etapa consistiría en un módulo de autocalibración a partir de los datos estimados en la reconstrucción de estructura [25][54].

El prototipo Orto3D desarrollado en este trabajo aporta una herramienta con funcionalidad para georreferenciar los escenarios obtenidos con Bundler a ortoimágenes. Hasta la fecha, no se ha encontrado ninguna aplicación similar publicada como código abierto que aporte esta funcionalidad. En el trabajo de Photo Tourism[70] se menciona una herramienta desarrollada por el autor de Bundler que permite la georreferenciación de forma interactiva pero sin embargo, ni su ejecutable ni su código están publicados. El trabajo

[1] La relación de aspecto (*aspect ratio*) se define como el ratio de proporción entre el tamaño de píxel en anchura y en altura.

realizado en este prototipo puede servir de base para desarrollar una aplicación en C/C++ o Java que realice esta funcionalidad de forma eficiente e independiente de la plataforma. Por medio de Orto3D, los escenarios georreferenciados permiten obtener medidas aproximadas de objetos o distancias así como de las posiciones de las cámaras. Al trabajar con estimaciones y no disponer de datos métricos de inicialización fiables ni para las cámaras, ni para los puntos, ni tampoco tener datos de calibración, éstas medidas serán imprecisas. Sin embargo, lo interesante del trabajo aquí presentado es que se pueden obtener datos aproximados tanto de los escenarios como del vuelo del vehículo. En ningún caso debiera usarse la proximación aquí seguida con el objeto de obtener medidas fiables o de precisión. El empleo de restricciones adicionales para las cámaras o puntos en la reconstrucción con Bundler permitirá resultados más fiables y precisos que los aquí presentados. En resumen, se puede decir que los resultados obtenidos son tomados en el peor contexto posible, sin tener información alguna del escenario ni de la cámara, así como tampoco de su calibración, y trabajando en algunos casos con imágenes de muy poca resolución.

Debido a la mala característica de los escenarios iniciales, los resultados obtenidos inicialmente con Bundler fueron decepcionantes y muy inferiores a los esperados. Este hecho y la carencia de un buen número de vídeos con los que poder trabajar, obligó a redefinir el método de trabajo en varias ocasiones para dar solución al problema. Para ello se tomaron dos opciones, una primera consistente en usar fotografías para validar la robustez de Bundler en un escenario conocido con imágenes de buena resolución, y la segunda en usar fuentes adicionales de vídeos aéreos. En el primer caso, las fotografías permitieron desarrollar el prototipo de georreferenciación. La buena resolución de estas imágenes, la cercanía de la cámara a los objetos y las estimaciones iniciales de las focales en las cabeceras EXIF permitieron obtener resultados de reconstrucción lo suficientemente válidos como para implementar el prototipo. Además, las fotografías son la finalidad concreta para la que se ha diseñado Bundler, por lo que estos resultados se tomaron como buen punto de partida para desarrollar Orto3D. Para el segundo caso, usar vídeos documentales de television ha sido una forma muy útil de disponer de una fuente de información adicional con la que trabajar sin tener que afrontar los costes de tomar este tipo de imágenes y la complejidad de usar UAV reales.

Con respecto a las aportaciones del trabajo, éste define una línea muy interesante para la implementación de una herramienta de reconstrucción propia a partir de vídeos tomados de plataformas aéreas. Los resultados pueden ser modelos 3D de edificios, estructuras o elevaciones digitales de mapas únicamente a partir de fotogramas aéreos. Estos modelos pueden incluso mejorarse con apoyo de imágenes tomadas en tierra y que en conjunto pueden dar escenarios 3D basados en información real. Los resultados presentados en [4] y [69] permiten dar una idea de las posiblidades de esta línea de trabajo, así como los resultados presentados en [21] y [22] permiten visualizar ejemplos de escenarios densos a partir de reconstrucciones de estructura (SfM) combinadas con MVS. De los anteriores, CMVS y PMVS son los visualmente más interesantes por la posibilidad de usar sus resultados para la implementación de simuladores o navegadores visuales de escenarios. La línea de trabajo aquí presentada puede continuarse con algoritmos MVS como CMVS y PMVS para obtener reconstrucciones densas de edificios, terrenos o estructuras.

En cualquier caso, para obtener resultados interesantes o aplicaciones realizables, la continuación de este trabajo exige recursos materiales y personales suficientes para afrontar su complejidad. Por un lado la obtención de reconstrucciones robustas y fiables exige disponer

de fuentes de vídeo adecuadas, con un número abundante de fotogramas, y unas características de vuelo concretas como se comenta en la memoria. Además, es indispensable disponer al menos de información sobre la focal de las cámaras para que la reconstrucción obtenida no caiga en valores erróneos. Conocer datos sobre la posición de las cámaras a partir de la telemetría de las imágenes o en algunos casos a partir de los caracteres sobreimpresos, aumentará la robustez, fiabilidad y precisión de la reconstrucción. Disponer de abundantes fotogramas permitirá probar con reconstrucciones a gran escala como las presentadas en [4]. Las reconstrucciones a gran escala de escenarios tienen una complejidad mayor que los escenarios mostrados en este trabajo. Por un lado el mayor número de imágenes es más susceptible de generar errores que sean arrastrados a lo largo de la reconstrucción y que acaben partiendo el escenario en dos fragmentos incoherentes, ya sea por similaridad de texturas en la imagen o malas inicializaciones de cámara como se ha podido ver en esta memoria. Un problema relacionado con este tema es el de las derivas [12]. Dado que los escenarios aquí mostrados son pequeños, éstas no se han tenido en cuenta, pero pudiera ser interesante su consideración en trabajos futuros con escenarios más grandes donde la precisión sea imprescindible. Por otro lado, trabajar con un gran volumen de imágenes y datos exige el uso de clústers de computación con múltiples nodos para obtener resultados visualmente interesantes en tiempos de ejecución razonables (del orden de días). En los ejemplos de [4] se usan más de 100 nodos. El uso de ordenadores personales tipo PC o estaciones de trabajo convencionales limita a trabajar únicamente con pequeños escenarios, similares a los mostrados en este trabajo.

A partir de la conclusión anterior se puede afirmar que el uso de aplicaciones de reconstrucción de estructura basados en Matlab tiene menos interés para escenarios muy pesados (con muchas cámaras) en comparación con aplicaciones desarrolladas en C/C++. Pese a que Matlab es un entorno de trabajo cómodo y potente, además de que permite un rápido desarrollo de prototipos o aplicaciones matemáticas, tiene la desventaja de consumir muchos recursos computacionales y ser muy lento en comparación con aplicaciones basadas en C/C++ o similares. Este problema ocurre especialmente cuando se realizan múltiples llamadas a funciones. Las aplicaciones de reconstrucción de estructura operan con una gran cantidad de datos y variables y a su vez se realizan llamadas continuas a procedimientos, que en Matlab son muy lentos en comparación con un código basado en C/C++. Esta afirmación no afecta al prototipo de georreferenciación, que aunque sería más eficiente en C/C++ que en Matlab, la pequeña cantidad de datos con los que trabaja en la optimización permite que los tiempos de respuesta de la aplicación sean razonables. La triangulación más compleja usa como mucho cuatro puntos por imagen (en el caso de marcado de poligonal) que se traduce en doce valores a optimizar como máximo: las tres coordenadas de cada punto. En este caso no se optimiza el resto del escenario, de ahí que no se trabaje con más parámetros.

Como resumen final, el trabajo presentado en esta memoria aporta un nuevo punto de vista a la aplicación de la visión computacional al campo aeroespacial. En concreto la aportación al sector de los UAV es el uso de software de código abierto para la reconstrucción de estructura en postprocesado. El problema propuesto hace frente a un escenario complejo donde se usan cámaras no calibradas y ningún dato del escenario ni de las poses. Los resultados presentados demuestran que en el estado en el que se encuentra el trabajo, no se puede implementar una aplicación fiable de SfM con cámaras no calibradas. Sin embargo, permite definir futuras líneas de desarrollo que sí pudieran dar resultados interesantes de cara a una aplicación práctica. Otro de los beneficios aportados es la ob-

tención de unos conocimientos sobre un campo de la visión que tendrá un desarrollo en el futuro interesante, como indica el interés de ciertas compañías u organismos en este tipo de desarrollos[2]. Este trabajo se puede continuar con otros trabajos fin de carrera o de investigación que exploten las ideas aquí presentadas.

6.2. Trabajos futuros

Los posibles trabajos de continuación son los siguientes, con un pequeño comentario en cada uno:

1. **Triangulación con una línea base pequeña y efecto de la resolución y compresión en los resultados SfM.** Uno de los problemas observados en el trabajo es la mala calidad de los puntos triangulados a partir de puntos marcados manualmente cuando las cámaras usadas tienen una separación pequeña. Este problema es mayor cuando la resolución de las imágenes es menor o cuando el rango de distancias del escenario es grande. Aunque el problema está bien identificado, no está cuantificado. Pudiera ser interesante realizar un trabajo adicional en esta línea, ya sea mediante la búsqueda de documentación o realización de pruebas con imágenes de diferente resolución. Una prueba adicional es la comprobación del efecto que tiene la compresión de las imágenes para vídeo sobre la detección de puntos y por ende en la reconstrucción. Otra comprobación empírica a realizar podría ser el efecto del rango del escenario y la resolución de las imágenes en los resultados de triangulación, empleando fotografías y vídeos bajo varias condiciones.

2. **Usar SURF en Bundler.** Con esta tarea se combina el estudio del algoritmo de búsqueda de correspondencias con un algoritmo de búsqueda de característicos ampliamente extendido como SURF [7]. Éste es un algoritmo de detección de puntos característicos similar en concepto a SIFT, pero no en su implementación. Las características de SURF, basado en imágenes integrales, hacen que presente una velocidad significativamente mayor en comparación con SIFT, haciéndolo muy interesante para aplicaciones de tiempo real[11]. El tiempo de tratamiento de una imagen con SURF se puede reducir si el algoritmo se optimiza para GPU (por ejemplo en tarjetas gráficas NVIDIA con CUDA[3]) [13]. El incoveniente de este método es que no es invariante afín lo cual podría ocasionar problemas en las reconstrucciones. Sin embargo, por los buenos resultados en tiempo de ejecución del algoritmo, sería interesante analizar la posibilidad de sustituir SIFT por SURF. La sustitución sería muy sencilla, ya que el código fuente a modificar es únicamente la etapa de búsqueda de correspondencias (matching). En la etapa de búsqueda de característicos basta con sustituir un ejecutable por otro. Los resultados de las implementaciones originales de SIFT[35] y de SURF[17] son similares en formato, ya que contienen las localizaciones de los puntos, la orientación y un descriptor. La diferencia principal es el tamaño del descriptor devuelto, que para SURF es de 64 componentes frente a las 128 de SIFT, pero esto no lleva asociada una excesiva complicación para su implementación en el código fuente. Un dato adicional no presente en la salida de

[2]El trabajo presentado en Photo Tourism ha sido parcialmente financiado por Microsoft Research, así como PMVS y CMVS han sido parcialmente financiados por Microsoft Research y Office of Naval Research (Estados Unidos)

[3]Compilador y conjunto de herramientas de desarrollo creadas por NVIDIA basadas en una variación de C para la codificación de algoritmos en GPUs de NVIDIA con el objeto de explotar las ventajas de las GPUs frente a las CPUs en tareas complejas, en este caso tratamiento de imágenes.

SIFT es el signo del Laplaciano. Gracias a este valor, la etapa de matching es más eficiente, puesto que se divide el conjunto de puntos en dos subconjuntos en base a este signo para facilitar la búsqueda sin añadir coste computacional[7]. Al usar un vector de menor tamaño, también optimiza la etapa de matching. Si aún así, SURF no devolviese resultados satisfactorios, y se plantease como necesidad importante el disponer de una búsqueda de característicos más eficiente, se podría implementar SIFT sobre GPU como se propone en [65] o sobre FPGA [62]. En cualquier caso la velocidad de la reconstrucción vendrá determinada por las etapas de matching y de optimización (bundle adjustment), que son las tareas más pesadas de la reconstrucción.

3. **Estimación inicial del par con otro método previo al método de Nistér.** Como se ha visto en la sección 4.7.1, el valor inicial de la focal es un parámetro muy importante para obtener una reconstrucción de estructura adecuada. Bundler intenta usar la focal extraída de la cabecera EXIF de la imagen cuando es posible y si no hay cabecera EXIF entonces inicializa esta focal a $f = 0{,}0$ que luego internamente inicializará a $f = 532{,}0$. Este valor puede no estar próximo al valor real causando malas estimaciones del par inicial que serán arrastradas a lo largo de todo el escenario, como se ha visto en los ejemplos comentados en la sección 5.3. En estos escenarios se han obtenido valores de focales negativos para el primer par, debido a usar una focal de inicialización lejana al valor real. Usar el valor de $f = 532{,}0$, solución por defecto para los escenarios que carezcan de focal en la cabecera, es poco adecuado en la mayoría de los casos, pero necesario al estimar el primer par mediante el algoritmo de Nistér. Éste necesita cámaras calibradas para obtener una estimación de la pose de la cámara a partir de los puntos correspondientes, lo que se traduce en la necesidad de conocer una estimación inicial para la focal próxima al valor real. El método de forzado de la focal manualmente con un valor estimado de ésta, aunque da resultados visualmente aceptables, es poco robusto e impreciso, además de no ser óptimo. De ahí la necesidad de encontrar un método más adecuado para la estimación precisa de la focal, incluso cuando no se dispone de cámaras calibradas. El valor obtenido podría usarse como inicialización para el método de Níster. Una opción a estudiar es el algoritmo de Hongdong Li [31] que permite obtener la focal únicamente a partir de dos vistas siempre y cuando éstas tengan la misma distancia focal, condición asumible en secuencias de vídeo sin zoom como las aquí presentadas. En este trabajo se han realizado algunas pruebas iniciales en esta línea, sin obtener ninguna conclusión válida. El algoritmo, al estar implementado en Matlab, es muy pesado y se han experimentado problemas de ejecución en estas pruebas iniciales (con el ordenador de sobremesa y el portátil 1 descritos en 5.7.1). Sería interesante hacer una revisión de esta tarea con un ordenador más potente (como el portátil 2 descrito en 5.7.1). La actividad se centraría en evaluar los resultados de aplicarlo a fotografías con focal conocida y a los fotogramas aéreos de este trabajo. Eventualmente, si los resultados fuesen válidos, se podría implementar este algoritmo en C/C++ para añadirlo a la secuencia de Bundler antes de llamar al algoritmo de Nistér. En esta actividad, también deberían buscarse otros métodos del estado del arte para evaluar las diferentes aproximaciones existentes.

4. **Añadir un método de autocalibración a Bundler.** Otra opción adicional es añadir un módulo en la secuencia de Bundler con funcionalidad para autocalibración. Esto podría evitar el problema encontrado con las focales. El módulo se podría implementar a partir de software del estado del arte de código abierto y en base a

la teoría presentada en [25] y [54] sobre el tema.

5. **Mejora de la aplicación Orto3D e implementación en C/C++.** El prototipo es una herramienta todavía abierta a la que se pueden añadir nuevas funcionalidades. Entre los trabajos de mejora identificados están:

   - Representacion de una gráfica con los puntos correspondientes por pares, tríos, cuartetos, etc. de cámaras, para ver qué combinaciones de cámaras son las que más puntos aportan a la reconstrucción.
   - Representación de la evolución de las focales del primer par y del conjunto de cámaras durante la optimización, así como de los parámetros de distorsión radial[4].
   - Mejora de la triangulación de los puntos marcados manualmente. Al triangular, estos puntos presentan un error grande cuando la línea base es pequeña o el rango de la reconstrucción es grande. Los puntos triangulados son optimizados con LMA, pero pudiera ser interesante optimizarlos con respecto al resto del escenario con SBA.
   - Mejorar la sobreimplesión de la imagen: la sobreimpresión del escenario sobre la ortoimagen se hace a partir de dos puntos (una recta) seleccionada en el escenario y en la ortoimagen. Ésta es una aproximación simple, válida para una primera versión de la herramienta, pero si se desean obtener mejores resultados es necesario modificarla. Una primera opción sería emplear más de una recta para estimar la relación entre la planta de la reconstrucción 3D y la ortoimagen. De esta forma se obtendría un resultado mas estable y robusto. Otra posible opción sería marcar una recta de forma análoga al marcado del eje Z y las poligonales. En estas imágenes se podría tomar una arista bien definida y paralela al suelo que ademas estuviese bien caracterizada en la ortoimagen. Con las coordenadas de la recta en ambos escenarios se estimaría la relación entre escenario y ortoimagen. También se podrían usar varias aristas en esta segunda opción con el objeto de tener una medida más robusta.
   - Realización de una interfaz o programa para poder hacer más sencilla la obtención de las ortoimágenes con los puntos de referencia UTM. Habría que ver cuales son las posibilidades que ofrecen las herramientas SIGPAC o Google Earth para acceder a las coordenadas y si es posible integrarlas en esta interfaz.

   A priori no es necesario codificar el prototipo a código C/C++, Java, Python o algún otro lenguaje eficiente, aunque puede ser una tarea interesante a largo plazo. Matlab es poco eficiente y el código desarrollado depende de que Matlab esté instalado en el ordenador. Una de las ventajas de este cambio es que el programa prototipo sería multiplataforma e independiente de aplicaciones comerciales de código cerrado desarrolladas por terceros. Esta tarea puede ser muy costosa en término de tiempo, por lo que se debe evaluar si este cambio es rentable con respecto al uso que se vaya a dar a la aplicación. Una opción posible sería implementarlo usando alguna librería de tratamiento de imágenes o de visión, como por ejemplo OpenCV [51].

6. **Documentación del código fuente de Bundler.** El código esta pobremente documentado con pocos comentarios y escasa o más bien nula documentacion, en

[4] En la memoria se presentan gráficas de la evolución del primer par, pero esta funcionalidad aunque implementada para las gráficas, no está integrada en Orto3D.

algunos casos insuficiente. En los artículos sobre Bundler [69][70][71] hay partes de la implementación que no se explican, siendo la tesis de Noah Snavely [68] el documento más detallado. En cualquier caso, se describe la implementación muy por encima, sin entrar en profundidad con el código fuente. La información de la distribución contenida en ficheros *readme* o similares se limita a un par de ficheros de texto que describen muy por encima el funcionamiento del programa, los algoritmos internos, el modelo de cámara y los modos de funcionamiento. En ningún documento se presenta en detalle el contenido de cada uno de los ficheros de salida de Bundler. Tan solo con la información disponible el programa queda como una caja negra para el usuario. Navegar por el código introduciendo comentarios para luego generar una documentación adecuada permitiría no solo conocer el código fuente sino además obtener una documentación generada con Doxygen. En el DVD que acompaña la memoria se incluye una versión inicial de esta documentacion con los comentarios originarios de Noah Snavely. Además, la descripción de la implementación de Bundler en 5 y el manual de Bundler incluido en el apéndice A se han redactado con la idea de poder servir de ayuda a esta tarea.

7. **Modificación de Bundler para crear una aplicación de propósito específico para imágenes aéreas a partir de una aplicación de propósito general.** Bundler es una herramienta de propósito general robusta, eficiente y muy interesante desde el punto de vista de los algoritmos que internamente implementa. Sin embargo esta aplicación se puede hacer más eficiente si se consideran unas ciertas características de operacion inherentes al campo de los UAV y que se han podido detallar a lo largo de la memoria. Bundler ha sido diseñado para poder trabajar con imágenes no secuenciales, totalmente aleatorias en localización, escala, tamaño, iluminación, o posición de cámara, para producir a partir de ellas modelos 3D adecuados. Este modo de funcionamiento es muy general, y no está optimizado para el caso concreto de secuencias de vídeo con UAV. Los fotogramas aquí empleados tienen continuidad al ser del mismo fragmento de vídeo. Además, aunque no se conozcan los parámetros de calibración de las cámaras, se conoce que son iguales para todos los fotogramas de la secuencia. También se sabe que las imágenes seguirán una trayectoria y la separación entre cámaras vendrá determinada por la frecuencia de muestreo de los fotogramas y por la velocidad del vehículo. A partir de estos datos se pueden definir unas restricciones de operación a Bundler con el fin de modificar el código fuente y crear así una nueva aplicación optimizada.

8. **Aplicar un programa de MVS a la salida de Bundler.** El objetivo sería obtener reconstrucciones densas de escenario. Estas reconstrucciones georreferenciadas podrían usarse para crear mapeados 3D o simular escenarios. Consistiría en aplicar el trabajo presentado en [21] y [22] pero con imágenes aéreas.

9. **Creación de una base de datos de vídeos.** Mediante las coordenadas de las cámaras georreferenciados se puede implementar una base de datos con una herramienta de búsqueda. El objetivo sería navegar en un conjunto grande de escenarios y organizar la información guardada en función de las coordenadas obtenidas tras la georreferenciación. Como resultado se podría acceder a los vídeos o secuencias de éstos de una manera organizada en base a la localización de los escenarios grabados.

10. **Aplicación de otras técnicas para trabajar con vídeo en tiempo real.** Aunque se sale de la línea presentada hasta ahora, es una opción interesante de cara al trabajo con UAVs. Este TFC se ha realizado pensando en una herramienta

de reconstrucción de estructura para postprocesado del escenario. En otras palabras, Bundler no se puede usar en un entorno de tiempo real. La reconstrucción de estructura basada en bundle adjustment es un método muy caro en cuanto a coste computacional y carga de trabajo, descartando su uso en entornos con restricciones temporales [16]. La mayoría de los métodos de tiempo real vistos hasta la fecha se basan en SLAM. Aunque el uso de SLAM se sale de la línea principal de trabajar con métodos SfM, es interesante por las posibilidades de aplicación que aporta al campo de los UAV. Éste método permite mapear un escenario a la vez que estimar la localización de la cámara. A diferencia de SfM, para SLAM se conocen los parámetros de calibración de la cámara y se asumen algunas restricciones sobre los movimientos de la cámara para optimizar la búsqueda de correspondencias. En comparación Bundler usa un método exhaustivo de búsqueda. Otra característica es que usa un número muy inferior de puntos correspondientes, con la ventaja de ser más eficiente y el coste de tener una solución no densa. En esta línea están varios trabajos realizados con UAV, como por ejemplo el comentado en [10]. Las aplicaciones pueden ser programas de seguimiento y detección de objetos o mapeado de terrenos.

Parte II

Pliego de condiciones y presupuesto

Capítulo 7

Pliego de condiciones

El proyecto es válido bajo una condiciones concretas de trabajo que a continuación se enumeran. Los recursos hardware necesarios son:

- Un ordenador con capacidad de proceso adecuada para trabajar con herramientas de procesamiento de imágenes y aplicaciones de cálculo pesadas. Dos de los ordenadores descritos en 5.7.1 son suficientes, pero pudieran experimentarse problemas de lentitud durante la ejecución con escenarios pesados (la mayoría de los empleados en este trabajo). Para un rendimiento adecuado se aconseja utilizar ordenadores con mayor capacidad de cómputo que los anteriores, por ejemplo el más avanzado de los descritos en 5.7.1 o similares.
- Cámara de fotos digital convencional para pruebas iniciales. Puede emplearse una camara similar a la utilizada en este trabajo (ver en el apartado 5.7.1) o superior. Cámaras de caracteristicas inferiores, de teléfonos móviles o webcams pueden ser igualmente válidas aunque no se ha probado su funcionamiento.

Los recursos software imprescindibles en el ordenador son los siguientes:

- Sistema Operativo Windows XP o Windows 7 instalado en el ordenador. No se ha comprobado con versiones inferiores ni con Windows Vista. Tampoco se ha probado en MAC OS ni en Linux.
- Bundler, preferentemente versión $v0.3$ o posteriores.
- Cygwin, versión 1.7.5 o posteriores.
- Meshlab, versión $V1.2.3$ o posteriores.
- Matlab, versión $R2008b$ o posteriores.
- VLC, versión 1.0.5 o posteriores.
- VirtualDub, versión $v1.9.9$ o posteriores.
- SIFT, versión 4 de la distribución de David Lowe.

Adicionalmente pudiera necesitarse software adicional descrito en 5.7.2, sin embargo la descrita aquí es la configuración mínima.

Las imágenes tomadas con la cámara convencional son necesarias sólo para pruebas básicas con Bundler, antes de probar con imágenes aéreas. Las fotografías no necesitan cumplir ninguna condición específica a priori, aunque en la práctica, las imágenes aéreas sí deben cumplir algunas condiciones:

- Las imágenes aéreas pueden ser tomadas tanto por vehículos áereos no tripulados como tripulados, siendo más interesantes los primeros por su gran rango de aplicaciones prácticas.
- A priori se puede emplear cualquier tipo de UAV que disponga de una cámara, siendo las características de la imagen y no el tipo de vehículo el que determine la validez de la reconstrucción. Sin embargo el tipo de vehículo aéreo, el tipo de cámara embarcada y el canal de transmisión pueden ser relevantes. El primero es determinante en la estabilidad de la cámara, por ejemplo por la vibración del motor o la inestabilidad del aparato. El segundo interviene en la estabilidad de la imagen y la resolución, entre otros factores aquí no identificados. El tercero influye en la resolución y compresión de la imagen. En cualquier caso el trabajo aquí presentado debiera probarse con la configuración específica requerida.
- Las imágenes empleadas pueden ser tanto fotografías como fotogramas de secuencias de vídeo. La resolución de éstos son un factor determinante en la reconstrucción.
- Los caracteres sobreimpresos en vídeos aeroportados tienen un efecto negativo en las reconstrucciones con Bundler. Para reducir éste se debe usar un filtro que elimine los puntos característicos detectados sobre estos caracteres, ya sea siguiendo la idea presentada en este trabajo o una aproximación similar.
- Este filtro no es necesario si no hay caracteres sobreimpresos en la imagen. En cualquier caso, si se mantiene el filtro en la secuencia de Bundler, no tendrá ningún efecto a la salida cuando la imagen carezca de caracteres sobreimpresos.

En cuanto a las condiciones adicionales de trabajo:

- Para georreferenciar un escenario es necesario tomar una ortoimagen de alguna herramienta como SIGPAC o Google Earth con unas coordenadas UTM de referencia (puntos de referencia).
- En el caso de trabajar con escenarios situados en zonas rurales en España, se recomienda usar SIGPAC preferentemente, al disponer éste de ortoimagenes de todo el territorio español.
- El número de puntos de referencia en la ortoimagen debe ser como mínimo de dos, pudiendo usar un número máximo no definido.
- Si las imágenes disponen de cabeceras EXIF con la distancia focal, éste valor se usará para inicializar la reconstrucción. En caso de no disponer de EXIF, será necesario forzar esta inicialización.
- El valor forzado de la longitud focal será en todo caso una estimación, pudiendo no estar próximo al valor real. Asignar un valor de 1,2 veces el ancho de la imagen es un método de inicialización usado cuando no se conoce valor alguno. Se trata de un valor empírico que puede no estar próximo al valor real, pudiendo presentar el problema de caer en un mínimo local. Otra opción es ejecutar dos veces Bundler, la

primera sin forzado y la segunda vez con forzado a partir de los valores obtenidos en la primera ejecución y suponiendo que todos los fotogramas se han tomado con la misma cámara. En cualquier caso este método no puede usarse para obtener escenarios fiables y precisos.

- Se han usado escenarios de los que se tiene un número bajo de fotogramas, inferior a 200. No se ha probado con un rango de imágenes mayor. Con un número mayor pudieran presentarse problemas relacionados con derivas en el escenario. Ver [12] y [68] para más información sobre este problema.

- El método de reconstrucción de estructura no permite obtener resultados en tiempo real, por lo que Bundler no es una opción si el tiempo es un requisito indispensable. En este caso, buscar otras aproximaciones diferentes.

- Las reconstrucciones obtenidas con Bundler y georreferenciadas con Orto3D no son precisas. Las posiciones de las cámaras y sus parámetros son estimaciones sin información a priori. Por ello, tanto las cámaras como los puntos tendrán errores de reconstrucción a los que se sumará el error de posicionamiento añadido por Orto3D. En ningún caso se puede usar este trabajo para obtener medidas precisas de escenarios, menos aún si los datos de calibración de cámara son desconocidos.

- El marcado manual de puntos en los escenarios con Orto3D, para a continuación triangularlos y añadirlos a las recosntrucciones, debe ser lo más preciso posible. El error de marcado puede tener un efecto considerable en el resultado final de la reconstrucción.

- La triangulación en Orto3D se basa en el marcado manual de puntos en tres imágenes. Para que el resultado de la triangulación sea correcto, es necesario que las tres cámaras que forman estas imágenes tengan una línea base bien separada entre sí. La separación mínima vendrá determinada por el rango de distancias del escenario.

- Para llevar a cabo un trabajo de estas características es necesario tener conocimientos sobre el campo de la visión computacional. El desarrollo del prototipo Orto3D, el entendimiento de Bundler y la evaluación de los resultados necesita éstos conocimientos. Para el caso de un usuario de estas aplicaciones, basta con un conocimiento básico en este campo, siempre que use los métodos recomendados de trabajo (sección 5.5). Para un funcionamiento más complejo, es indispensable conocer el tema con mayor profundidad.

El trabajo aquí presentado también pudiera aplicarse a sistemas terrenos no tripulados (UGV) o automóviles, aunque esta aplicación no se ha probado en la práctica.

Capítulo 8

Presupuesto

En las siguientes tablas se muestra el coste de los recursos necesarios para llevar a cabo este proyecto. El beneficio industrial y los honorarios se estiman en un 15 % del coste total antes de impuestos[1][2].

Software	
Concepto	**Coste**
Microsoft Windows 7	100.00€
Bundler	0.00€
Cygwin	0.00€
SIFT	0.00€
Meshlab	0.00€
Scanalyze	0.00€
Matlab R2008b	700.00€
Microsoft Visual Studio 2008 Express	0.00€
Microsoft Visual Studio 2010 Express	0.00€
ImageJ	0.00€
Sigpac	0.00€
Google Earth	0.00€
LEd	0.00€
MiKTeX	0.00€
OpenOffice	0.00€
Microsoft Visio 2003	330.00€
VLC	0.00€
VirtualDub	0.00€
PrintScreen	0.00€
VidShot Capturer	0.00€
Subtotal software	*1130.00*€

Tabla 8.1: Coste software

[1]Pese a haber usado más de un ordenador, el trabajo podría haberse realizado únicamente con uno, por esta razón, a efectos de costes sólo se computa éste.

[2]El coste estimado de Matlab incluye una licencia académica individual además de los siguientes paquetes (necesarios para Orto3D): Optimization Toolbox, Image Processing Toolbox.

Hardware	
Concepto	**Coste**
Ordenador portátil	900.00€
Cámara de fotos	150.00€
Subtotal hardware	*1050.00€*

Tabla 8.2: Coste hardware

Vídeos	
Concepto	**Coste**
Vídeos SIVA	0.0€
Vídeos documentales	30.00€
Subtotal vídeos	*30.00€*

Tabla 8.3: Coste vídeos

Otros conceptos	
Concepto	**Coste**
Libro de referencia [25]	70.00€
Material de oficina	50.00€
Impresión y encuadernación (memoria)	150.00€
Subtotal otros conceptos	*270.00€*

Tabla 8.4: Coste otros conceptos

Total coste material	
Concepto	**Coste**
Coste software	*1130.00€*
Coste hardware	*1050.00€*
Coste vídeos	*30.00€*
Otros conceptos	*270.00€*
Total coste material	**2480.00€**

Tabla 8.5: Coste material

Coste mano de obra			
Concepto	**Coste/hora**	**Nº horas**	**Coste total**
Análisis y desarrollo	20.00€/hora	1500 horas	30000.00€
Documentación de trabajo y resultados	10.00€/hora	250 horas	2500.00€
Total mano de obra		**1750 horas**	**32500.00€**

Tabla 8.6: Coste mano de obra

Concepto	**Coste**
Coste material	2480.00€
Coste mano de obra	32500.00€
Coste ejecución material	**34980.00€**
Honorarios y beneficio industrial (15 %)	*5247.00€*

Tabla 8.7: Coste ejecución material y beneficio industrial

Concepto	**Coste**
Coste ejecución material	34980.00€
Honorarios y Beneficio industrial	5247.00€
Subtotal	**40227.00€**
IVA(16 %)	6436.32€
Total	**46663.32€**

Tabla 8.8: Importe total del proyecto

El importe total del proyecto asciende al valor de cuarenta y seis mil seiscientos sesenta y tres euros con treinta y dos céntimos.

En Alcalá de Henares a día 16 de junio de 2010,

Fdo.: David Núñez Clemente

Ingeniero de Telecomunicación

Parte III

Apéndices

Apéndice A

Manual de Bundler

En este manual se explica cómo proceder a la instalación y uso de Bundler. Conviene no obstante, revisar el procedimiento de instalación incluido en el README.TXT de la distribución [67]. Bundler está desarrollado completamente en C/C++, bajo licencia GPL. Las versiones empleadas en este trabajo son la $v0.2$ y la $v0.3$ aunque en el momento de escribir esta memoria se ha publicado el código fuente de la versión $v0.4$. Los cambios entre la segunda, la tercera y la cuarta versión son menores y se limitan a añadir funcionalidad[1]. El proceso de instalación es sencillo, y puede realizarse de dos modos distintos. Por un lado se puede descargar el código fuente y proceder a su compilación, por otro lado, se pueden instalar los binarios disponibles en la página de la aplicación [67]. La recomendación es proceder con la segunda opción, por ser más sencilla y rápida, pero la primera es igualmente válida. Bundler puede usarse sobre Windows o Linux, incluso también sobre MacOS. Para este último, es mejor contactar con el autor para conocer el proceso de instalación ya que ha sido realizado por terceros y no está publicado en la web de Bundler.

La instalación en el presente trabajo se ha realizado completamente sobre Windows XP y Windows 7, de ahí que a continuación se explique con más detalles la instalación con estos sistemas operativos (el procedimiento es idéntico para ambos). Para ejecutar sobre Windows, hay que tener en cuenta la necesidad de instalar antes un emulador de Linux sobre Windows llamado Cygwin. Por tanto, la explicación del proceso de instalación se comenzará con la instalación de Cygwin.

A.1. Instalar Cygwin

Cygwin es un entorno emulador de Linux para Windows consistente en dos partes bien diferenciadas, una que actúa como un emulador de la API de Linux, y otra parte que contiene una colección de herramientas con la misma funcionalidad que éstas tienen en Linux. Cygwin es una herramienta de software libre, y por tanto gratuita bajo las condiciones de la licencia GPL. Las versiones de la distribución empleadas en las pruebas descritas en este trabajo son la 1.5.25 y la 1.7.5, aunque el software debiera funcionar con cualquier nueva versión. La distribución dispone de numerosos paquetes opcionales que pueden ser instalados de forma independiente en función de las necesidades del usuario. Para Bundler es necesario instalar unos paquetes concretos que más adelante se detallan.

Cygwin se instala a partir del ejecutable *setup.exe*, disponible en la web de Cygwin [14],

[1] Los cambios se describen en http://phototour.cs.washington.edu/bundler/Changelog

que descarga los paquetes desde alguno de los servidores disponibles. Durante la instalación para este trabajo, se observaron ciertos problemas de interacción entre Cygwin y algún antivirus instalado pudiendo producirse inslataciones erróneas. En concreto los problemas se encontraron con McAffee. Según se describe en la página oficial de Cygwin [14], se han experimentado algunas incompatibilidades con ciertos programas, entre ellos:

- Sonic Solutions burning software containing DLA component
- Norton/MacAffee/Symantec antivirus or antispyware
- Logitech webcam software with "Logitech process monitor"service
- Kerio, Agnitum or ZoneAlarm Personal Firewall
- Iolo System Mechanic/AntiVirus/Firewall
- LanDesk
- Windows Defender
- Embassy Trust Suite fingerprint reader software wxvault.dll
- NOD32 Antivirus
- ByteMobile laptop optimization client

Para más información sobre este asunto, conviene consultar el apartado FAQ de Cygwin, publicado en la web [15].

Si se observan dificultades con la instalación de paquetes se aconseja hacerlo desde un directorio local. Para ello es necesario ejecutar *setup.exe* y elegir la opción de sólo descarga de los archivos. A continuación, se debe ejecutar de nuevo *setup.exe* y elegir la opción de instalar desde directorio local. Esta opción primero descarga todos los paquetes deseados, y los guarda en un directorio para proceder a instalarlos en local a posteriori. En el foro se comenta la posibilidad de ejecutar Cygwin desde CD sin necesidad de instalación, sin embargo la única información al respecto corresponde a recomendaciones realizadas por algunos usuarios en el foro, y por tanto sin soporte. En general se describe como un proceso tedioso y cuyo éxito no esta garantizado por lo que se recomienda la instalación en el disco duro.

Es importante tener cuidado con la instalación desde diferentes servidores fuente (mirrors) cuando se instalan paquetes en diferentes momentos (es decir la instalación básica tomada de un servidor, a los dos días algunos paquetes de otro servidor, etc.). Existen multitud de servidores con la distribución de Cygwin, pero hay que tener en cuenta que algún servidor pudiera no tener la última actualización de algún paquete y tener versiones inconsistentes con otros servidores. Para evitar hacer instalaciones de diferentes versiones de paquetes, se recomienda realizar la instalación desde un único servidor. Si aún así se opta por usar diferentes servidores, y se aprecian constantes incompatibilidades entre ficheros, paquetes o varios errores sin conocer la causa, la mejor opción es volver a instalar Cygwin desde un único servidor.

Los paquetes indispensables para usar Bundler son:

- perl (en *interpreters*)

- zip (en *archive*)
- unzip (en *archive*)
- ImageMagick (en *graphics*)

En caso de que también se quiera compilar código C/C++ en Cygwin, es necesario incluir también los siguientes elementos:

- make
- gcc: compilador.
- librerías gcc: se instalan por defecto cuando se selecciona gcc.
- binutils: ensamblador y enlazador necesario para compilar (*as.exe*, *ld.exe*)

Una vez instalado se puede empezar a usar. La línea de órdenes es exactamente igual que en Linux por lo que no supone ningún problema trabajar con ella a personas que ya conozcan bien este entorno. En caso de que no se esté familiarizado con Linux, se recomienda buscar algún manual de usuario de Cygwin, teniendo en cuenta que el conocimiento necesario para usar Bundler sobre Cygwin es muy básico. Simplemente es necesario ir al directorio donde se encuentran las imágenes, en adelante referido como carpeta del escenario, y ejecutar el script *RunBundler.sh*. Más adelante se describirá el uso con más detalle.

A.2. Instalar Bundler

Los ficheros binarios se encuentran disponibles en la página web de Bundler [67], donde también se puede encontrar el código fuente. Se recomienda usar el binario antes que el código fuente a compilar, por ser más sencillo y rápido. Sin embargo la tarea de compilar no es muy complicada para personas familizarizadas con compiladores, siendo incluso más sencillo a partir de la versión $v0.3$ de Bundler ya que incluye ficheros de solución y proyecto de Microsoft Visual Studio 2005, pudiendo usarse cualquier versión superior. Con este compilador simplemente se necesita cargar el archivo de solución y compilar. Más adelante se comentará algo más en detalle sobre la compilación, y sobre los archivos presentes en el código fuente.

A.2.1. Instalación a partir de los binarios

Para empezar, en los siguientes párrafos se explica la instalación a partir de los binarios y también, el proceso de configuración previa a ejecutar Bundler. Esta configuración es necesaria antes de empezar a usar esta herramienta, ya que se deben colocar archivos y realizar algunos cambios en los ficheros script de la aplicación. Los pasos son los siguientes:

1. Descargar el binario de Bundler.
2. Extraer los archivos en un directorio. En los ficheros script se referirá a este directorio como $BASE\_PATH$.
3. Descargar el ejecutable para Windows de SIFT desde la página web de SIFT [35], y copiarlo en el directorio $BASE\_PATH/bin$.

4. Abrir el script $BASE\_PATH/RunBundler.sh$ y modificar la línea en la que aparece la variable $BASE\_PATH$. Sustituir esta línea por el directorio en el que se encuentra Bundler. Por ejemplo, si el directorio es $/cygdrive/c/bundler$, la variable debe dejarse como $BASE\_PATH =\text{"}/cygdrive/c/bundler\text{"}$.

5. A continuación hay que pasar el archivo de formato DOS(Windows) a formato UNIX para que pueda ser leído por las herramientas de Cygwin. Se debe a la diferencia de formato de escritura de ficheros de texto entre Windows y Unix. Para ello ejecutar en la línea de órdenes de Cygwin: $dos2unix$ –U $RunBundler.sh$

6. Se debe modificar el script $BASE\_PATH/bin/ToSift.sh$ haciendo lo mismo que en el punto anterior con $BASE\_PATH$. Además se debe modificar la variable que referencia al ejecutable SIFT, $SIFT =\$BASE\_PATH/bin/siftWin32$, con $siftWin32.exe$ el ejecutable SIFT.

7. Pasar el archivo de formato DOS a formato UNIX. Ejecutar en la línea de órdenes de Cygwin: $dos2unix$ –U $ToSift.sh$

8. Por último, en $BASE\_PATH/bin/extract\_focal.pl$ hacer exactamente lo mismo que en los casos anteriores.

Una vez realizadas estas tareas, ya se puede ejecutar Bundler. En la siguiente sección se explica cómo.

A.2.2. Instalación a partir del código fuente

La compilación a partir de la versión $v0.3$ es muy sencilla ya que se dispone de una solución Visual Studio 2005 con todas las dependencias necesarias, siendo tan sencillo como simplemente cargar la solución y compilar. Se pueden usar también versiones superiores a la 2005, siendo las versiones Visual Studio 2008 Express y la Visual Studio 2010 Express las empleada en este trabajo. Al descargar el código fuente se encontrarán una serie de carpetas con el código fuente, cuya estructura de directorios se describe en la sección A.6

A.3. Usar Bundler

Para empezar a usar Bundler una vez configurado, ir al directorio de la reconstrucción del escenario, es decir, el directorio en el que se hayan guardado las imágenes a usar. Todas las imágenes deben ser JPEG y tener extensión $*.jpg$. Tal y como está implementado el script no reconoce las extesiones $*.jpeg$ ni otras variantes). Para usar otro tipo de imágenes es necesario modificar el script. A continuación, copiar en esta carpeta el fichero $RunBundler.sh$ dado en la distribución de Bundler. Para un funcionaiento básico y primer contacto con la herramienta se recomienda usar directamente este script suministrado, que puede ser modificado para configurar otras opciones de ejecución. Por último, ya se puede usar escribiendo en la línea de órdenes de Cygwin: $./RunBundler.sh$.
El script realiza las siguientes tareas:

1. Crea una lista de imágenes usando el script $extract\_focal.pl$. Extrae la información sobre distancia focal y la guarda en un fichero. Este fichero $list.txt$ contiene una lista de los nombres de las imágenes con sus focales cuando las cabeceras EXIF de las imágenes estén disponibles o sólo con los nombres cuando no lo estén.

2. Buscar los puntos característicos de la lista de imágenes con SIFT.

3. Buscar correspondencias entre los puntos característicos de las imágenes. Es un proceso que puede ser lento. Los puntos correspondientes se guardan en un fichero llamado *matches.init.txt*.

4. Se ejecuta Bundler con el fichero options.txt creado por *./RunBundler*, que lee las correspondencias del fichero *matches.init.txt*.

Para el caso de los vídeos SIVA con caracteres sobreimpresos se ha introducido un paso adicional entre los puntos 3 y 4 consistente en un filtro, que se detalla en la memoria en 5.3.2. Este filtro es adicional a la distribución original de Bundler y por ello se ha considerado adecuado no incluirlo en la lista de pasos de la aplicación. El filtro sólo es útil si las imágenes usadas presentan caracteres sobreimpresos o líneas que pudieran ser detectadas como puntos característicos y que den lugar a reconstrucciones erróneas. La diferencia usando este filtro, es que se debe modificar el script para que sea llamado durante la ejecución justo antes de ejecutar Bundler y después de la búsqueda de correspondencias, y que se comenta al final de esta sección.

Si al llamar al script no se produce la reconstrucción sino que no lee ninguna imagen, se recomienda revisar la extensión de las imágenes así como su formato, teniendo en cuenta que el script lee imágenes en JPEG para escribirlas en PNG, formato legible por el ejecutable de SIFT. Para el ejecutable de Bundler los formatos de imagen son indiferentes, ya que únicamente trabaja con coordenadas de imagen.

En caso de que aparezca el siguiente error o similar, se debe a que no se ha realizado el paso de formato DOS a formato UNIX. Para solucionarlo, ver las líneas anteriores en esta misma sección. Pudieran encontrarse también problemas con otros elementos de la

```
$./RunBundler.sh
./RunBundler.sh: line 14: $'\r': command not found
./RunBundler.sh: line 17: $'\r': command not found
./RunBundler.sh: line 81: syntax error: unexpected end of file
```

Tabla A.1: Error de formato de archivo en Cygwin

reconstrucción tales como por ejemplo que las imágenes sean muy grandes. En concreto se han experimentado problemas con imágenes de tamaño 2272×1704 especialmente con las muy heterógeneas. Cuando el ejecutable SIFT original [35] trabaja con imágenes grandes y heterogénes detecta una gran cantidad de puntos, que también necesitan gran cantidad de memoria física, por encima de 1.5GB en alguno de los escenarios empleados en este trabajo. Esto puede generar que el sistema operativo aborte la ejecución, caso que se ha podido observar en este trabajo. Problemas similares pueden encontrarse durante el proceso de búsqueda de correspondencias, o la ejecución del ejecutable Bundler por idéntico motivo. Si el problema es en la ejecución de SIFT o el matching, se recomienda usar imágenes de menor tamaño, si el problema viene dado en la etapa de reconstrucción, se recomienda reducir el tamaño de las imágenes o dividir el total de imágenes en subconjuntos y reconstruirlos por separado. Usar un ordenador más potente si es factible es otra opción muy recomendada.

Al usar Bundler, hay que tener en cuenta que es una herramienta robusta y fiable, pero que necesita de una gran capacidad de cómputo, que incrementa conforme crece el número de puntos detectados y el número de imágenes. En problemas muy grandes como por ejemplo

aquellos presentados en la reconstrucción de Roma o Venecia con miles de imágenes [3], el programa puede tardar desde días hasta semanas con ordenadores de múltiples procesadores en paralelo. Las etapas más lentas son la búsqueda de puntos correspondientes y la reconstrucción de estructura con optimización bundle adjustment. Gracias a la ausencia de dependencias entre imágenes en las búsquedas de puntos característicos, esta etapa puede ejecutarse en paralelo, reduciendo el tiempo de cómputo considerablemente. Ocurre de manera similar para la búsqueda de correspondencias.

A.3.1. Opciones

Bundler tiene una serie de parámetros internos que permiten ejecutar diferentes opciones básicas:

- $--$ *match_table matches.init.txt* : especifica donde se guardan los matches
- $--$ *output bundle.out* : especifica el nombre de la salida final de la reconstrucción, por defecto *bundle.out*.
- $--$ *output_all bundle_* : especifica que todas las reconstrucciónes intermedias deben ser ficheros con el prefijo *bundle_*.
- $--$ *output_dir bundle* : el directorio en el cual deben escribirse todos los ficheros de salida, por defecto *bundle*.
- $--$ *variable_focal_length* : permite a Bundler optimizar la longitud focal variable para cada imagen.
- $--$*use_focal_estimate* : obliga a Bundler a usar las estimaciones de longitud focales tomadas de las etiquetas EXIF de cada imagen.
- $--$ *constrain_focal* : restringe la longitud focal de cada cámara a la estimación de la focal inicial tomada de la etiqueta EXIF. Esta opción añade penalizaciones a la función de coste de bundle adjustment.
- $--$ *constrain_focal_weight* 0,0001 : peso de la restricción de la longitud focal.
- $--$ *estimate_distortion* : indica a Bundler que estime los parametros de distorsion para cada imagen.
- $--$ *run_bundle* : ejecuta SfM

Adicionalmente a las opciones por defecto configuradas en la distribución de Bundler, se incluyen también las siguientes opciones:

- $--$ *init_pair*1 $<$ *image_idx*1 $>$
- $--$ *init_pair*2 $<$ *image_idx*2 $>$: esta opción y la anterior especifican las imágenes a usar como el par inicial, muy útil cuando el par inicial elegido automáticamente por Bundler resulta en una mala reconstrucción.
- $--$ *sift_binary* $<$ *sift* $>$: localización del binario SIFT en la instalación.

- $--add\_images < add\_list >$: dada una reconstrucción existente especificada con $--bundle$, intenta añadir a la reconstrucción las imágenes listadas en el fichero $< add\_list >$, volcando los resultados a $'bundle.added.out'$. La nueva lista de imágenes se escribe en $list.added.txt$ y usa $extract_focal.pl$ para extraer las focales y añadir las imágenes en una lista al fichero $< add\_list >$. Al usar esta opción, no incluir $--run\_bundle$ y añadir la opción de SIFT $--sift\_binary$.
- $--options\_file < options\_file >$: lee una lista de opciones del archivo especificado en la orden.
- $--help$: imprime la salida completa de opciones

A.4. Modificaciones necesarias para usar el filtro

Para usar el filtro es necesario realizar unas pequeñas modificaciones en el script *RunBundler.sh* y copiar el ejecutable del filtro en la carpeta de binarios de Bundler (el ejecutable se adjunta en el DVD ver apéndice E). A continación se detalla el procedimiento:

1. Copiar el ejecutable $filter\_v0.1$ en la carpeta $BASE\_PATH/bin/$
2. En el script *RunBundler.sh*, después de la línea
   $TO\_SIFT = BASE\_PATH/bin/ToSift.sh$
   añadir la línea
   $FILTER = BASE\_PATH/bin/filter/filter\_v0.1$
3. Después de la línea
   $rm\ –f\ options.txt$,
   añadir
   $FILTER\ matches.init.txt\ matches.filtered.txt\ 1 \gg salidaOrdenes.txt$
4. Por último, modificar la línea
   $echo\ —match\_table\ matches.init.txt \gg options.txt$
   por
   $echo\ —match\_table\ matches.filtered.txt \gg options.txt$

A.5. Analizar resultados

Para trabajar con los datos devueltos por Bundler, se debe operar con los ficheros de texto que genera con toda la información sobre la reconstrucción, ya sean los puntos característicos, las correspondencias, los parámetros de las cámaras o los puntos reconstruidos entre otros datos. En esta sección se describe brevemente el contenido de cada uno de los ficheros de resultados obtenidos.

A.5.1. Ficheros de resultados

Los resultados se guardan en los siguientes ficheros de texto:

- $bundle\backslash bundle.out$
- $bundle\backslash bundle.init.out$
- $bundle\backslash bundle_{<}n > .out$

- *bundle\points < n > .ply*
- *constraints.txt*
- *list.txt*
- *list_keys.txt*
- *list_tmp.txt*
- *prepare\list.txt*
- *matches.corresp.txt*
- *matches.init.txt*
- *matches.prune.txt*
- *matches.ransac.txt*
- *nmatches.ransac.txt*
- *nmatches.corresp.txt*
- *nmatches.prune.txt*
- *options.txt*
- *pairwise_scores.txt*
- *nombre_imagen.key.gz*

De todos estos, los más interesantes son los contenidos en la carpeta *bundle*, especialmente los ficheros *bundle.out*, *bundle_<n>.out*, con n el número de cámaras que han sido registradas en la reconstrucción para ese fichero. A continuación se explica el contenido de cada fichero.

Ficheros de reconstrucción

Son los ficheros *bundle.out*, *bundle.init.out*, *bundle_<n>.out* en la carpeta *bundle*. Todos siguen el formato mostrado en la tabla A.2. El fichero muestra los parámetros de cámara y las coordenadas de los puntos reconstruidos con referencias a los puntos correspondientes en las imágenes. Estos ficheros se crean durante la etapa de reconstrucción de estructura, y son creados por el ejecutable *Bundler.exe*. La diferencia entre ellos es el momento en el que se crean. El fichero *bundle.init.out* contiene la inicialización del proceso de reconstrucción de estructura justo después de usar Níster para obtener los parámetros del primer par. En el fichero se puede apreciar que una de las cámaras tendrá como matriz de rotación la matriz identicada y como vector de traslación un vector nulo, y que la segunda cámara estará referida a ésta. El resto de cámaras se inicializan a valores nulos, y al final del fichero se añaden todos los puntos comunes al primer par. Después de una iteración de SBA con estos datos, se genera el fichero *bundle_*001*.out*, con el par y los puntos optimizados y que sirve para inicializar la siguiente iteración de SBA. Así se continúa la ejecución como se explica en el apartado 5.1.1, y al finalizar cada iteración se vuelve a escribir un fichero *bundle_<n>.out*, donde n es el número de cámaras reconstruidas. Para facilitar su lectura, en las siguientes, se presenta la estructura que siguen estos ficheros.

La tabla A.2 muestra la estructura general. La tabla A.3 muestra la estructura de los parámetros de cámara, donde las cámaras se identifican en el orden en el que aparecen en la lista de imágenes, siendo la primera cámara la identificada con el número 0. La tabla A.4 muestra el formato de presentación de los puntos, donde cada punto viene determinado por su posición 3D y las vistas donde aparece. El formato de presentación de las vistas se muestra en la tabla A.5, donde <*camera*>es el índice de cámara empezando por el 0, <*key*>es el índice del punto característico obtenido con SIFT donde se detectó el punto en la imagen, y <*x*>y <*y*>son las posiciones de este punto en la imagen.

<num_cameras><num_points>
<camera1>
<camera2>
...
<cameraN>
<point1>
<point2>
...
<pointM>

Tabla A.2: Fichero bundle.out

<f><k1><k2>	the focal length and radial distortion coeffs
<R>	a 3x3 matrix representing the camera rotation
<t>	a 3-vector describing the camera translation

Tabla A.3: Cámaras en fichero bundle.out

<position>	a 3-vector describing the 3D position of the point
<color>	a 3-vector describing the RGB color of the point
<view_list>	a list of views the point is visible in

Tabla A.4: Puntos en fichero bundle.out

<numviews><camera><key><x><y>

Tabla A.5: Trayectorias en fichero bundle.out

Ficheros de visualización

Por otro lado se tienen los ficheros de visualización, de formato PLY. Éstos reciben el nombre de la forma *points*<*n*>*.ply*, con n el número de cámaras visualizadas, de forma análoga a los ficheros *bundle_*<*n*>*.out*. De hecho, el fichero *.ply* de valor n es la presentación de los resultados contenidos en el fichero *.out* de idéntico índice. Los PLY son un tipo de ficheros de descripción de objetos que fue diseñado como un formato apropiado para investigadores que trabajasen con modelos poligonales. Este formato fue desarrollado

en la Universidad de Stanford y su contenido puede ser texto legible por cualquier editor de texto o binario. Bundler devuelve siempre el formato texto. Éste consiste en una cabecera seguida de una lista de vértices y polígonos. La cabecera especifica los vértices y polígonos y también las propiedades asociadas con cada vértice, tales como coordenadas (x, y, z), y el color. Las caras de los polígonos son listas de índices referidas a la lista de vértices, empezando cada línea con el número de elementos que describen la cara. En la tabla A.6 se presenta un ejemplo de este tipo de ficheros.

Para poder visualizar un PLY se puede usar Meshlab [43] o Scanalyze [60] entre otras

```
ply
format ascii 1.0
element face 0
property list uchar int vertex_indices
element vertex 14604
property float x
property float y
property float z
property uchar diffuse_red
property uchar diffuse_green
property uchar diffuse_blue
end_header
6.792931e-001 -5.277387e-001 -3.538098e+000 134 130 118
-1.801587e-001 -3.709610e-001 -3.997770e+000 10 13 2
...
```

Tabla A.6: Fichero PLY

opciones. Tener en cuenta que la notación seguida para la representación de las cámaras es la siguiente: puntos rojos para las cámaras con índices pares, verdes para las cámaras con índices impares y amarillos para los puntos principales (en ambas)[2].

Fichero de ratios de paridad

Hay otros ficheros a la salida de Bundler. Uno de estos ficheros es *pairwise_scores.txt*, cuyo contenido se muestra en la tabla A.7, donde $cam[i]$ y $cam[j]$ indica el identificador de

cam[0]	cam[1]	hom[0][1]
cam[0]	cam[2]	hom[0][2]
⋮	⋮	⋮
cam[n-2]	cam[n-1]	hom[n-2][n-1]

Tabla A.7: Fichero pairwise_scores.txt

la cámara y $hom[i][j]$ indica el ratio de inliers que se ajustan a la homografía encontrada para el par de cámaras $cam[i]$,$cam[j]$.

[2]Notar que Orto3D no sigue esta notación, por lo que todas las cámaras se indican con un único punto rojo.

Fichero de opciones

El fichero de opciones *options.txt* sirve para inicializar la ejecución de Bundler. Los valores más comunes que contiene se presentan en la tabla A.8.

–match_table matches.init.txt
–output bundle.out
–output_all bundle_
–output_dir bundle
–variable_focal_length
–use_focal_estimate
–constrain_focal
–constrain_focal_weight 0.0001
–estimate_distortion
–run_bundle

Tabla A.8: Fichero opciones de ejecución de Bundler

Fichero de restricciones

Es el fichero que contiene las restricciones de cámaras y trayectorias[3]. El contenido se muestra en la tabla A.9. Las trayectorias se guardan con el formato presentado en la tabla A.10, con un formato similar al de las vistas de *bundle.out*.

num_imágenes	
num_transformaciones	
id_cámara_a	id_cámara_b
homografía(9 componentes)	
matriz_fundamental(9 componentes)	
ratio_inliers_a_b	
num_inliers_a_b	
...	
...	
num_tracks	
track[0]	
...	
track[num_tracks-1]	

Tabla A.9: Fichero de restricciones

...			
num_camaras	id_cámara_a	id_característico_a	...
...			

Tabla A.10: Trayectoria en fichero de restricciones

[3] A pesar de no haber referencia a este fichero en la documentación existente, se ha deducido su contenido a partir de la función *BaseApp*::*WriteGeometricConstraints*, en *BaseGeometry.cpp*.

Ficheros de listas

Los ficheros *list.txt*, *list_keys.txt*, *list_tmp.txt*, y *prepare\list.txt* contienen las listas de las imágenes usadas y los nombres de los ficheros de característicos. El fichero *list_tmp.txt* contiene los nombres de las imágenes. Este archivo es generado por el script *RunBundler.sh*. Las imágenes son todas las que se encuentren en la misma carpeta que el script y tengan extensión *.jpg* (aunque sean JPEG si tienen otra extensión el script no las tendrá en cuenta). En la tabla A.11 se muestra la estructura de este fichero. A partir

./nombre_imagen_1.jpg
...
./nombre_imagen_n.jpg

Tabla A.11: Fichero list_tmp.txt

de este fichero y las focales extraídas con *extract_focal.pl* se generan los ficheros *list.txt*, y *prepare\list.txt* que contiene el nombre de la imagen y la focal extraída para ésta. Este fichero se usará para inicializar la focal de cada cámara durante la etapa de SfM. El contenido se puede ver en la tabla A.12. Por último, en *list_keys.txt* se listan los ficheros de

./nombre_imagen_1.jpg	0	focal_imagen_1
...		
./nombre_imagen_n.jpg	0	focal_imagen_n

Tabla A.12: Fichero list_tmp.txt

puntos característicos para cada imagen. Tienen extensión *.key*, son ficheros de texto y para reducir su tamaño se comprimen al terminar la búsqueda de correspondencias. Por tanto estos ficheros se encuentran en la carpeta del escenario con el mismo nombre que aparece en este fichero, seguido de la extensión *.key.gz*. La estructura de éste se puede ver en A.13.

./nombre_imagen_1.key
...
./nombre_imagen_n.key

Tabla A.13: Fichero list_keys.txt

Ficheros de correspondencias

Los ficheros se nombran con el formato *matches.xxxx.txt* y contienen las correspondencias encontradas con el algoritmo de matching. En concreto son *matches.init.txt*, *matches.corresp.txt*, *matches.prune.txt*, *matches.ransac.txt*. *matches.prune.txt* contiene las correspondencias encontradas con el algoritmo ANN. Estas correspondencias se ajustan a un modelo de una forma robusta mediante el algoritmo RANSAC. Las que no cumplen el modelo, de descartan del conjunto. En la tabla A.14 se presenta el formato de estos dos ficheros, que es idéntico para ambos. Por otro lado está el fichero *matches.corresp.txt*. En ningún documento del programa se explica su funcionalidad así que aquí haré únicamente referencia al fichero. Carece de contenido. Por último está el fichero *matches.init.txt* que es el que sirve para inicializar la etapa de reconstrucción de estructura. Bundler toma de este fichero las correspondencias robustas, y las va añadiendo a la reconstrucción. La estructura de este fichero se presenta en A.15.

id_característico_a	id_característico_b	...
id_característico_a	id_característico_c	...
...		

Tabla A.14: Ficheros matches.prune.txt y matches.ransac.txt

id_camara_a	id_camara_b
num_matches_par_a_b	
id_característico_a	id_característico_b
...	
id_camara_a	id_camara_c
...	

Tabla A.15: Fichero list_keys.txt

Ficheros contadores de correspondencias

Son los ficheros *nmatches.corresp.txt*, *nmatches.prune.txt*, *nmatches.ransac.txt*. Contienen matrices con la cantidad de puntos correspondientes entre pares de cámaras. Para el caso del fichero *nmatches.corresp.txt*, el fichero contiene una primera línea con el número de cámaras del escenario, sea éste m. El resto del fichero contiene una matriz nula cuadrada de tamaño igual al número de cámaras.

Para el caso de los ficheros *nmatches.prune.txt* y *nmatches.ransac.txt*, los ficheros tienen la misma estructura anterios pero a diferencia, las líneas que siguen a la primera, contienen una matriz triangular superior con diagonal nula de tamaño $m \times m$. Las filas y columnas indican las cámaras, de forma que por ejemplo la primera fila de la matriz indica las correspondencias que tiene con el resto del escenario, siendo nulo el primer valor (consigo misma), el segundo valor serían las correspondencias con la segunda imagen, y así sucesivamente.

Fichero de puntos característicos

Para cada imagen se genera un fichero comprimido con el fichero de texto que contiene los puntos característicos encontrados con SIFT y con nombre *nombre_imagen.key.gz*. El fichero de texto es de nombre igual que el comprimido pero con extensión *.key*. En la tabla A.16 se muestra la estructura de estos ficheros.

numero_puntos tamaño_descriptor
localización_x_1 localización_y_1 orientación_1 escala_1
descriptor_128_elementos_1
...
localización_x_n localización_y_n orientación_n escala_n
descriptor_128_elementos_n

Tabla A.16: Fichero de puntos característicos SIFT

A.5.2. Usar MeshLab

Meshlab tiene una interfaz de usuario mucho más atractiva que Scanalyze y además no presenta los problemas de lentitud en renderización asociados a la otra herramienta. El inconveniente es que las versiones iniciales de Bundler (la $v0{,}1$ y $v0{,}2$) no dan ficheros PLY totalmente compatibles con MeshLab. Este inconveniente se puede solucionar editando el fichero PLY como si fuera un fichero de texto. El problema viene dado porque los ficheros de salida de Bundler carecen de caras ("*FACE*") y por tanto no es posible cargarlos sin especificar que el número de caras es nulo. Para solucionarlo, añadir después de la última propiedad (*property*) de vértice (*element vertex xxx*) las siguientes líneas

Es una tarea tediosa para muchos ficheros PLY, aunque normalmente se puede reducir a

element face 0 property list uchar int vertex_indices

Tabla A.17: Modificaciones en el fichero PLY

cambiar el fichero con mayor índice, que es el que contiene el resultado final de la reconstrucción. En la versión $v0.3$ ya se obtienen ficheros de salida totalmente compatibles con Meshlab.

Aquí no se detalla el proceso de instalación de Meshlab, que puede encontrarse en la página de la aplicación [43], donde además se puede encontrar el instalable.

A.5.3. Usar Scanalyze

Scanalyze es una herramienta de la Universidad de Stanford para visualizar archivos PLY. La visualización por defecto de Scanalyze no usa el color verdadero ("True Color"), usa un fondo negro, y renderización por polígonos. Para poder ver correctamente los puntos de Bundler es necesario activar las opciones que permiten usar el color verdadero ("True Color"), desactivar la opción "Lit"(Light) y usar renderización por puntos ("Points"), a las que se accede desde el menú:

- Render >Lit
- Render >Points
- Render >True Color

Por lo demás la herramienta es bastante sencilla de usar, y bastante intuitiva, aunque muy lenta y pesada. Si se desea conocer el proceso de instalación, acudir a la página de la aplicación [60].

A.6. Estructura de directorios del código fuente

La distribución con el código fuente se encuentra en los directorios que a continuación se detallan:

- bin : carpeta en la que se encuentran todos los ejecutables de la aplicación. Además de los usados directamente por Bundler, destacan los siguientes[4]:

[4]Para Linux se usan los mismos ejecutables pero sin la extensión *.exe*

  - *jhead.exe* : es un ejecutable para leer las etiquetas EXIF.
  - *Bundle2PMVS.exe* : sirve para convertir el fichero *bundle.out* de salida de Bundler al formato de entrada de PMVS.
  - *RadialUndistort.exe* : permite eliminar la distorsión radial en las imágenes usando los parámetros de distorsión estimados por Bundler (parámetros $k1$ y $k2$).

- examples : carpeta que contiene los ejemplos por defecto dados con la distribución de Bundler. Se recomienda usar otro conjunto de imágenes para pruebas, a ser posible imágenes de la esquina de un edificio donde se pueda reconocer perfectamente la planta del mismo una vez reconstruido.
- include : ficheros de cabecera
- lib : contiene el código fuente de todos los algoritmos y módulos usados por Bundler y que no forman parte del núcleo de la aplicación. En algunos casos han sido desarrollados por terceros y el autor los ha reunido e integrado en Bundler, y en otros casos han sido implementados por él.
- src : aquí se encuentra el código fuente del núcleo de la aplicación. Desde este código se llama al resto de funciones o algoritmos, los desarrollados por terceros y los desarrollados por el autor pero no pertenecientes al núcleo de Bundler.
- vc++ : son los ficheros de solución y proyectos para poder ser usados con Microsoft Visual Studio 2005 o superior.

Para facilitar su visualización, se incluye la figura A.1 con la estructura de la distribución.

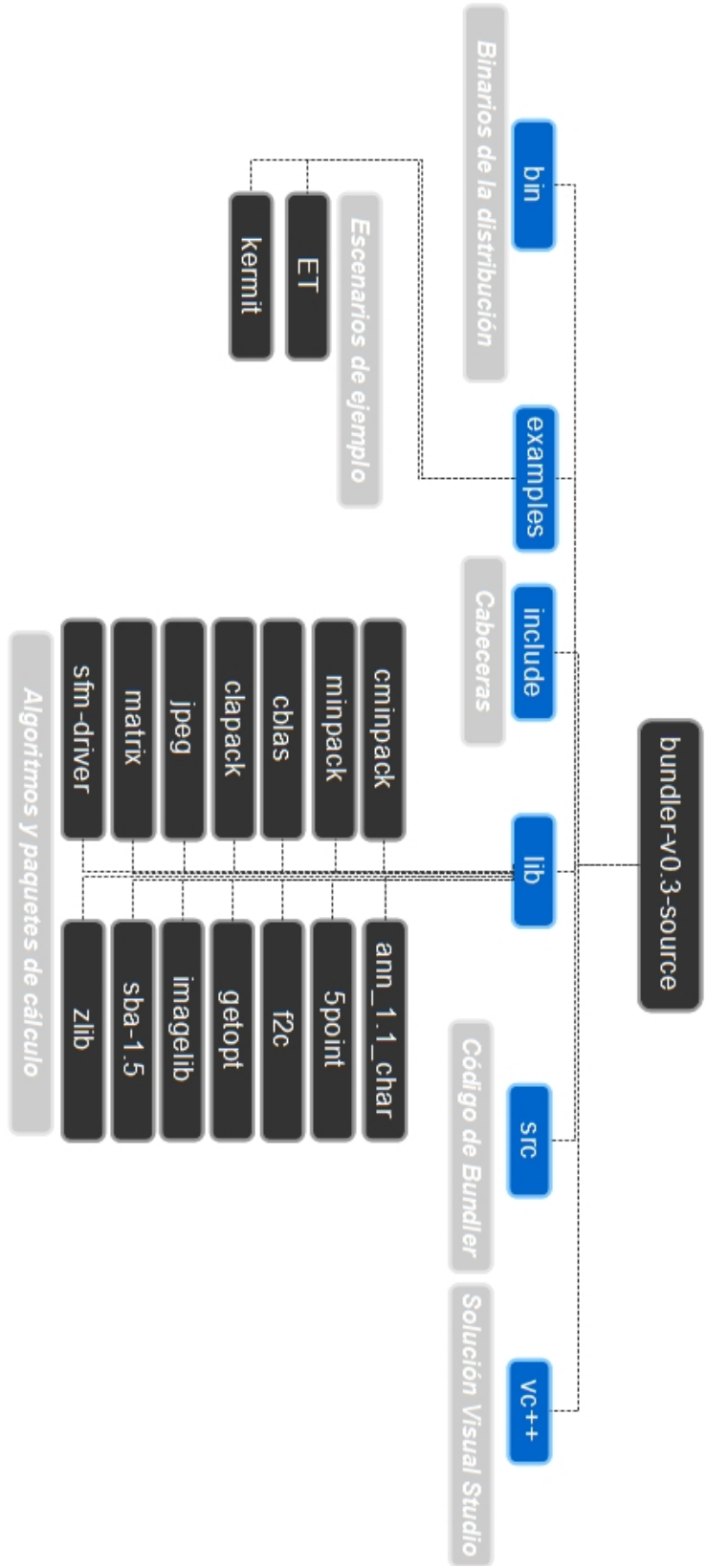

Figura A.1: Estructura del código de Bundler

Fuente: e.p.

Apéndice B

Manual de Orto3D

B.1. Introducción a la aplicación

B.1.1. ¿Qué es?

Orto3D es una aplicación completamente desarrollada en Matlab que permite evaluar los datos obtenidos por Bundler y georreferenciarlos a una ortoimagen conocida. El programa está implementado con una interfaz gráfica basada en la GUI de Matlab y realizada con el editor de interfaz gráfica GUIDE, que integra ventanas, scripts y funciones desarrollados completamente en este programa matemático. Las funciones y el código se apoyan en algunas funciones internas de Matlab, y parte del código ha sido generado a partir de funciones desarrolladas por el grupo de visión de la Universidad de Oxford, basadas en el libro [25] y por David Lowe en su distribución de SIFT [35]. El código de éstas funciones ha sido modificado para obtener la funcionalidad necesaria en la presente aplicación.

El objetivo es implementar un prototipo con una funcionalidad básica para la validación de los resultados de Bundler, que se puede ampliar o mejorar conforme se identifiquen nuevas necesidades con el uso de Bundler. Incluso podría servir de base para generar una aplicación C/C++ basada en el código Matlab del prototipo. El prototipado en Matlab es interesante desde el punto de vista la rapidez que permite frente al desarrollo en código C/C++, con la desventaja de ser menos eficiente.

La versión mostrada en este manual es la $v1.0$, sin embargo se pueden introducir características adicionales o mejoras en nuevas versiones.

B.1.2. Motivación de la aplicación

Dada la dificultad de reconocer objetos en los escenarios reconstruidos por Bundler y ante la necesidad de disponer de funcionalidad para validación, se vió imprescindible el desarrollo de una aplicación que realizase estas funciones. Los problemas causantes de la dificultad de reconocimiento son varios. Por un lado, las distancias entre objetos y cámaras del escenario introduce errores en la triangulación (ver apartado 4.7.6). Por otro, el desconocimiento de la focal de las cámaras, y por tanto de su calibración, también tiene efectos negativos. Relacionado con ambos está el efecto producido por la resolución de las imágenes. Teniendo distancias grandes y una resolución en píxeles baja, cada píxel corresponderá a una medida grande. Ésta provocará errores de indeterminación en el escenario, como por ejemplo en los vídeos del SIVA. La baja resolución de las imágenes viene

determinada por la limitación de ancho de banda entre UAV y estación terrena. Si en la nube de puntos obtenida hay un gran número de puntos erróneos, será dificil identificar los objetos y mucho más saber con certeza cuál es la planta de la reconstrucción.

Teniendo en cuenta estos problemas junto con la poca versatilidad y funcionalidad aportada por las herramientas de visualización existentes, (Scanalyze [60] y Meshlab [43]), se tomó la decisión de implementar un prototipo para visualizar los puntos en el espacio tridimensional, y principalmente para proporcionar al usuario funcionalidad para comprobar la validez de los escenarios. De las herramientas existentes, Scanalyze es excesivamente lenta para la funcionalidad requerida, y Meshlab, aunque mejor en eficiencia con respecto a la anterior, carece de funcionalidad específica para validación y comprobación de escenarios de Bundler.

La comprobación de los escenarios, mediante el chequeo de la posición de las cámaras y la nube de puntos, y la validez de los parámetros de cámara obtenidos mediante triangulación es el núcleo de la aplicación, siendo la única de este tipo aplicada a Bundler. Aunque también incluye la visualización de los puntos 3D y de las posiciones de las cámaras, la interfaz de visualización de Meshlab es superior al prototipo aquí desarrollado. Una posible línea de trabajo futura podría ser combinar la funcionalidad de Orto3D con la de Meshlab en una aplicación C/C++ gracias a que la segunda es de código abierto. Tras introducir estos antecedentes y volviendo al tema de la motivación de Orto3D, el objetivo perseguido con esta aplicación es:

- Mostrar los puntos 3D del escenario.
- Mostrar las posiciones de las cámaras (y del movimiento del vehículo si se trata de una secuencia de vídeo).
- Comprobar los puntos característicos y correspondientes en las imágenes del escenario.
- Seleccionar manualmente puntos correspondientes para formar aristas y cuadriláteros presentes en objetos visibles en el escenario en varias vistas.
- Triangular a partir de estos puntos correspondientes marcados manualmente para conocer su posición tridimensional.
- Cambiar todo el sistema de referencia tomando como eje de referencia de altura (componente Z) una arista marcada manualmente.
- Validar el escenario reconstruido con respecto a una ortoimagen.
- Validar los parámetros de cámara, en función de la posición de la cámara a partir de sus parámetros, de la dirección del eje principal y de los resultados de triangulación.
- Visualizar de forma cómoda los datos devueltos por Bundler.

Esta funcionalidad se comenzó a implementar a base de scripts y funciones, ejecutadas desde la línea de órdenes de Matlab. La primera versión de Orto3D introduce una interfaz gráfica de usuario que integra todos los scripts y funciones anteriormente llamadas desde la línea de órdenes. La idea detrás de visualizar los puntos característicos y los puntos correspondientes es evaluar la calidad de los mismos y ver si por alguna razón la reconstrucción presenta puntos erróneos que den lugar a malas reconstrucciones de estructuras.

Para algunos vídeos se obtuvieron malas reconstrucciones causadas en parte por la presencia de caracteres y artefactos sobreimpresos en los fotogramas tomados con UAV. En estos escenarios también se pudo apreciar cómo Bundler ha sido incapaz de obtener estimaciones de focal adecuadas (ver en apartado 5.3.4).

La seleccion manual de aristas y paredes se añade como una forma de identificar objetos al triangular estos puntos. Por ejemplo, triangulando las paredes de una casa se puede conocer la posicion de ésta, así como del suelo. De esta forma se puede trabajar en planta asumiendo que, salvo raras escepciones, el suelo será el plano ortogonal a la pared. Adicionalmente, la arista formada por la intersección de dos paredes (una esquina) se puede asumir perpendicular al suelo, y se puede tomar como referencia en el eje Z del sistema de coordenadas. Gracias a este cambio de sistema de referencia se puede visualizar la planta de un escenario. Esta tarea es fácil en la gran mayoría de escenarios con infraestructuras, especialmente los escenarios urbanos.

De cara a la interfaz de la aplicación, el fin buscado ha sido el desarrollo de una herramienta sencilla, cómoda para el usuario, con una interfaz gráfica, y que permita georreferenciar fácilmente los escenarios con ortoimágenes. La validación del escenario (y por ende de las cámaras) se realiza a través del marcado de puntos, el cálculo de las líneas epipolares y la triangulación de los puntos marcados manualmente. Las líneas epipolares serán coherentes con la imagen y con el punto marcado sí los parámetros de las cámaras han sido recuperados correctamente. Además, la triangulación confirma si son válidas las transformaciones realizadas a las cámaras y a sus parámetros al georreferenciar. El cambio del sistema de referencia permite ver la planta del escenario, sirviendo también como método de validación. Después de la georreferenciación y de la validación, todo este escenario se guarda en ficheros de texto que pueden ser leídos con posterioridad por esta aplicación, además en un fichero con formato PLY que puede ser visualizado con Meshlab. Adicionalmente se guarda la ortoimagen con formato PLY de forma que se puedan cargar el PLY de la reconstrucción y el de la ortoimagen en diferentes capas. Aunque orto3D incluye toda la funcionalidad necesaria para visualizar los puntos y validar los escenarios, debido a la menor eficiencia de Matlab con respecto a aplicaciones desrrolladas en C/C++, la visualización grafica del escenario es mucho mejor con Meshlab que con Orto3D, por esta razón se incluyen los PLY como una de las salidas de Orto3D.

A la fecha de redacción de esta memoria, la versión $v1.0$ incluye toda la funcionalidad básica identificada a lo largo del trabajo. En futuros trabajos podría incluirse funcionalidad adicional como operar con coordenadas geográficas (latitud,longitud)[1], obtener ficheros de salida compatibles con Google Earth, incluir funciones para obtencion de puntos de referencia en ortoimagenes, presentar gráficas de estimación del primer par de la reconstrucción o mostrar la evolución durante la optimización de los valores de focal y de los parámetros de distorsión de las cámaras.

B.2. Instalación y configuración

La instalación y configuración de Orto3D es muy sencilla, sin que suponga ni mucho tiempo, ni mucho trabajo para el usuario. Antes de proceder a configurar Orto3D es

[1]Se incluyen entradas para estos valores en la interfaz gráfica para presentarlos en futuras modificaciones del prototipo.

necesario instalar Matlab, en concreto alguna versión igual o posterior a la R2008b[2]. Puede que con versiones inferiores también funcione pero no se ha comprobado.

B.2.1. Instalación de ficheros

A continuación se explica cómo instalar los ficheros para poder usar la aplicación:

1. Crear una carpeta en el lugar deseado para la instalación, por ejemplo *orto3D*.
2. Copiar en esta carpeta el fichero *orto3Dv1_0.rar*.
3. Extraer el contenido del fichero comprimido.
4. Abrir Matlab e ir a *File>>SetPath>>AddFolder*.
5. Añadir al *PATH* de Bundler el directorio donde se han guardado los ficheros extraídos, en el ejemplo ...*orto3D**orto3Dv1_0*.
6. Añadir también al *PATH* el directorio *html* donde se han guardado los ficheros de ayuda, en el ejemplo ...*orto3D**orto3Dv1_0**html*.
7. Salvar cambios con el botón *Save* y cerrar la ventana con *Close*.
8. Ya se puede proceder a configurar los escenarios para usar la aplicación.

B.2.2. Configuración de escenarios

Para visualizar cada uno de los escenarios es necesario realizar unas configuraciones previas que se describen a continuación:

1. Ir a la carpeta de la reconstrucción
2. Crear aquí una carpeta con nombre *orto3D*
3. Guardar en esta carpeta la ortoimagen usada para georreferenciar el escenario. La imagen debe tener formato JPEG, extensión $*.jpg$[3], y tener igual tamaño de alto que de ancho (imagen cuadrada). Además el nombre de la imagen debe ser *ortoimagen.jpg*
4. Guardar en esta carpeta un fichero de texto con nombre *ortoimagen.txt* que contenga los puntos de referencia con el formato descrito en la tabla B.1. Para crear un fichero de este tipo, simplemente abrir un editor de texto y un programa de edición de imágenes como ImageJ [28]. También se puede usar la función Matlab *vntn_marcar_puntos_utm.m*, contenida en la carpeta de código de Orto3D. El programa o la función, son necesarios para conocer las coordenadas de los puntos que se desean tomar como referencias UTM. A continuación es necesario conocer las coordenadas para ese punto en la ortoimagen, por ejemplo, con SIGPAC [45] es un procedimiento sencillo. Colocando el puntero encima de la imagen se puede conocer la coordenada. En la versión 6.3.0 de SIGPAC, se pueden usar los datum WGS84, ETRS89, ED50. En versiones anteriores sólo estaba disponible ED50. Si la ortoimagen corresponde a la zona de Madrid, el huso UTM de ésta será el 30, estando la

[2]Versión usada para desarrollar el prototipo.

[3]Extensiones como $*.jpeg$, $*.JPG$ o $*.JPEG$ no son reconocidas por Orto3D.

Península Ibérica en los husos 29, 30, 31 y las Islas Canarias en el 27 y 28. Como dato adicional para futuras versiones, se puede incluir la medida de latitud (N o S) y longitud (W o E) en los ficheros (de hecho en los ejemplos se incluyen), pero la versión actual (1.0) no usa éstos. Todos estos datos se añaden al fichero de texto, dejando una línea inicial con el número de puntos de referencia usados. El número de puntos de referencia debe ser como mínimo dos. Si se usan más, entonces la relación metros/píxel tomada es una media de la calculada con todos los pares de puntos de referencia disponibles. En las tablas B.1 y B.2se puede ver la estructura del fichero y un ejemplo numérico.

5. Con esto ya se puede empezar a usar la aplicación, para ello poner como directorio activo la carpeta *bundle* de la reconstrucción de alguno de los escenarios, y escribir en la línea de órdenes *orto*3*Dinicio*.

npuntos								
*x*1_*img*	*y*1_*img*	*x*1_*utm*	*y*1_*utm*	*huso_utm*	lat_1	*N*/*S*	$long_1$	*W*/*E*
...								
xn_img	*yn_img*	*xn_utm*	*yn_utm*	*huso_utm*	lat_n	*N*/*S*	$long_n$	*W*/*E*

Tabla B.1: Puntos UTM de referencia

3								
384.95	219.01	472208.49	4489627.04	30	40.556241667	N	3.328244444	W
616.70	300.21	472572.44	4489547.32	30	40.555533333	N	3.323941667	W
369.50	581.74	472193.17	4489264.33	30	40.552972222	N	3.328408333	W

Tabla B.2: Ejemplo de puntos UTM de referencia

B.2.3. Solución de errores

- Si al poner en la línea de órdenes *orto*3*Dinicio* indica que el directorio no es válido, o muestra un mensaje de error avisando que no encuentra algún fichero: lo más probable es que sea debido a que el directorio activo no es la carpeta *bundle* del directorio de reconstrucción. Para corregirlo, revisar el directorio activo.
- Si el problema anterior persiste pese a revisar el directorio activo, comprobar que la ruta de orto3D añadida al PATH de Matlab es la correcta. Para más información sobre este procedimiento consultar B.2.1.
- Si las imágenes de las ventanas no se cargan correctamente puede deberse a que ha perdido la referencia a la ventana. Si esto ocurre, cerrar las ventanas de orto3D y empezar la georreferenciación de nuevo. En lo sucesivo, esperar a que las ventanas con imágenes se carguen antes de cambiar de ventana activa o realizar alguna operación con Matlab. Hacer lo contrario hace que la ventana activa no sea la correcta y que por tanto no se carguen las imágenes como debieran.
- Si al llegar a la sobreimpresión del escenario sobre la ortoimagen Orto3D no puede cargar la ortoimagen, comprobar que ésta se encuentra con el nombre y formato especificado en la carpeta orto3D dentro del escenario. Para más información consultar B.2.2.

- Si al llegar a la georrefenciación, Orto3D no puede cargar el fichero con los puntos, comprobar que en la carpeta *orto3D* se encuentra el fichero *ortoimagen.txt* con los puntos de referencia UTM. Para más información consultar B.2.2.

B.3. Funcionalidad implementada

Orto3D incluye las siguientes funciones:

- Marcado manual de puntos: permite marcar puntos adicionales en las imágenes para luego reconstruirlos. Estos puntos son marcados manualmente por el usuario en tres imágenes diferentes.
- Representación de las líneas epipolares como apoyo al marcado: permite marcar los puntos en las tres imágenes con la ayuda de las líneas epipolares. Las líneas epipolares indican la línea sobre la que debiera estar un punto en la imagen, tomando como referencia la imagen 1. Además de ayudar al marcado de los puntos, permiten validar la estimación de los parametros de las cámaras.
- Selección de eje Z: permite cambiar todo el sistema de referencia a partir de una arista definida por dos puntos marcados manualmente por el usuario. El eje Z estaría en la vertical con coordenadas $X = 0$, $Y = 0$.
- Selección de cuadriláteros (poligonales): añade un cuadrilátero a la reconstrucción a partir de cuatro puntos marcados por el usuario.
- Selección de aristas: similar al marcado del eje Z pero con la diferencia de que en este caso no se altera el sistema de coordenadas inicial.
- Selección de punto XY: selecciona un punto perteneciente al plano XY del sistema para que sirva como origen de coordenadas de altura. Aunque esta funcion es muy útil a la hora de asignar alturas al escenario, hay que tener en cuenta que todas las alturas serán relativas al punto marcado en XY, (coordenadas $(X_xy, Y_xy, 0)$), para obtener alturas absolutas con respecto al nivel del mar se necesita más información sobre el escenario, por ejemplo la altura sobre el nivel del mar de un punto concreto del escenario. Esta funcionalidad no se incluye en la presente versión aunque pudiera ser interesante para futuras actualizaciones.
- Triangulación: aplica la triangulación a los puntos 2D seleccionados. Para ello, primero calcula el punto 3D a partir de un método de triangulación lineal, luego aplica un método no lineal a partir del valor calculado linealmente.
- Visualización de la nube de puntos en planta: una vez elegida la arista de referencia del eje Z se puede modificar todo el escenario para asignar este eje como referencia. A partir de aquel se puede visualizar todo el escenario en planta. Un eje se podría obtener a partir de la arista de un edificio, teniendo en cuenta que salvo casos muy contados, los edificios se construyen con paredes perpendiculares al suelo.
- Georreferenciación: georreferencia aproximadamente el escenario de Bundler a una ortoimagen. Para ello es necesario conocer cierta información adicional sobre el escenario, no sus coordenadas UTM sino una localizacion aproximada del mismo, siendo una tarea más de fotointerpretación que de geolocalización. Un ejemplo aplicado es el caso de la Torre Picasso(Madrid) en uno de los escenarios de ejemplo en la sección

5.4.3. El edificio es perfectamente conocido en Madrid por ser durante cierto tiempo el edificio mas alto de la ciudad. Además en la imagen se ven otros edificios de la zona tales como el Santiago Bernabeu, Torre Europa o la zona comercial Azca. Conociendo estos datos se puede acudir a una herramienta de visualizacion de ortoimagenes como SIGPAC [45] o Google Earth [24] y se puede tomar una ortoimagen de esta zona donde se vean los edificios reconocidos. Las coordenadas se obtendran a partir de una serie de puntos de referencia obtenidos en las ortimagenes, por ejemplo tres o cuatro que permitan obtener una relación entre las medida en metros del escenario real y la medida en píxeles en la imagen (relación metros/píxel) que luego será usada para asignar las coordenadas a todo el escenario.

- Guarda de puntos y cámaras en coordenadas UTM: guarda en el fichero *orto3D.out* todos los datos obtenidos de *bundle.out* y convertidos a coordenadas en metros relativas a un punto de referencia UTM. Se evita guardar las coordenadas UTM absolutas al ser éstas de valor muy grande y poder introducir errores en las operaciones con las matrices de las cámaras. En vez de ésto, se usa un punto de referencia conocido, siendo éste el punto definido por el píxel inferior izquierdo. Este punto es el más cercano al punto de referencia del huso (esquina inferior izquierda del huso UTM).

- Guarda de escenario y ortoimagen en formato PLY: guarda el escenario PLY con las coordenadas en metros y la ortoimagen en ficheros separados. Estos se pueden cargar en Meshlab y ser visualizarlados en diferentes capas.

- Representación de las cámaras en el escenario 3D y de los ejes principales: representa las posiciones de las cámaras en la interfaz de la aplicación así como el eje principal que da al usuario una idea de la dirección de vista de la cámara. Esta característica es muy útil para validar la estimación de las cámaras ya que permite intuir *hacia donde miran* por la imagen que proyectan del escenario. Los datos de salida incluyen las posiciones de las cámaras.

- Sobreimpresión de puntos correspondientes sobre la imagen: para validar la correción de los puntos, éstos se visualizan sobreimpresos en las imágenes usadas para la reconstrucción. Si los puntos correspondientes presentados son erróneos de puede descartar el escenario.

- Marcado de puntos con ayuda de la correlación: el marcado se hace con tres imágenes, además es opcional el uso de la correlación para afinar en mayor medida la selección de los puntos. Ésta debe ser precisa para que la triangulación dé buenos resultados, en caso contrario fallará por el error cometido en el marcado. Dependiendo del escenario, un error de marcado de uno o dos píxeles puede suponer un buen número de metros en la reconstruccion. Para reducir este efecto se recomienda usar una línea base bien separada (ver apartado 4.7.6) y adicionalmente, la ayuda de la correlación. La correlación se aplica a las tres imágenes en pares de dos, la primera con la segunda y la primera con la tercera, refinando la posición del punto marcado con respecto a una ventana alrededor de éste.

- Interfaces de diálogo con avisos al usuario: avisa al usuario de errores en la georreferenciacion y otra información referente al manejo de la aplicación, así como muestra los resultados de triangulación y georreferenciación.

- Marcado en la ortoimagen de puntos UTM de referencia para luego usarlos como referencias en la georreferenciación: los puntos de referencia se tomarán en un paso previo al uso de orto3D (funcionamiento por defecto), pero también se pueden añadir puntos de referencia sobre la marcha. Para ello es necesario conocer las coordenadas de puntos presentes en la ortoimagen usada en el escenario. Se marcan estos puntos y se añaden las referencias. Como mínimo deben ser dos puntos para obtener una relacion correcta metros/píxeles. Por ser este procedimiento tedioso, se recomienda hacer esta tarea como paso previo a la georreferenciación para generar un fichero que contenga estas coordenadas en píxel y metros, y evitando tener que marcar las referencias manualmente.

- Lectura de fichero de puntos de referencia ya guardados: una vez georreferenciado el escenario se puede cargar para continuar trabajando con él, ya sea evaluando los datos de focal, modificando la información del escenario o incluso añadiendo nuevas aristas o cuadriláteros.

- Presentación de resultados en 3D: los puntos y las cámaras se presentan en la interfaz gráfica. Se prensentan sin el color asociado para reducir la carga computacional en Matlab. Para ver los puntos con sus colores verdaderos es mejor usar Meshlab [43].

- Zoom para ayudar en el marcado de puntos: para marcar los puntos de forma más precisa e inequívoca.

- Obtención de un fichero de datos de la reconstrucción: en este fichero se guarda información relativa a la reconstrucción, como localización UTM, nombre del escenario, número de puntos, número de cámaras, número de cámaras válidas, porcentaje de cámaras válidas, descripción del escenario, y otros datos relacionados.

- Marcado de puntos en metros y triangulación: adicionalmente al marcado de puntos con coordenadas Bundler se incluye el marcado de puntos con coordenadas métricas. Las coordenadas métricas son relativas a un punto de referencia UTM, para evitar el error introducido al usar matrices con valores muy altos (del rango del millón de metros). Aunque se empleen valores relativos a un punto de referencia, a efectos prácticos es lo mismo, siendo ventajos desde el punto de vista del cálculo. Si el marcado de puntos, proporciona las epipolares y la triangulación correctas, se puede decir que la georreferenciacion es válida para el escenario reconstruido. Puede entenderse así como otro método adicional de validación de resultados.

- Gráficas de la focal: se puede ver información sobre las focales de las cámaras y ver la distribución de sus valores en una gráfica. Si se presuponen que las cámaras son idénticas y no tienen zoom (esto se asume en los vídeos empleados en las prueabs), todas las focales estimadas debieran estar en un rango de valores muy cercano. Valores muy dispares indican claramente un error en la estimación de éstas. Además de ver una gráfica de las focales, se puede ver también un histograma de la distribución de éstas.

- Gráficas de la distribucion de puntos en las imágenes: distribución de puntos característicos por imagen, puntos correspondientes y número de cámaras en cada trayectoria. Toda esta información adicional ayuda a conocer más datos sobre el escenario y la reconstrucción.

En la siguiente sección se presenta en detalle cómo se usa esta funcionalidad para georreferenciar un escenario, y para cargar los resultados una vez georreferenciados.

B.4. Georreferenciar escenario

La funcionalidad de georreferenciar el escenario permite referenciar la nube de puntos y las cámaras obtenidas con la reconstrucción a una ortoimagen de la que se conocen las coordenadas métricas UTM. Debido a que los puntos y las camaras se obtienen en un escenario con escala arbitraria, es necesario usar algún método que permita pasar de este marco de coordenadas a otro con escala métrica. Para implementarlo se ha seguido parcialmente la idea propuesta por el autor de Bundler en [69]. El método en concreto se basa en ajustar de manera interactiva la nube de puntos con la ortoimagen. El ajuste se apoya en la identificación de puntos o características visibles y bien reconocibles tanto en el escenario como en la ortoimagen. Esta tarea es sencilla en el caso de edificios o estructuras, especialmente en escenas urbanas, ya que las plantas de éstos serán fácilmente visibles tanto en la reconstrucción como en la ortoimagen. Por esta razón el primer paso requerido para la georreferenciación es elegir un sistema de coordenadas para la reconstrucción que permita conocer sin discrepancia cuál es la planta del edificio. Este sistema de coordenadas es elegido por el usuario en base a un par de puntos seleccionados. Como se conocen los parámetros de las cámaras, la selección manual de puntos correspondientes en el escenario presentes en varias imágenes debe dar como resultado un punto en el escenario 3D. Si ahora este par de puntos marcados forman una recta perpendicular al suelo, se tiene el eje Z de la reconstrucción. Ésto es fácil porque al tratarse de escenarios con edificios y otras estructuras, las paredes de éstos serán prácticamente en su totalidad perpendiculares al suelo. De estas paredes se pueden tomar aristas como eje Z del sistema de referencia. Adicionalmente es necesario disponer de alguna poligonal que defina alguna pared o tejado (u otros elementos identificables en la imagen) para tener un método de validación adicional. Por último es necesario marcar un punto en el suelo que permita tener una referencia de altura en el sistema y cuya tercera componente se tomará como origen de coordenadas en los valores del eje Z. Es decir, se debe aplicar una transformación al escenario tal que la tercera componente de este punto cumpla $Z = 0$ y además que las coordenadas en X e Y no sean modificadas. La transformación es entonces una traslación en la componente Z de valor opuesto a la tercera componente del punto.

Una vez establecido este marco de referencia se puede visualizar todo el escenario en planta de manera inequívoca. En esta vista es fácil identificar las plantas de edificios u otras estructuras tanto en la nube de puntos como en la ortoimagen. Mediante el marcado de dos puntos de georreferenciación en la ortoimagen y otros dos en la planta de la nube se puede definir una recta correspondiente entre imágenes. Esta recta en ambas imágenes describe el cambio de escala, la rotación y la traslación que es necesario aplicar a la reconstrucción 3D para que la planta se ajuste a la ortoimagen, pero siempre trabajando únicamente en dos dimensiones. Trabajar con dos dimensiones permite simplificar la tarea de georreferenciación. Para ello, durante el marcado de las rectas de referencia se asume que la tercera componente de los punto tridimensionales que la definen es $Z = 0$. Es decir, que esta recta se encuentra en el plano suelo (plano XY). Tras esta operación se sobreimprime una imagen sobre la otra, lo que se muestra en la figura B.10. Ahora es necesario cargar una coordenadas métricas de referencia para poder obtener la relación metros/píxel de la ortoimagen que a su vez permitirá pasar el escenario de las coordenadas arbitrarias de Bundler a un escenario con coordenadas métricas. Esta es la última etapa de la georreferenciación aquí propuesta, que consiste en aplicar una transformación de escala a todos los puntos y cámaras para pasarlos al marco de coordenadas métricas. Las opciones que siguen son guardar los datos e incluso marcar más puntos para completar el escenario. Los

puntos adicionales sirven también para poder validar los resultados obtenidos, ya que las epipolares y la triangulación indicarán la validez de los cambios de sistema de coordenadas aplicados.

De cara al usuario, el proceso de georreferenciación es muy sencillo. Éste se ha implementado con una serie de ventanas que permiten al usuario realizar cada uno de los pasos requeridos para la georeferenciación. En primer lugar se debe abrir el entorno Matlab y comprobar que se ha configurado concretamente el directorio dónde se encuentra la aplicación orto3D (ver la sección B.4). Luego se debe revisar que el directorio activo de Matlab es el que contiene la carpeta *bundle* del escenario, donde se almacenan los resultados de salida de Bundler. Una vez comprobado esto se puede introducir en la línea de órdenes la llamada mostrada en la tabla B.3, que limpia la línea de órdenes, limpia las variables

```
≫
≫ orto3Dinicio
≫
```

Tabla B.3: Llamada de inicio a Orto3D

existentes en el entorno creadas por ejecuciones previas y cierra todas las figuras abiertas(no las ventanas), abre la ventana de la figura B.1 y presenta por la línea de órdenes la información mostrada en la tabla B.4. La ventana abierta permite varias opciones, de las

```
Orto3D Aplicación para visualizar reconstrucciones 3D en ortoimágenes

David Núñez Clemente
david.nunez.clemente (at) gmail.com

Marzo 2010
Versión v1.0
```

Tabla B.4: Mensaje de inicio a Orto3D

cuáles únicamente se explicará en esta sección el uso de la primera. El resto de opciones se explican en las secciones B.5 y B.6. Al pulsar en el botón georreferenciar escenario, se inicia el proceso de georreferenciación que comienza cargando los datos de salida de Bundler. El proceso de carga puede durar de unos pocos segundos a algunos minutos, lo que depende del peso de los datos del escenario (es decir del número de cámaras y puntos que tenga la reconstrucción) y del ordenador empleado[4]. La siguiente ventana cargada es la ventana de selección de cámaras para la triangulación. Como se ha comentado anteriormente, se emplean tres de los fotogramas de la secuencia de vídeo empleada para la reconstrucción. Estos tres son seleccionados manualmente por el usuario con la condición de que la línea base de las imágenes sea suficientemente grande con respecto al tamaño del escenario y que el solape entre las imágenes sea suficiente como para poder marcar puntos correspondientes. La ventana se muestra en la figura B.2. Después de seleccionar

[4]Para el escenario TORRE_PICASSO_AZCA_MOD_1 con 54 cámaras y 14496 puntos un ordenador con procesador Intel Centrino 1.50GHz tarda unos 10 minutos, mientras que otro con procesador Intel Core i7 1.60GHz tarda unos 2 minutos y medio.

Figura B.1: Ventana Orto3Dinicio

Ventana de inicio de la aplicación Orto3D. Fuente: e.p.

la imagen deseada, es necesario pulsar el botón *fijar cámara* para que el índice de ésta se guarde para las siguientes partes de la georreferenciación. Cuando se han seleccionado todas las imágenes, se pulsa el botón *aceptar*, que continúa con la siguiente ventana. En esta ventana se muestra información adicional: el nombre de la imagen, la longitud focal estimada, y el número de puntos correspondientes para esa imagen. La siguiente ventana es la ventana de inicio de la georreferenciación B.3. Ésta muestra por un lado el escenario 3D con cámaras y puntos. La posición de las cámaras se representa con un triángulo rojo posicionado en el centro de la cámara y un número sobre éste que indica el índice de cámara dentro de la reconstrucción. Todas las cámaras tienen un índice que comienza con el valor 1 en la representación[5]. También se representa el eje principal que une el centro de la cámara con el punto principal (sección 4.3). La razón de representar éste es que permite ver cómodamente hacia donde mira la cámara reconstruida, pudiendo detectar posibles anomalías a la hora de validar. Por ejemplo, si la cámara enfocase hacia el lado opuesto donde no hay puntos indicaría que Bundler ha cometido un error con esta cámara. Esta ventana dispone de varias opciones, de las cuales tres consisten en el marcado de puntos manuales, otras tres corresponden a cambios en la visualización y las dos últimas permiten ver la planta de la reconstrucción y continuar con la siguiente ventana. Al cargar esta ventana también se carga un aviso, que se presenta en la figura B.4. Éste era importante en las primeras versiones de la aplicación, ya que se había implementado para seguir el orden de marcado siguiente:

1. Marcado de eje Z.
2. Marcado de poligonales.
3. Marcado de punto en el suelo (plano XY).

Sin embargo en la versión definitiva del prototipo se ha modificado para que el marcado se pueda hacer en cualquier orden con el mismo resultado final, aunque se recomienda seguir éste por ser el más comodo para trabajar. Para seguir este orden se explicará primero el marcado manual de puntos para seleccionar el eje Z. Como método adicional para mejorar los resultados del marcado de puntos se añade una función de correlación. Ésta se puede activar opcionalmente para reducir el eventual error introducido por el marcado manual, especialmente si la resolución en píxeles de la imagen es pequeña. Cuando se use

[5]Equivalente al valor 0 en el formato de salida de Bundler, ya que si n es el número de cámaras, para Orto3D (basado en Matlab) los identificadores van de la 1 a n, mientras que para Bundler (escrito en C/C++) las mismas cámaras se numeran de 0 a $n-1$.

la correlación hay que tener en cuenta que se pueden presentar problemas si las imágenes presentan rotaciones, por lo que no debe usarse en todos los escenarios. Para activar la función, marcar la pestaña en la esquina inferior derecha de la ventana de inicio de georreferenciación (figura B.4). Al pulsar el botón de marcado de puntos se cargan tres ventanas, cada una con una imagen de la reconstrucción que corresponde a la imagen de la ventana seleccionada en la ventana B.2. Como se podrá ver en las figuras B.6, B.7, además de presentar la imagen, también se presentan datos como la focal, el nombre de la imagen y la cantidad de puntos correspondientes. Cuando el marcado con correlación está desactivado, aparecerá el mensaje por pantalla que se reproduce en la figura B.5.

El proceso de marcado es sencillo y usa únicamente tres cámaras, que de aquí en adelante se denominan cámara 1, cámara 2 y cámara 3. También se puede hacer uso del zoom o del desplazamiento para poder marcar mejor los puntos y en todos los casos se usa la ayuda de la geometría epipolar. Para la cámara 2 se representa la línea epipolar con respecto al punto marcado en la cámara 1 y para la cámara 3 se representan las epipolares de los puntos marcados en las cámaras 1 y 2. Para que las epipolares se calculen de manera correcta es obligatorio seguir el orden de marcado en las cámaras: primero en la cámara 1, luego en la cámara 2 y por último en la cámara 3. En caso contrario la aplicación devolverá un error.

Para iniciar el marcado en cada imagen se pulsa el botón *seleccionar punto* de su ventana respectiva. Cuando se han seleccionado todos los puntos en todas las imágenes se pulsa el botón *finalizar eje Z*, que únicamente se encuentra en la ventana de la cámara 1. Al pulsarlo se guardan los puntos seleccionados, se triangulan, se añaden al escenario, y se calculan las transformaciones necesarias (traslaciones y rotaciones) para que esta arista esté contenida en el eje Z (de ahí que las coordenadas de la arista sean $X = 0$ e $Y = 0$ después de aplicar la transformación). Debido a que no se usan restricciones adicionales, hay dos ángulos de rotacion diferenciados 180^{o} que dan el mismo resultado al usar la arista como referencia. Por ello se añade funcionalidad adicional para corregir esta indeterminación. Estas funciones son *Girar 180º en XY* y *Girar 180º en XZ*. Si el resultado devuelto por Bundler no es el acertado porque se encuentra rotado 180^{o}, estas funciones corrigen este problema. Tras el marcado, es necesario pulsar el botón *Actualizar Vista 3D* para que se vuelva a cargar el escenario con las modificaciones realizadas.

El proceso de marcado de poligonales y del punto en el plano XY es similar, con la única particularidad de que el número de puntos seleccionados es diferente (cuatro por imagen para los cuadriláteros y un punto por imagen para el punto en XY). En realidad el marcado de poligonales permite seleccionar únicamente cuadriláteros, por ser el tipo de poligonal más sencilla y útil en la selección de paredes o tejados del escenario. Éstos permiten reconocer de forma fiable objetos presentes en el escenario tales como edificios, especialmente cuando la nube de puntos es ambigua o no permite un reconocimiento visual directo. En la imagen B.7 se presenta un ejemplo de marcado de poligonal sobre un lateral del edificio. Como se puede ver, se dibuja la recta o la poligonal seleccionada según se van marcando los puntos. En rojo se presentan las líneas epipolares y en magenta las rectas que unen los puntos marcados. En los escenarios se carecerá de origen de coordenadas en altura siendo este valor arbitrario. Para evitar esto, el punto XY tiene como única funcionalidad servir de referencia de altura Z, siendo éste un punto en el plano suelo. Para ello se asume que el plano suelo esta en el plano XY y que por tanto tiene altura $Z = 0$. En este prototipo no se emplea la altura sobre el nivel del mar, asumiendo siempre el plano suelo como origen de coordenadas. Por tanto, al emplear un punto

en el suelo, los valores georreferenciados por la herramienta son relativos a la altura del escenaro sobre el nivel del mar. Como herramienta adicional se puede ver la planta en otra ventana diferente pulsando en el botón *ver planta* (figura B.8), aunque también se puede hacer lo mismo con la herramienta de giro de Matlab disponible en la barra de herramientas de la ventana de inicio de georreferenciación. Seleccionando esta herramienta y haciendo click en el segundo botón del ratón se podrá elegir el plano que se quiere ver.

Los puntos marcados se representan en la visualización 3D del prototipo con color azul si se trata del eje Z, con color verde si se trata de una poligonal o con un círculo verde. Como se puede ver, el color rojo se reserva a la cámara.

Una vez realizado todo el procedimiento de marcado de puntos de apoyo para la validación del escenario y la visualización de la planta, se puede proceder a hacer la georreferenciación. En primer lugar, se debe pulsar en el botón *Siguiente* de la ventana de inicio de la georreferenciación. Este botón abrirá la ventana $vntn\_marcar\_rectas\_referencia$ presentada en la figura B.9. Esta ventana presenta tres imágenes, por un lado, la planta de la reconstrucción, por otro, la ortoimagen y en la parte inferior un fotograma de la secuencia usada para la reconstrucción. Ésta última puede cambiarse por otras imágenes del conjunto pulsando al botón *Cambiar imagen*. Para visualizar cómo se distribuyen los puntos correspondientes en las imágenes se puede activar la pestaña de sobreimpresión que carga los puntos correspondientes de cada imagen y los sobreimprime en ella con un color amarillo.

Adicionalmente se presentan los botones de *Zoom in/out* para acercar las imagenes y el botón de *Cerrar* para cancelar la georreferenciación y cerrar todas las ventanas abiertas. La georreferención se realiza pulsando el botón *Marcar rectas* que permite seleccionar una recta en la planta de la reconstrucción y esa misma recta en la ortoimagen. Se recomienda que las rectas seleccionadas sean claramente visibles en las plantas del escenario, siendo las más factibles las correspondientes a paredes o estructuras. Una vez seleccionadas las rectas, se puede pulsar en *Sobreimpresión* que tomará todo el escenario y lo modificará aplicándole una escala, rotacion y traslacion definida por las rectas marcadas y que permitirá sobreimprimir la escena 3D sobre la ortoimagen. La sobreimpresión se presentará en la ventana *georreferenciar* presentada en la figura B.10.

Una vez obtenida la sobreimpresión, se puede proceder a ajustar las coordenadas conocidas de la ortoimagen con las coordenadas de la reconstrucción. La ortoimagen tendrá una relación de metros/píxel bien definida por el número de píxeles que tiene la imagen y los metros a los que corresponde. Conocida la relación entre el escenario reconstruido y la ortoimagen y además conocida la relacion entre la ortoimagen y las coordenadas métricas, es sencillo pasar de coordenadas del escenario a coordenadas en metros.

Para trabajar con las coordenadas de la ortoimagen se trabaja con puntos de referencia que han sido seleccionados en la ortoimagen de manera externa a la aplicación Orto3D. Para ello es necesario elegir un programa de visualización de ortoimagenes como SIGPAC, por disponer de imágenes de buena calidad de todo el territorio español (incluyendo zonas rurales). Este programa permite conocer las coordenadas, métricas de puntos seleccionados en la ortoimagen con el sistema UTM ED50 o también con WGS84[6]. Si se toman dos puntos con sus coordenadas UTM (es el mínimo indispensable para seguir este procedimiento)

[6]Disponible en las últimas versiones.

se puede conocer la relación metros/píxeles siempre y cuando se conozcan las coordenadas de estos puntos en píxeles. Hay dos procedimientos para introducir estos puntos de referencia, por un lado está el método manual que introduce los puntos de referencia uno a uno mediante la ventana presentada en la figura B.11. Esta se activa al pulsar el botón *Marcar Puntos UTM*. Por otro lado está el método automático, que se activa al pulsar el botón *Cargar puntos UTM*.

El primer método es más tedioso y es el método implementado inicialmente. Para facilitar la interacción con el usuario se introdujo el segundo método que necesita un fichero con las coordenadas en píxeles y en metros de los puntos de referencia. Para más información sobre este fichero, ver el apartado B.2.2, donde se comenta el formato del fichero en la tabla B.1 y se presenta un ejemplo en la tabla B.2.

Una ver cargados los puntos de referencia UTM, se puede georreferenciar el escenario completo dando al botón *georreferenciar* que simplemente pasa a coordenadas métricas todos los puntos así como las cámaras. También se georreferencian los puntos marcados manualmente. Al finalizar el cambio de coordenadas se presentará una imagen de *debug* con el escenario en metros. Las coordenadas se dan con valores relativos a una coordenada UTM. Esto se hace para evitar la operación con números del orden del millón, que producen errores a la hora de trabajar con las matrices de las cámaras en este rango. Todo el escenario métrico se referencia a la esquina inferior izquierda de la imagen, que se considera el centro de coordenadas relativo del escenario[7].

Antes de guardar todo el escenario, es aconsejable añadir una descripción y una localización que se guardarán en el fichero de información del escenario. La descripción puede tratar de los objetos visualizados en el escenario, mientras que la localización puede indicar la región, ciudad donde se ha realizado el vuelo. Esto permitirá recorrer diferentes escenarios de una forma sencilla en base a la informacion de localización añadida. El proceso es muy sencillo, basta con añadir texto en los campos de la ventana *vntn_anadir_descripcion* mostrada en la figura B.12 y pulsar *aceptar*.

El siguiente paso es *Guardar todo*, que almacenará el escenario georreferenciado en un fichero con nombre *orto3D.out*. El formato de este fichero es idéntico a *bundle.out*. También generará los ficheros de visualización del escenario *orto3D.ply* y *orto3D_plano.ply*. El primero contiene el escenario en metros (tanto puntos como cámaras) y el segundo la ortoimagen en formato PLY. Adicionalmente, se obtienen los ficheros $orto3DInfo.txt$, que contienen información sobre el escenario, con coordenadas, descripción, número de puntos, etc. y *orto3Dmanual.txt* con los puntos marcados manualmente.

La ventana de georreferenciar (figura B.10) también permite seleccionar más aristas o cuadriláteros para la reconstrucción. Esta funcionalidad no solo sirve para añadir puntos, sino que también permite validar las transformaciones de las cámaras. Si las transformaciones son correctas, tanto las líneas epipolares como los puntos triangulados se emplazarán en un sitio correcto. Si no fuera así habría un error en la transformación de la camara. En cualquier caso es posible que la triangulación introduzca un error, especialmente si se

[7]En las ortoimágenes UTM empleadas, el origen de coordenadas está en el extremo inferior izquierdo del huso, de ahí que se tome la esquina inferior izquierda de la ortoimagen como origen de coordenadas relativo. La Tierra se divide en varios husos, siendo el huso 30 el correspondiente a la mayor parte del territorio de la Península, el 31 para las Baleares y la zona más al este de la Península, el 30 para la zona de Galicia, y los husos 27 y 28 para las Islas Canarias.

usan cámaras con una línea base pequeña, o un marcado manual poco preciso. En la triangulación se aplica el método de Levenberg-Marquardt para la optimizacion del resultado, pero no se realiza un proceso de optimización con SBA para todo el escenario completo. Debido al número de variables implicadas en esta optimización, SBA consumiría un tiempo de computación considerable, haciendo poco interesante su integración en el prototipo, que pretende ser sencillo y ligero para el usuario. La optimización de los puntos mauales con SBA queda como tarea para futuras mejoras del prototipo.

Figura B.2: Ventana se selección del trío de cámaras.

Selección manual de las tres cámaras usadas para triangular los puntos marcados manualmente. Es necesario que el trío de cámaras tenga una buena línea base entre sí, de ahí que se deje esta tarea al usuario para que compruebe visualmente que las cámaras son adecuadas. Fuente: e.p.

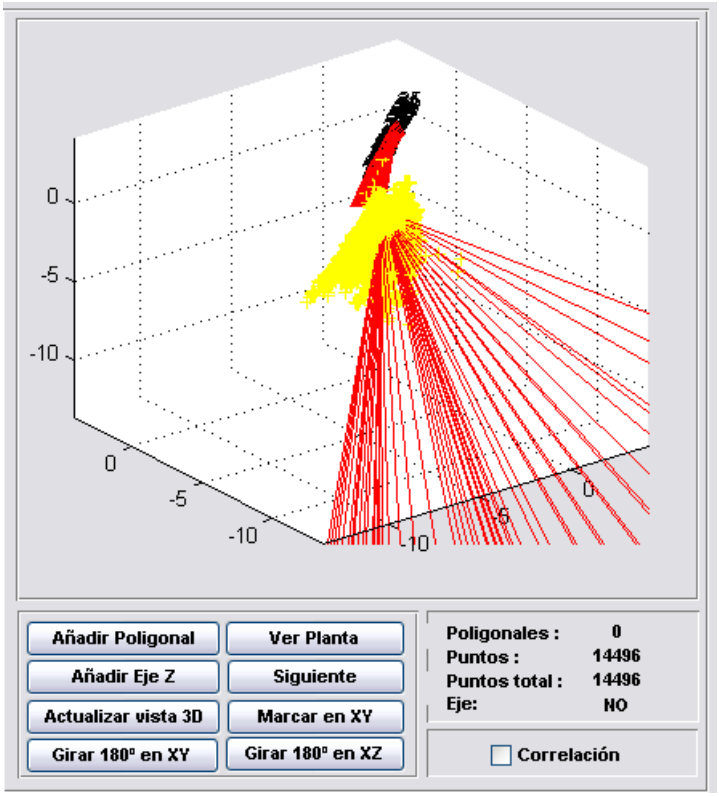

Figura B.3: Ventana de inicio georreferenciación

En Orto3D se usa la siguiente notación: los puntos amarillos representan los puntos reconstruidos por Bundler, los triángulos rojos representan las posiciones de las cámaras en el escenario y las líneas rectas son sus ejes principales. Para cada cámara se representa también su identificador. Fuente: e.p.

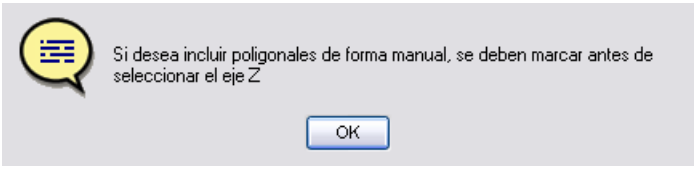

Figura B.4: Aviso antes del marcado

Al iniciar la georreferenciación se avisa al usuario del orden de marcado de los puntos. Fuente: e.p.

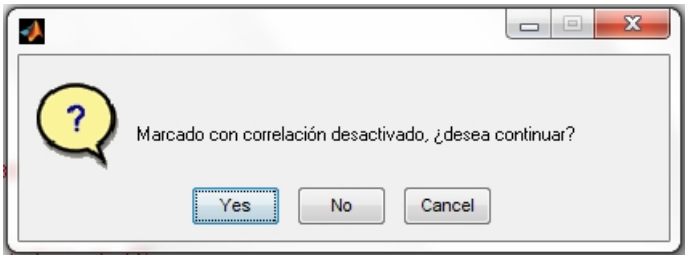

Figura B.5: Aviso correlación

Ventana para indicar al usuario que la opción de marcado con correlación está desactivada. Fuente: e.p.

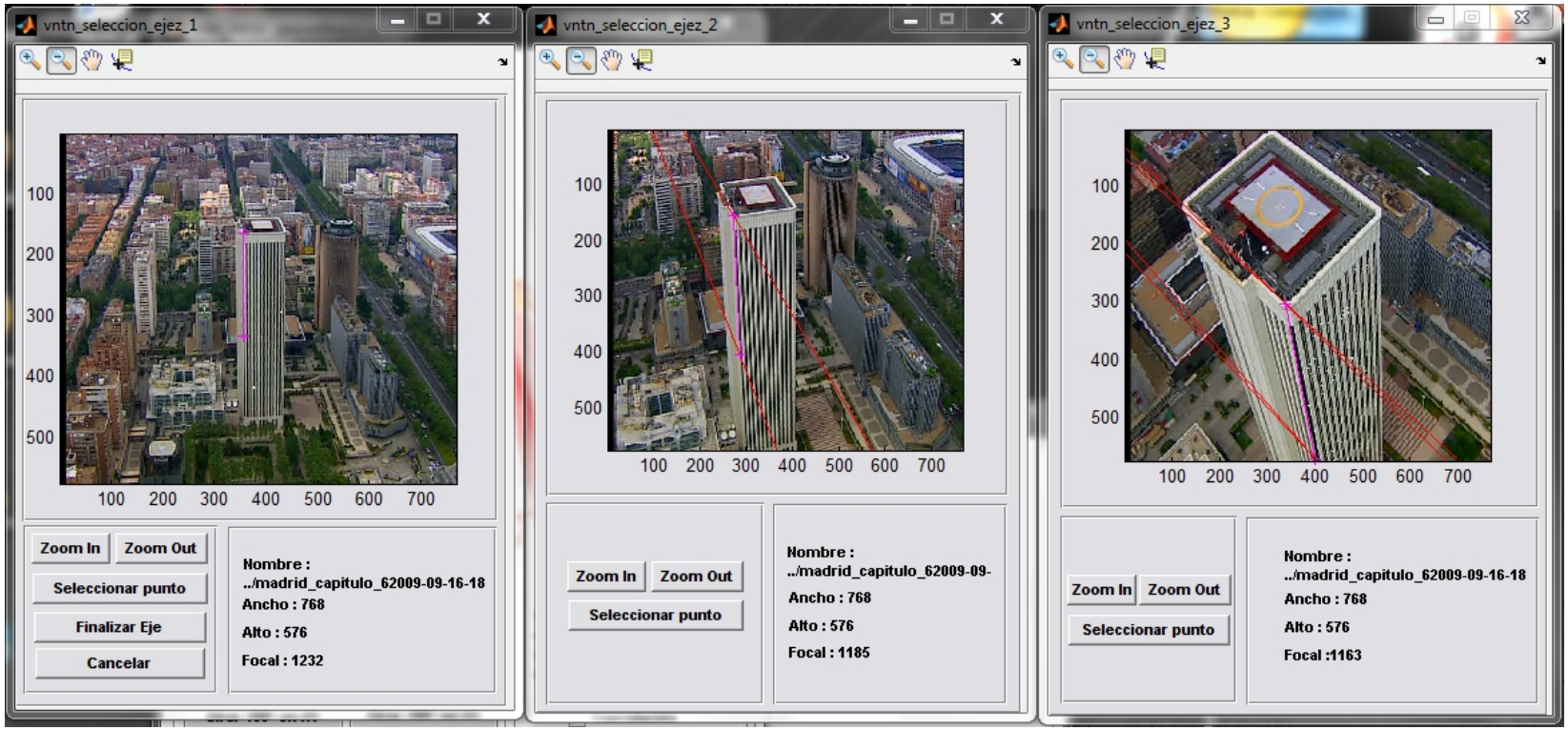

Figura B.6: Marcado de arista como eje Z

Marcado de puntos correspondientes en tres imágenes para seleccionar una arista (en magenta) que se tomará como eje Z del sistema de referencia. El marcado se realiza con ayuda de las líneas epipolares (en rojo). Fuente: e.p.

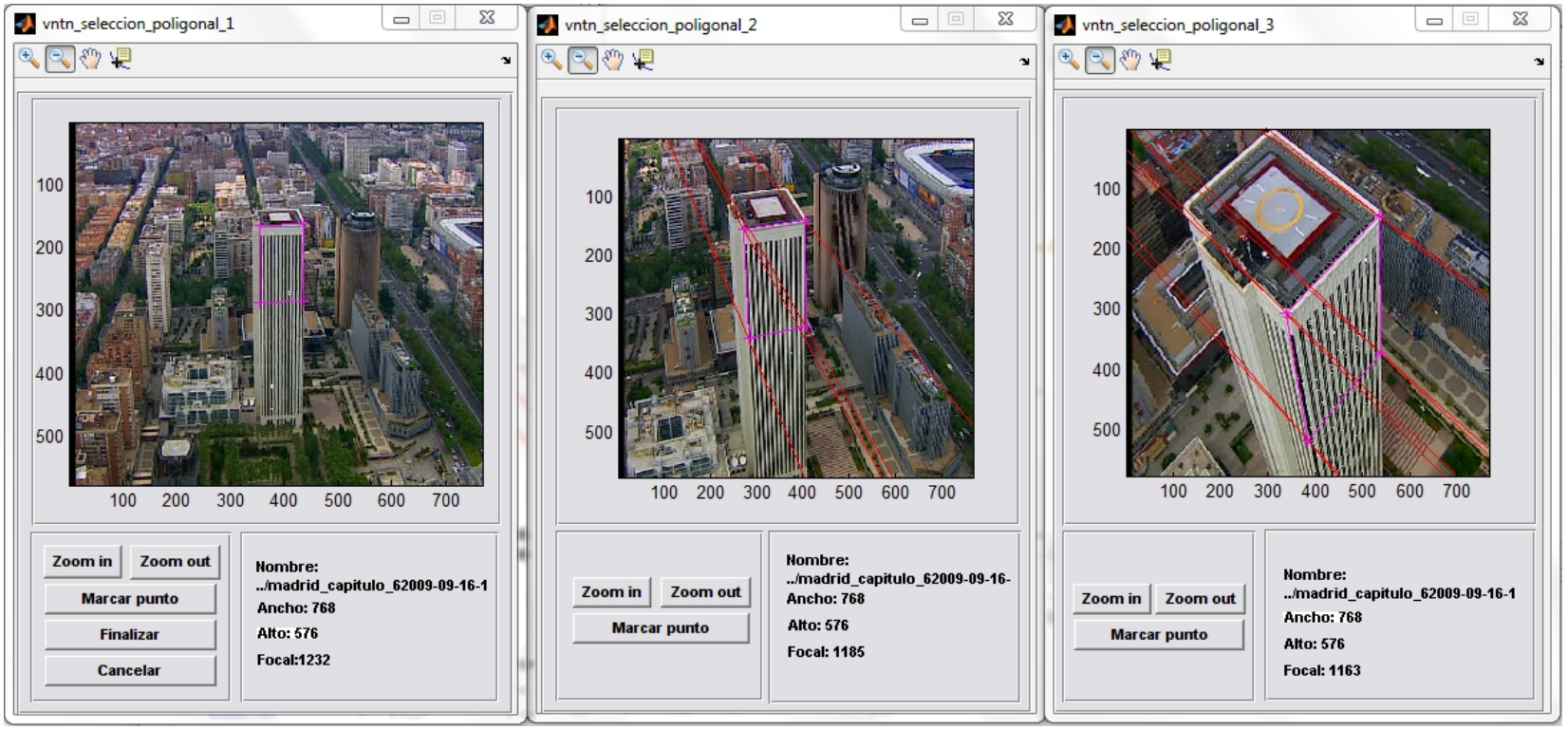

Figura B.7: Marcado de poligonal

Marcado manual de cuatro puntos correspondientes en tres imágenes para obtener un cuadrilátero en el escenario. El proceso es similar al marcado de una arista, con la diferencia de marcar cuatro puntos en vez de dos. Fuente: e.p.

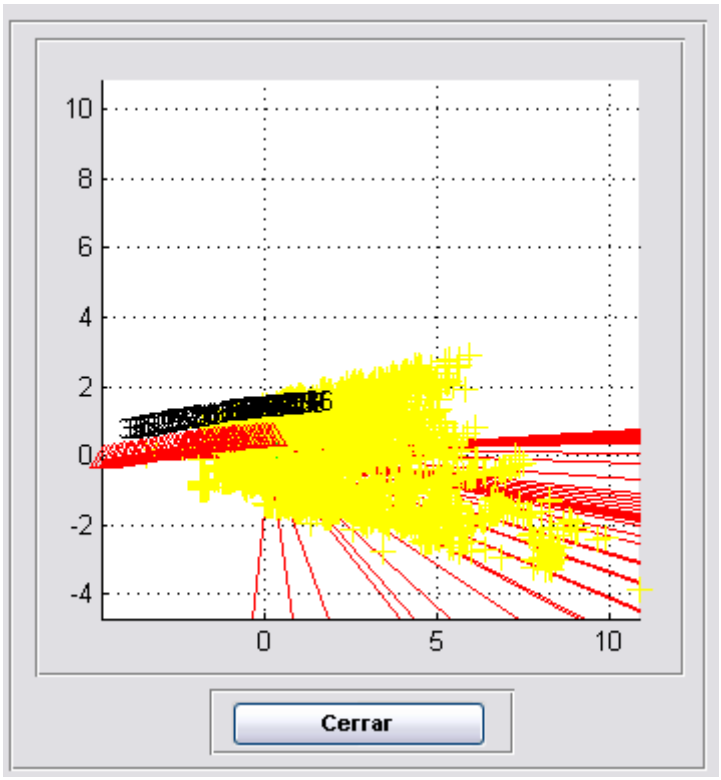

Figura B.8: Ventana ver planta

Visualización del escenario en planta. Se sigue la misma notación que en la figura B.3. Fuente: e.p.

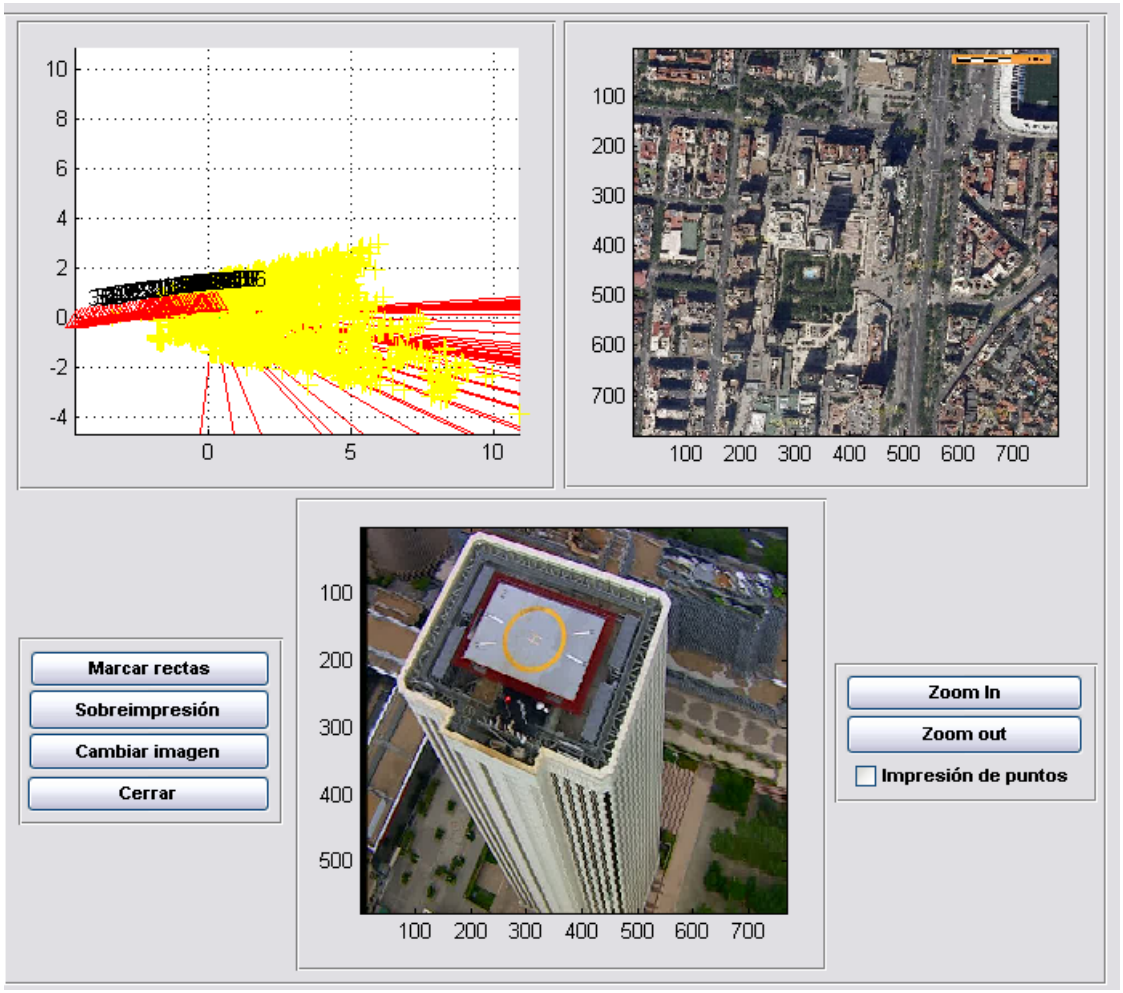

Figura B.9: Ventana marcar rectas referencia

Marcado de rectas de referencia en la planta de la reconstrucción y en la ortoimagen para la posterior georreferenciación del escenario. En la parte superior izquierda se presenta la planta del escenario, en la derecha la ortoimagen y en la parte inferior los fotogramas del escenario. Fuente: e.p.

Figura B.10: Ventana georreferenciar

Georreferenciación del escenario a partir de las rectas correspondientes marcadas en la ventana anterior. La representación sigue la notación comentada en la figura B.3. Fuente: e.p.

Figura B.11: Ventana marcar referencia UTM

Permite marcar manualmente puntos de referencia UTM en la ortoimagen de los que conoce sus coordenadas para con ellos obtener la relación metros/píxel del escenario. Fuente: e.p.

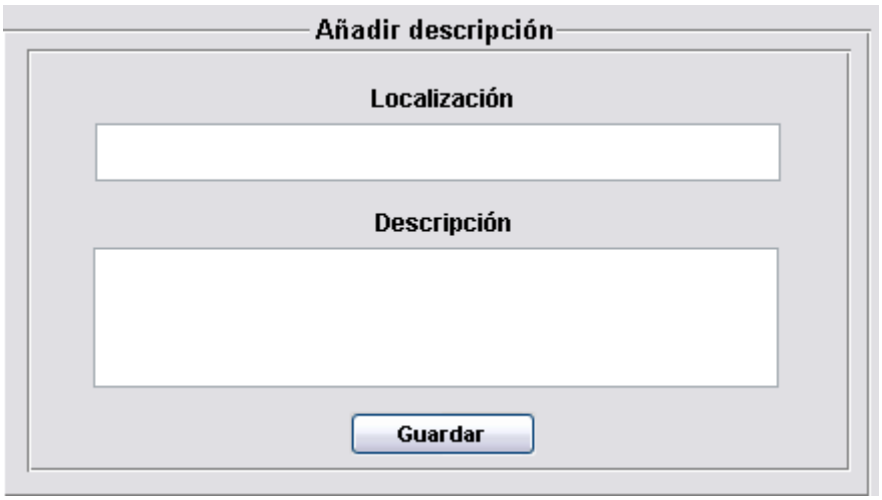

Figura B.12: Ventana añadir descripción

La ventana permite añadir una breve descripción acerca del escenario. Fuente: e.p.

B.5. Cargar escenario

Una vez georreferenciado un escenario es posible visualizar los resultados y añadir nuevas aristas y nuevas poligonales. Esto se hace por medio de la funcionalidad de *Cargar escenario* llamada desde la ventana de inicio de orto3D (figura B.1). Lo primero que carga es la ventana de selección de cámaras (figura fig:seleccion3camaras) para la triangulación cuya funcionalidad y funcionamiento es idéntico al explicado en la sección B.4.

Una vez seleccionadas las tres cámaras, la siguiente ventana que se abre es la ventana del escenario cargado (figura B.13). Esta ventana presenta a la izquierda la ortoimagen con la nube de puntos y las cámaras sobreimpresas y a la derecha la nube de puntos y las camaras en 3D. La notación seguida para los puntos y para las cámaras es la misma que para la georreferenciación del escenario. Los puntos 3D se presentan en color amarillo, y las cámaras con un triángulo rojo, con su número índice y con el eje principal. En la parte superior de la ventana se incluye una barra de herramientas con zoom, desplazamiento de imagen, rotación de imagen, y visualización de coordenadas de los puntos. En la parte inferior se incluye funcionalidad específicamente desarrollada para la aplicación. Esta funcionalidad permite ver las imágenes con los puntos correspondientes sobreimpresos (*ver imagen*), las coordenadas de un punto en la ortoimagen (*ver coordenadas*), y la distribución de puntos característicos y correspondientes en las imágenes. Permite también ver las focales y la información del escenario que se describen en la sección B.5. La razón de incluir la funcionalidad en la ventana principal de orto3D y en la funcionalidad de cargar es poder acceder a esta información desde ambos puntos, siendo la primera más directa y útil cuando se desa conocer las focales estimadas sin necesidad de cargar todo el escenario.

Además de la anterior funcionalidad, a la izquerda se incluyen botones que permiten añadir aristas, poligonales, y a diferencia de la georreferenciación, también se pueden tomar medidas. Las medidas se basan en el marcado de aristas en el escenario de forma análoga a cómo se hace el marcado con el eje Z. El módulo de la arista seleccionada será la medida en metros deseada, cuya utilidad es servir de dato adicional para validar los escenarios. Al comparar los datos obtenidos con medidas reales de edificios o estructuras conocidas se podrá tener una idea del error de reconstrucción y georreferenciación. A continuación se describen cada una de las funciones comentadas.

La primera función descrita es la de visualización de imágenes, iniciada cuando se acciona el botón *Ver imagen*. Se carga la ventana presentada en la figura fig:vntnverimagen, que presenta dos de las imagenes de la secuencia de vídeo solapadas, con todos los puntos correspondientes sobreimpresos en amarillo. Las líneas y puntos verdes presentan únicamente los puntos correspondientes entre las imágenes que se presentan en esta ventana. En el menú inferior se presenta el nombre de la imagen, así como la longitud focal estimada para la cámara, y el ancho y alto de la imagen. Los botones *Cambiar imagen*, uno para cada imagen, permiten cambiar la imagen mostrada por otra del conjunto de fotogramas usados para reconstruir. Los botones *Puntos 2D* permiten desactivar la presentación de los puntos correspondientes sobre las imágenes. Por defecto, siempre se muestran en ambas, pero cuando se desactiva este botón se presenta sólo en la imagen izquierda.

Volviendo a la ventana de cargado del escenario, la opción de ver coordenadas presenta las coordenads UTM del punto seleccionado en la ortoimagen. Las coordenadas mostradas son el eje X, el eje Y, y el huso a los que pertenece la imagen.

La información de puntos muestra datos sobre la correspondencia de puntos entre imágenes. Por un lado se presenta el número de correspondencias por imagen, es decir el número de puntos correspondientes que una imagen comparte con el resto de imágenes. Un ejemplo de esta ventana se muestra en la figura B.15. La figura B.16 muestra la distribución de correspondencias, es decir, cuántas imágenes comparten una misma trayectoria. Adicionalmente se presentan también el número de puntos correspondientes que presenta cada imagen. Como se puede ver en la figura B.17, el número de puntos correspondientes es muy inferior al número de puntos característicos encontrados.

En esta parte de la aplicación también se pueden añadir nuevas poligonales y nuevas aristas mediante la selección de puntos manuales en tres imágenes de manera análoga a como se hace en la parte de georreferenciación de la aplicación. Por ser la misma funcionalidad, no se comentará más sobre ello. Si se requiere más información, revisar la sección B.4 que explica el procedimiento de georreferenciación en este mismo capítulo. La única novedad introducida en esta parte de la aplicación es la función de medida. Como se ha comentado anteriormente en esta sección, la diferencia con la selección de una arista es que calcula el módulo de la arista elegida y la presenta en la figura derecha de la ventana de cargado del escenario (figura B.13). En la figura B.18 se muestra un ejemplo de medida.

Una vez seleccionados nuevos puntos, poligonales y aristas, es posible guardar todo el escenario de la misma manera que se guarda en georreferenciar, sobreescribiendo los ficheros existentes.

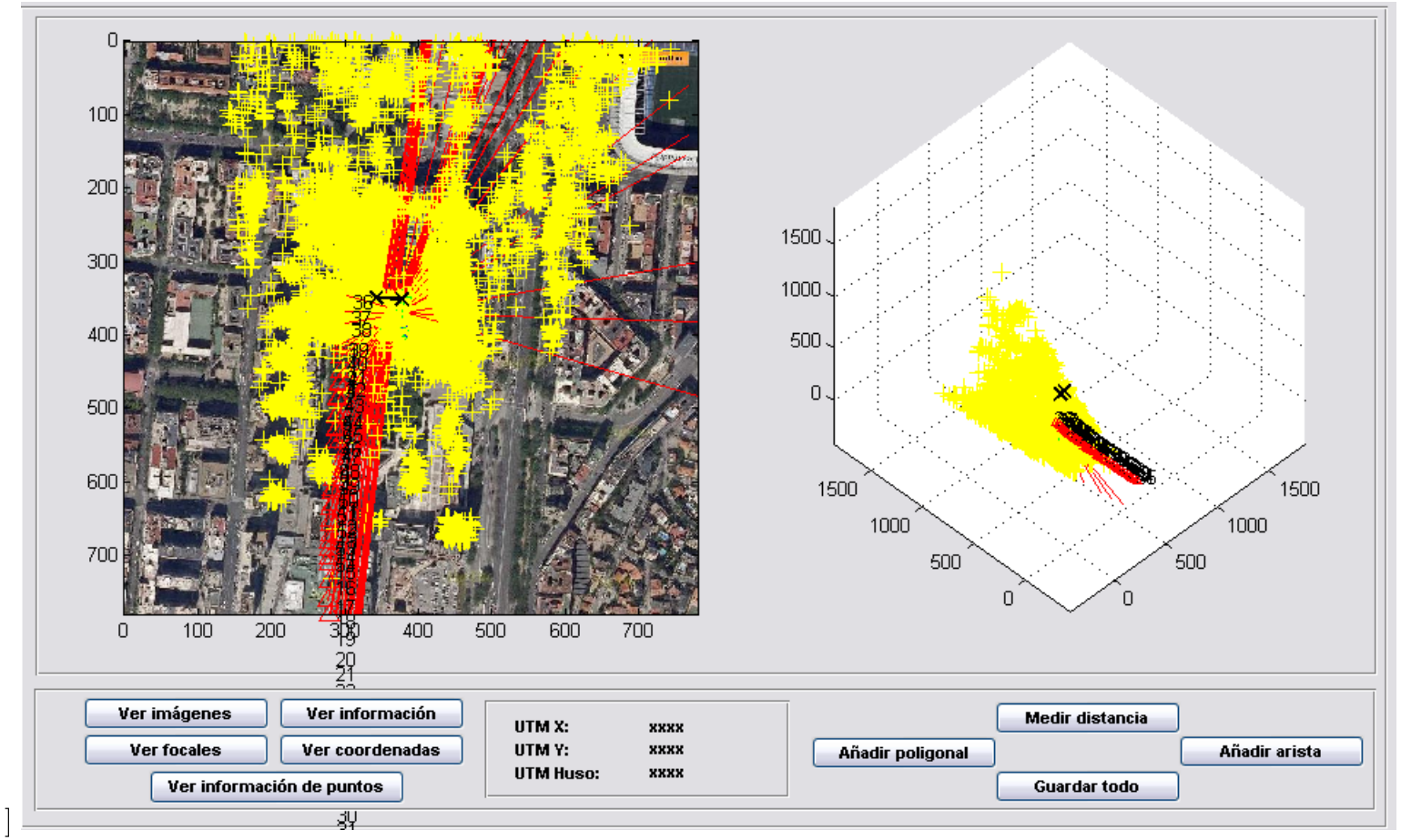

Figura B.13: Ventana escenario cargado

La ventana carga un escenario ya georreferenciado y lo presenta con una vista en 3D y con la planta superpuesta a la ortoimagen. Se sigue la notación comentada en la figura B.3.

Fuente: e.p.

Figura B.14: Ventana ver imagen

Permite ver por pares los fotogramas usados en la reconstrucción para analizar las correspondencias encontradas. Fuente: e.p.

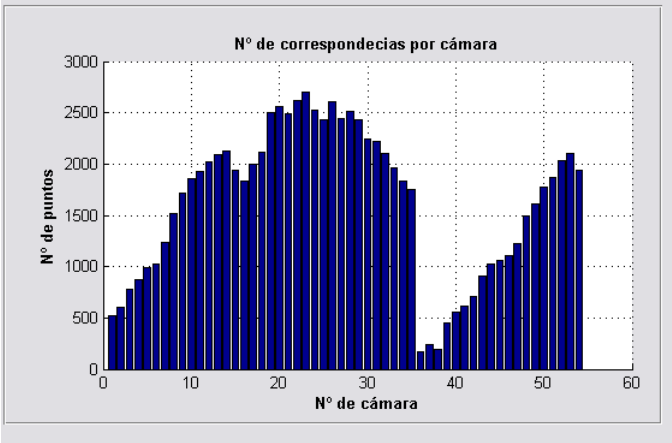

Figura B.15: Gráfica de correspondencias por imagen

Presenta una gráfica con el número de correspondencias encontradas en cada imagen. Fuente: e.p.

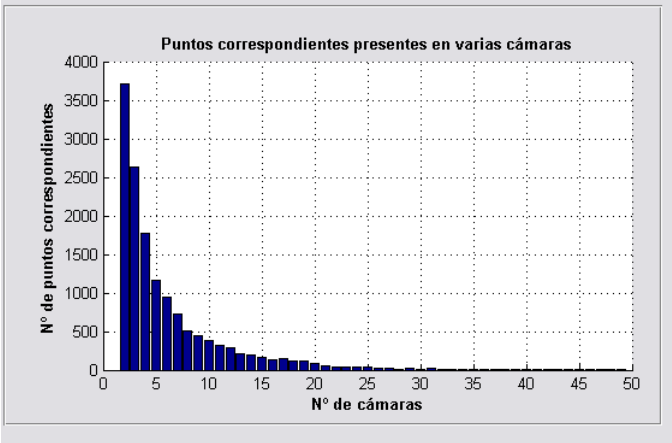

Figura B.16: Gráfica de distribución de correspondecias

Presenta la cantidad de trayectorias con respecto al número de cámaras en cada una. Fuente: e.p.

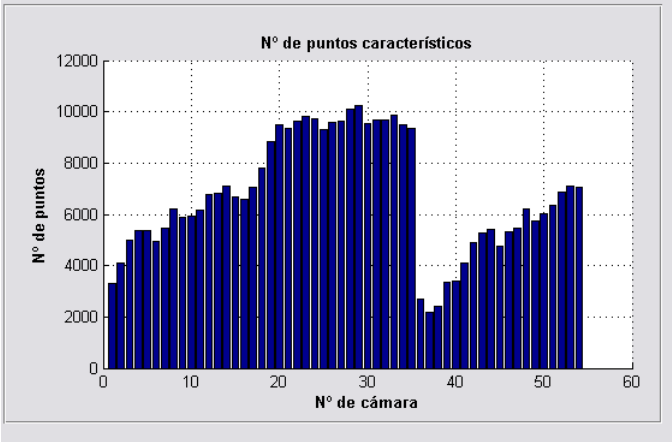

Figura B.17: Gráfica de característicos por imagen

Presenta la cantidad de puntos característicos encontrados para cada una de las imágenes. Fuente: e.p.

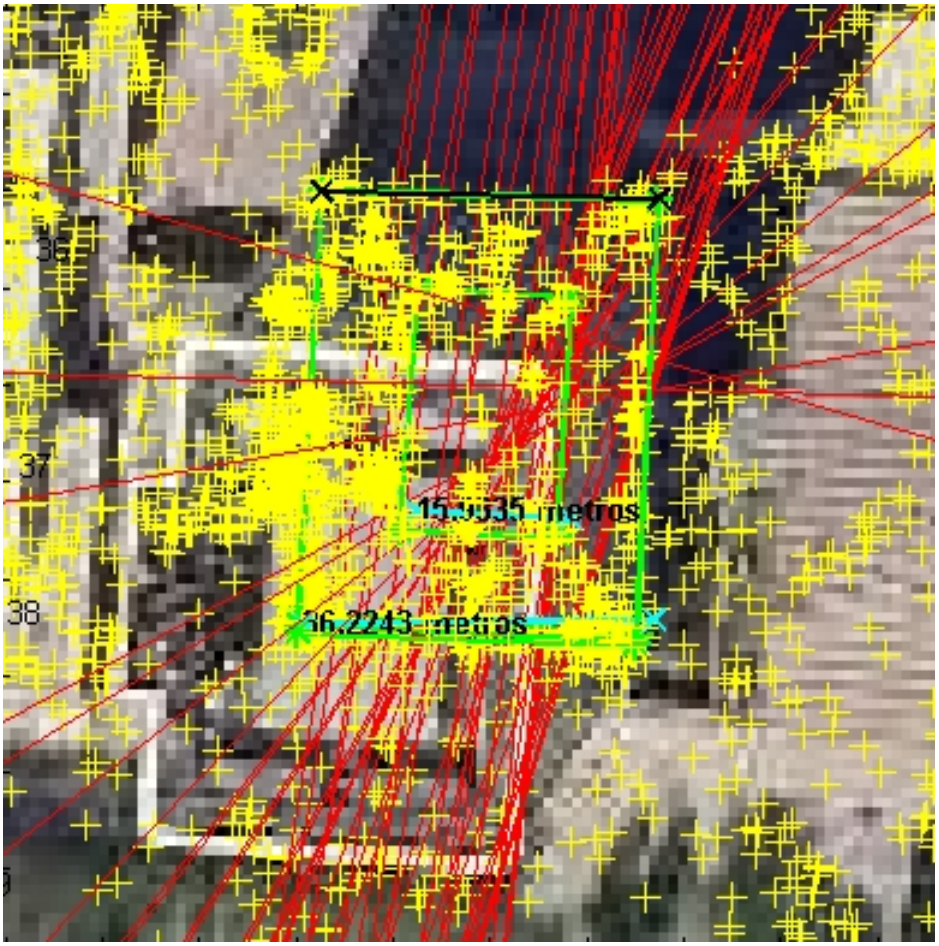

Figura B.18: Ejemplo de medida con Orto3D

Medida aproximada en metros de una recta marcada en el escenario TORRE_PICASSO_AZCA_MOD_1 georreferenciado previamente con Orto3D. Se presenta un detalle de la ortoimagen de la figura C.3 con el escenario reconstruido y las medidas sobreimpresas. Fuente: e.p.

B.6. Mostrar focales y datos

En la ventana de inicio de la aplicación (figura B.4) se puede selecionar la visualización de la focal del escenario sin necesidad de iniciar la georreferenciación ni el cargado del escenario. La presentacion de la focal permite visualizar las estimaciones de focal realizadas por Bundler para el escenario, pudiendo descartar aquellos con focales muy dispares antes incluso de realizar todo el procedimiento de georreferenciación. Para conocer más sobre las causas por las que las focales recuperadas pueden ser erróneas ir a 5.3.4. La presentación de las focales se hace en una ventana que presenta una gráfica con los valores de las focales y el valor medio de éstas. Éste se calcula tomando únicamente los valores absolutos de las focales en el rango (f_{min}, f_{max}, con $f_{min} = 0{,}8(ancho\_imagen)$ y $f_{max} = 200(ancho\_imagen)$. Con este rango se descartan las focales nulas (cámaras no recuperadas por Bundler) y las excesivamente grandes. Este rango incluye la mayoría de las cámaras del mercado incluyendo las de televisión[8]. El valor por el que se multiplica el ancho de la imagen es un ratio que se calcula como $ratio = \frac{f_{mm}}{ancho_CCDmm}$. Por poner un par de ejemplos, la cámara Canon Powershot A520 tiene un rango de $(1{,}0069, 4{,}0278)$, y una cámara similar a la usada para la grabación de los documentales de las pruebas tiene un rango de $(1{,}7045, 82{,}5000)$[9] [10] [11] [12] [13]. Otra opción disponible es ver estos resultados en un histograma de valores, para conocer como se distribuyen en contenedores de valores. Estas ventanas se presentan en las figuras B.19, B.20.

Una vez realizada la georreferenciación, explicada en la sección B.4 y guardados los datos de georreferenciación, se dispone de la información sobre el escenario que se presenta con el botón *Mostrar datos escenario*. La información presentada en esta ventana (figura B.21) permite tener una idea rápida sobre el escenario y su localización, así como unas coordenadas UTM correspondientes al origen de la ortoimagen (coordenadas de la esquina inferior izquierda de la imagen), versiones de Bundler y Orto3D usadas, fecha de realizacion de la reconstruccion, fecha de la georreferenciación, número de fotogramas/cámaras usadas en la reconstrucción y cantidad de ellas reconstruidas o descartadas, así como número de puntos reconstruidos, cantidad de puntos añadidos manualmente, número de rectas y el numero de poligonales añadidas. Como se puede ver en la figura B.21, las coordenadas geográficas (latitud/longitud) no se emplean, pero se añaden al prototipo para ser implementadas en futuras versiones de la aplicacion, al igual que el error de posicionamiento tampoco se presenta[14]. Al no disponer de coordenadas de posición de cámaras de referencia tomadas con GPS para las imágenes aéreas no se puede calcular este error. La relación metros/píxel es la calculada mediante los puntos de referencia UTM.

[8] http://www.canon.com/bctv/products/index.html

[9] Ejemplo para sistema WESCAM 16"con cámara Sony BVP-570, sensor CCD 2/3"de ancho 8,8mm, y con juego de lentes de zoom Canon de focales 15–726mm

[10] http://www.aerialcamerasystems.com/stab_wescam16.php

[11] http://www.aerialcamerasystems.com/cameras_sony570-T450.php

[12] http://en.wikipedia.org/wiki/Charge-coupled_device

[13] http://www.helipro.co.nz/presentation/HeliProPres.aspx?ID=9566

[14] nd: no disponible.

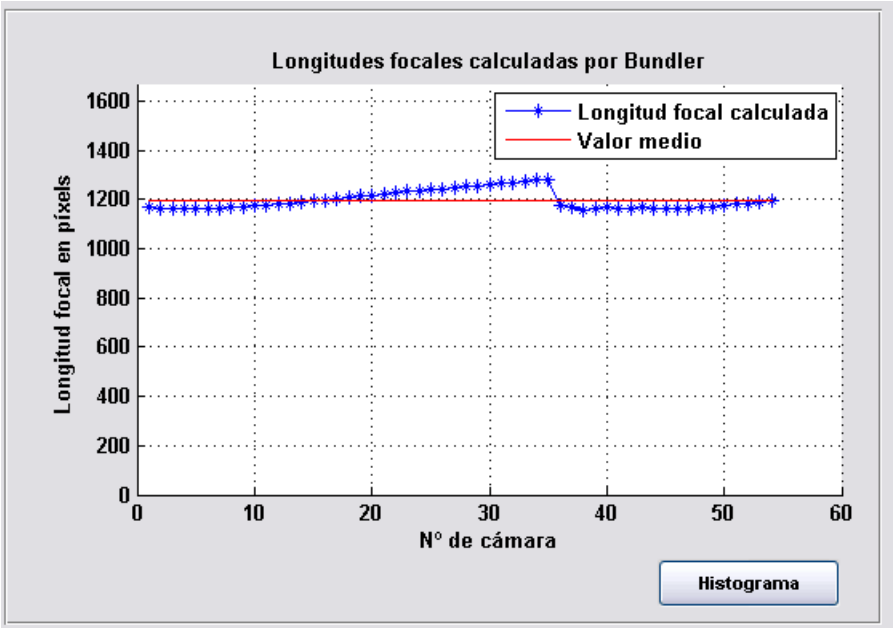

Figura B.19: Ventana presenta focales

Valores de longitudes focales para uno de los escenarios. Fuente: e.p.

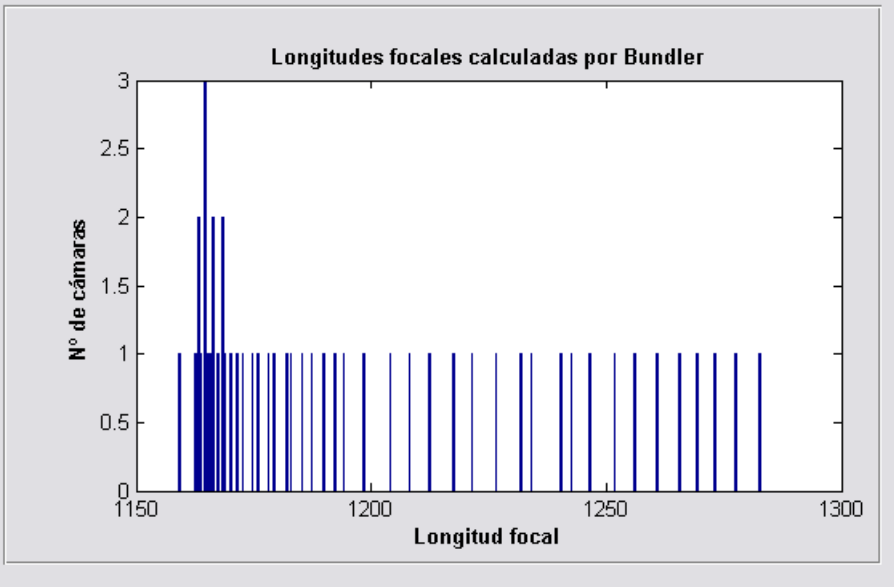

Figura B.20: Ventana presenta histograma de focales

Distribución de las longitudes focales para uno de los escenarios. Fuente: e.p.

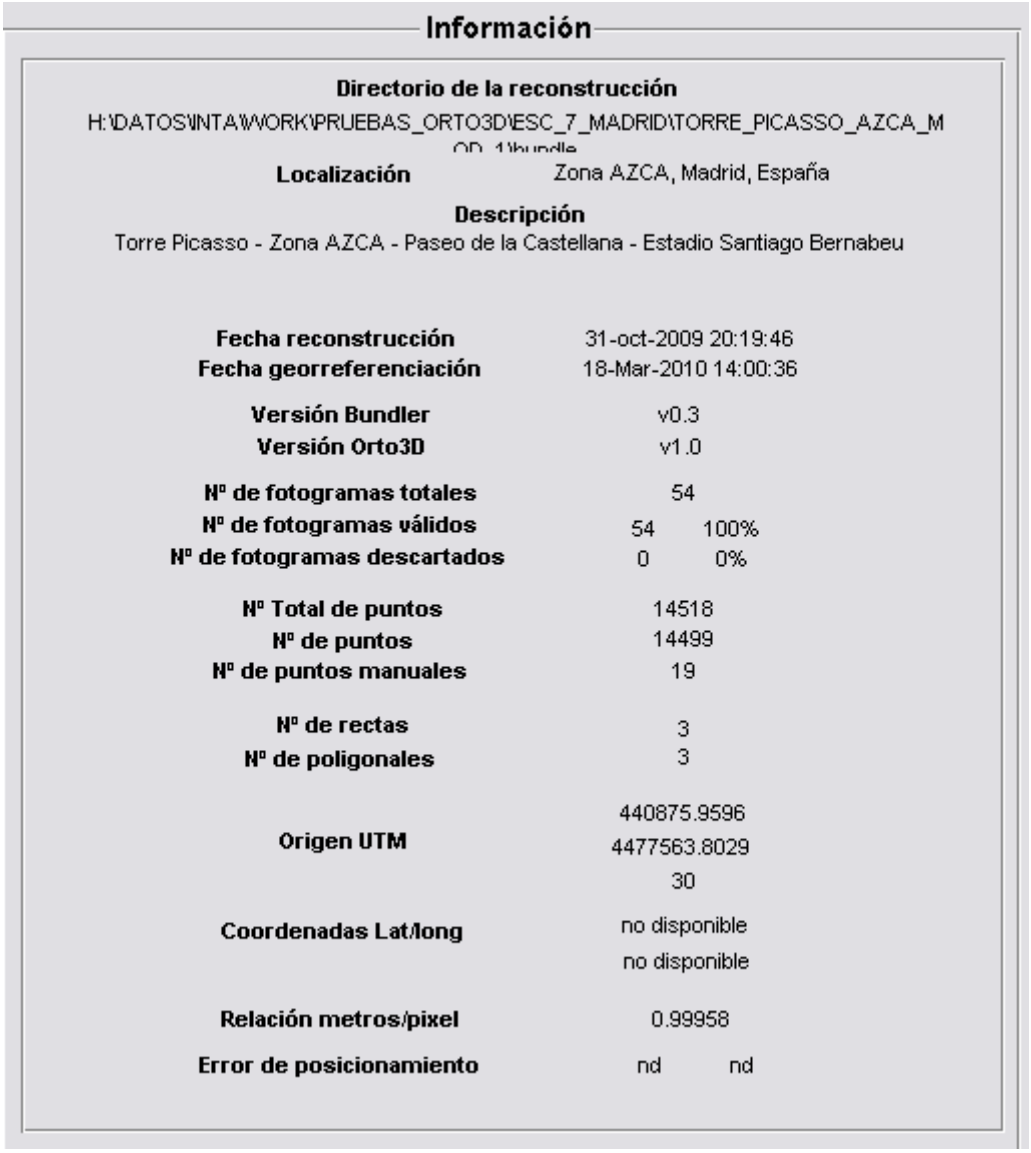

Figura B.21: Ventana mostrar datos del escenario

Ventana de Orto3D que presenta información sobre el escenario tales como localización, descripción, número de cámaras entre otros datos. Fuente: e.p.

B.7. Resultados de la aplicación

Orto3D genera los siguientes ficheros de resultados:

- *orto3D.out*
- *orto3D.ply*
- *orto3D_plano.ply*
- *orto3DInfo.txt*
- *orto3Dmanual.txt*
- *orto3D_test_georreferenciar.out*
- *error_reproyeccion.txt*
- *error_reproyeccionLMA.txt*
- *reprojectedPoints.txt*
- *reprojectedPointsLMA.txt*

El fichero *orto3D.out* tiene un contenido similar al fichero *bundle.out*. La única diferencia son las unidades, que para el caso del primer fichero, se dan en metros y referidas a un punto UTM. Para evitar la operación con valores muy grandes, se referencia todo el escenario a las coordenadas UTM de la esquina inferior izquierda de la ortoimagen empleada. De esta ortoimagen se tienen dos coordenadas, y la tercera se obtiene asumiendo que la ortoimagen se encuentra en $Z = 0$ en el escenario tridimensional. El fichero *orto3D.ply* es exactamente igual que los ficheros PLY obtenidos por Bundler, por lo que no se comentará más sobre éste. La diferencia viene dada por *orto3D_plano.ply* que es una representación en PLY de la ortoimagen usada. Si se usan ambos ficheros en diferentes capas, se puede visualizar el escenario sobre la ortoimagen. El fichero de información *orto3DInfo.txt* contiene los datos mostrados en la opción de mostrar datos del escenario en el prototipo. Los datos contenidos son el directorio del escenario, la descripción y localización, así como las fechas en las que se reconstruyó y se georreferenció junto con las versiones de Bundler y Orto3D empleadas. También se incluye el número de cámaras del escenario, las válidas y las no válidas (ambas con su porcentaje del total), el número de puntos reconstruidos, los puntos seleccionados manualmente, las poligonales y rectas marcadas y por último las coordenadas UTM del origen del escenario y la relación metros/píxel de la ortoimagen. *orto3Dmanual.txt* contiene las coordenadas de los puntos marcados manualmente sobre las imágenes.

El resto de ficheros se usan para tareas de depuración del código. Se explicarán por encima al no ser importantes para manejar la herramienta, salvo que se desee modificar el código fuente. El fichero *orto3D_test_georreferenciar.out* contiene el escenario georreferenciado para tareas de depuración. Se crea si la variable global $DEBUG\_GEORREF = true$. El fichero *error_reproyeccion.txt* contiene el error de reproyección de los puntos al finalizar la optimización de la triangulación con Levenberg-Marquardt. El fichero *error_reproyeccionLMA.txt* guarda el error de reproyección para cada iteración de LMA. Los ficheros *reprojectedPoints.txt* y *reprojectedPointsLMA.txt* contienen las coordenadas de los puntos reproyectados para los errores de reproyección escritos en los ficheros anteriores.

Apéndice C

Escenarios de pruebas

En este apéndice se presentan las ortoimágenes y coordenadas UTM de referencia usadas para alguno de los escenarios incluidos en el DVD. En concreto se incluyen aquellos usados para el desarrollo de Orto3D:

- POLITÉCNICA
- MECO_3_MOD_1
- TORRE_PICASSO_AZCA_MOD_1

Todas las ortoimágenes han sido extraídas de SIGPAC [45]. Para las coordenadas se han usado dos sistemas diferentes, ED50 y WGS84. Las primeras versiones de SIGPAC empleaban únicamente el primero, mientras que las últimas permiten trabajar con ambos sistemas.

C.1. Politécnica

	ESC_2_POLITECNICA
Localización	Alcalá de Henares, Madrid
Edificio	Escuela Politécnica, esquina oeste
Descripción	Esquina del edificio de la escuela
Comentarios	Las paredes forman un ángulo de 90º
Otros datos	Edificio fácil de reconocer

Tabla C.1: Descripción escenario Escuela Politécnica

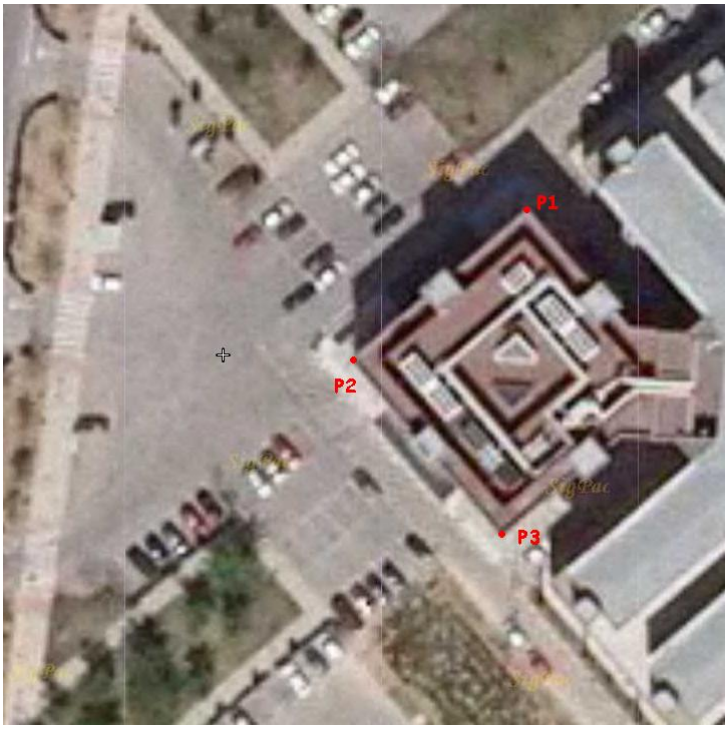

Figura C.1: Ortoimagen de la Escuela Politécnica, Alcalá de Henares
Imagen tomada de SIGPAC y usada para las pruebas de georreferenciación. Los puntos presentados en la imagen son los puntos de referencia UTM, cuyas coordenadas se pueden ver en la tabla. Fuente: e.p. a partir de [45].

Datum=ED50	P1	P2	P3
UTM x	470417.85	470396.38	470414.77
UTM y	4485002.76	4484983.98	4484962.48
UTM huso	30	30	30
GEO latitud	40º30'52.27"N	40º30'51.66"N	40º30'50.97"N
GEO longitud	3º20'57.03"W	3º20'57.94"W	3º20'57.16"W

Tabla C.2: Coordenadas escenario Escuela Politécnica
Coordenadas de los puntos presentados en la figura C.1

C.2. Meco

	ESC_7_MADRID\MECO_MOD_1
Localización	Meco, Madrid
Edificio	Iglesia de Nuestra Señora de la Asunción
Descripción	Campanario de la iglesia
Comentarios	Secuencia de vídeo tomada alrededor de la torre
Otros datos	

Tabla C.3: Coordenadas escenario Meco

Figura C.2: Ortoimagen del centro de Meco, Madrid
Imagen tomada de SIGPAC y usada para las pruebas de georreferenciación. Las coordenadas de los puntos marcados en la imagen se pueden ver en C.4. Fuente: e.p. a partir de [45].

Datum=ED50	P1	P2	P3	P4
UTM x	472004.1	472208.5	472572.4	472193.2
UTM y	4489444.2	4489627.0	4489547.3	4489264.3
UTM huso	30	30	30	30
GEO latitud	40.55458611º N	40.55624167º N	40.55553333º N	40.55297222º N
GEO longitud	3.33065000º W	3.32824444º W	3.32394167º W	3.32840833º W

Tabla C.4: Puntos de referencia escenario Meco

C.3. Torre Picasso - zona Azca (Madrid)

	ESC_7_MADRID\TORRE_PICASSO_AZCA_MOD_1
Localización	Zona Azca, Madrid
Edificio	Torre Picasso
Descripción	Torre Picasso y alrededores de la zona comercial Azca
Comentarios	Secuencia de vídeo tomada en *forward motion* con respecto a la torre
Otros datos	

Tabla C.5: Coordenadas escenario Torre Picasso - Zona Azca

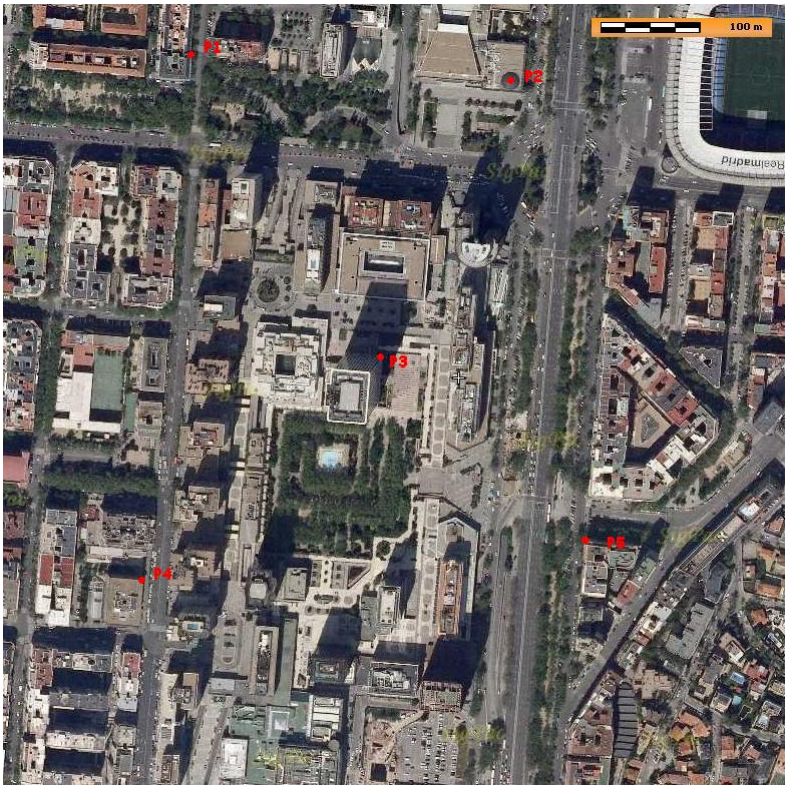

Figura C.3: Ortoimagen de la Torre Picasso y la zona Azca, Madrid

Imagen tomada de SIGPAC y usada para las pruebas de georreferenciación. Las coordenadas de los puntos marcados en la imagen se pueden ver en C.6. Fuente: e.p. a partir de [45].

Datum=WGS84	P1	P2	P3	P4	P5
UTM x	441063.9	441382.2	441254.7	441016.2	441456.4
UTM y	4478292.5	4478266.3	4477990.9	4477767.6	4477809.2
UTM huso	30	30	30	30	30
GEO latitud	40.45320556º N	40.45299167º N	40.45050278º N	40.44847222º N	40.44888056º
GEO longitud	3.69505833º W	3.69130278º W	3.69278056º W	3.69557222º W	3.69038333º

Tabla C.6: Puntos de referencia escenario Azca

Apéndice D

Notas Matemáticas

D.1. Factorización QR

Se denomina factorización o descomposición QR de una matriz a la descomposición de ésta como producto de una matriz ortogonal por una matriz triangular superior. Esta descomposición se emplea para el cálculo de vectores y valores propios de una matriz.

Sea una matriz cuadrada real A, la descomposición QR es tal que

$$A = QR \tag{D.1}$$

con Q una matriz ortogonal tal que $Q^T Q = I$ y R una matriz triangular superior. Para una explicación más detallada, ver [79].

D.2. Factorización SVD

Se denomina SVD a la factorización de una matriz real o compleja A de tamaño $m \times n$ de la forma

$$A = UDV^T \tag{D.2}$$

con U una matriz ortonormal unitaria de tamaño $m \times m$, D una matriz $m \times n$ diagonal con componentes reales no negativos, y V^* el transpuesto conjugado de la matriz V de tamaño $n \times n$. Las componentes diagonales de D están ordenadas en orden decreciente de manera que la última componente de la diagonal tiene el menor valor. Los valores de la diagonal D se denominan valores singulares de A. Para una explicación más detallada, ver [79].

D.3. Matriz y determinante Jacobiano

La matriz Jacobiana es una matriz con las derivadas parciales de primer orden de una función.

Sea F una función determinada por m funciones reales, en n variables, la matriz Jacobiana

viene definida por

$$J = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{pmatrix} \tag{D.3}$$

Cuando la matriz Jacobiana es cuadrada $n \times n$, entonces se puede calcular el determinante Jacobiano, también llamado simplemente Jacobiano.

D.4. Matriz Hessiana

También conocido simplemente como Hessiano, es la matriz cuadrada de derivadas parciales de segundo orden de una función tal que:

$$H = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} \tag{D.4}$$

D.5. Matriz Antisimétrica

Una matriz antisimétrica (en la bibliografía inglesa "skew-symmetric matrix") es una matriz cuadrada $n \times n$ cuya transpuesta es igual a su negativa,

$$A = -A^T \tag{D.5}$$

Por esta razón, si las entradas de la matriz son de la forma a_{ij} con i filas y j columnas, la condición de simetría obliga a que $a_{ij} = -a_{ij}$. El determinante de esta matriz es nulo cuando el número n es impar, ya que

$$\det(A) = \det(A^T) = \det(-A) = (-1)^n \det(A) \tag{D.6}$$

condición que sólo se cumple si $\det(A) = 0$ para n impar.

Cuando se use la notación $[a]_x$ indica que se usa el vector $a = (x, y, w)$ convertido a forma matricial antisimétrica tal que

$$[a]_x = \begin{bmatrix} 0 & w & -y \\ -w & 0 & x \\ y & -x & 0 \end{bmatrix} \tag{D.7}$$

Mediante esta matriz antisimétrica se puede calcular el producto vectorial de dos vectores a y b

$$a \times b = [a]_x b = (a^T [b]_x)^T \tag{D.8}$$

cumpliendo además que

$$a \times a = [a]_x a = 0 \tag{D.9}$$

Apéndice E

Contenido del DVD

A continuación se describe el contenido de cada carpeta dentro del DVD:

- Bundler: distribuciones de Bundler, binarios y código fuente, documentación Doxygen de la tercera versión.
  - V0.1: código fuente y binarios de la primera versión de Bundler.
  - V0.2: código fuente y binarios de la segunda versión de Bundler.
  - V0.3: código fuente, binarios y documentación Doxygen de la tercera versión de Bundler.
    - source: código fuente de la tercera versión.
    - bin: binarios.
    - doxygen: documentación del código en HTML con Doxygen.
  - V0.4: código fuente de la cuarta versión (no se han publicado los binarios).
- Orto3D: prototipo en Matlab.
  - source: funciones y ventanas de la aplicación.
  - html: ayuda de Orto3D.
- Tests: resultados de las pruebas.
  - Vídeos SIVA: vídeos tomados desde UAV y fotogramas de éstos.
  - Vídeos DOCU: fotogramas de documentales aéreos tomados para televisión.
  - Fotografías: fotografías usadas para probar la reconstrucción.
- Docs: manuales, memorias, notas del trabajo.
  - Memoria TFC: este documento y las imágenes empleadas.
  - Anteproyecto: el documento del anteproyecto y las imágenes empleadas.
  - Notas Técnicas: notas sobre los algoritmos, o aspectos técnicos del trabajo.
  - Bibliografía: papers, tutoriales, páginas web, etc.

En la figura E.1 puede verse una representación más clara de esta estructura.

Figura E.1: Estructura de ficheros en el DVD

Carpetas y subcarpetas contenidas en el DVD adjunto a la memoria. Fuente: e.p.

Bibliografía

[1] ARC Automatic Reconstruction Conduit *http://www.arc3d.be*, Katholieke Universiteit Leuven, 2006.

[2] ARC 3D Webservice *http://homes.esat.kuleuven.be/∼visit3d/webservice/v2/index.php* Katholieke Universiteit Leuven, 2006.

[3] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz and Richard Szeliski *Building Rome in a Day*, International Conference on Computer Vision, Kyoto, Japan, 2009.

[4] Building Rome in a Day, *http://grail.cs.washington.edu/rome/*, Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz and Richard Szeliski, Washington University, 2009.

[5] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, A.Y. Wu, *An optimal algorithm for approximate nearest neighbor searching fixed dimensions*, J. of the ACM 45, 6, 891-923, 1994.

[6] S. Arya, D.M. Mount, *Approximate Nearest Neighbor Queries in Fixed Dimensions*, Proceedings of the 4th annual ACM-SIAM Symposium on Discrete algorithms, pp 271-280, 1993.

[7] Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool, *SURF: Speeded Up Robust Features*, Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346–359, 2008

[8] M.D.F. Bento, *Unmanned Aerial Vehicles: an Overview* Inside GNSS, pp. 54-61, Jan/Feb 2008.

[9] F. Caballero, L. Merino, J. Ferruz, A. Ollero, *Vision-Based Odometry and SLAM for Medium and High Altitude Flying UAVs*, Journal of Intelligent and Robotic Systems, Nº 1-3, pp. 137-162, 2009.

[10] Pascual Campoy, Juan F. Correa, Iván Mondragón, Carol Martínez, Miguel Olivares, Luis Mejías, Jorge Artieda, *Computer Vision Onboard UAVs for Civilian Tasks*, Journal of Intelligent Robot Systems, 2009.

[11] M. Cazorla, A. Hernández, J. Nieto, E. Nebot, D. Viejo, *Large Scale SLAM with visual features*, 10º Workshop de Agentes Físicos, Septiembre 2009.

[12] Kurt Cornelis, Frank Verbiest and Luc Van Gool, *Drift Detection and Removal for Sequential Structure from Motion Algorithms*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.26, No. 10, October 2004.

[13] Nico Cornelis, Luc Van Gool, *Fast Scale Invariant Feature Detection and Matching on Programmable Graphics Hardware*, Computer Vision and Pattern Recognition Workshops, pp. 1-8, 2008.

[14] Cygwin, *http://www.cygwin.com*, Cygwin Project, 2010.

[15] Cygwin FAQ, *http://www.cygwin.com/faq/*, Cygwin Project, 2010.

[16] A.J. Davison, *SLAM with a Single Camera*, SLAM/CML Workshop at ICRA 2002.

[17] Speeded Up Robust Features, *http://www.vision.ee.ethz.ch/∼surf/*, Eidgenössische Technische Hochschule Zürich(ETHZ), 2010.

[18] David Fernández Llorca, *Sistema de Detección de Peatones Mediante Visión Estereoscópica para la Asistencia a la Conducción*, Tesis Doctoral, Departamento de Electrónica, Universidad de Alcalá, 2008.

[19] Flickr, *http://www.flickr.com*, 2010.

[20] Fulmar, *http://www.aerovision.com*, Aerovisión, 2010.

[21] Patch-based Multi-view Stereo Software, *http://grail.cs.washington.edu/software/pmvs/*, Yasutaka Furukawa, Jean Ponce, 2010.

[22] Clustering Views for Multi-view Stereo, *http://grail.cs.washington.edu/software/cmvs/*, Yasutaka Furukawa, 2010.

[23] Yasutaka Furukawa, Jean Ponce, *Accurate, Dense and Robust Multi-View Stereopsis*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, July 2007.

[24] Google Earth, *http://earth.google.es*, Google, 2010.

[25] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, Second Edition, 2003.

[26] R.I. Hartley, P. Sturm, *Triangulation*, Computer Vision and Image Understanding, Vol. 68, No. 2, pp. 146-157, November 1997.

[27] E. Hygounenc, I.K. Jung, P. Soueres, S. Lacroix, *The Autonomous Blimp Project of LAAS-CNRS: Achievements in Flight Control and Terrain Mapping*, The International Journal of Robotics Research, Vol. 23, No 4-5, pp. 473-511, 2004.

[28] ImageJ, *http://rsbweb.nih.gov/ij/*, Research Services Branch, National Institutes of Health, 2010.

[29] L.F. Johnson, S. Herwitz, S. Dunagan, B. Lobitz, D. Sullivan, R. Slye, *Collection of Ultra High Spatial and Spectral Resolution Image Data over California Vineyards with a Small UAV*, Proceedings International Symposium on Remote Sensing of Environment, 2003.

[30] R. Kumar, H. Sawhney, S. Samarasekera, S. Hsu, H. Tao, Y. Guo, K. Hanna, A. Pope, R. Wildes, D. Hirvonen, M. Hansen, P. Burt, *Aerial video Surveillance and Exploitation*, Proceedings of the IEEE, Vol. 89, Nº 10, pp. 1518-1539, October 2001.

[31] Hongdong Li, *A simple solution to the Six Point Two View Focal Length Problem*, ECCV, pp. 200-213, 2006.

[32] A Generic Sparse Bundle Adjustment C/C++ Package Based on the Levenberg-Marquardt Algorithm, *http://www.ics.forth.gr/∼lourakis/sba/*, Manolis Lourakis, FORTH - Institute of Computer Science, January 2010.

[33] Manolis I. A. Lourakis, Antonis A. Argyros, *SBA: A Software Package for Generic Sparse Bundle Adjustment*, ACM Transactions on Mathematical Software, Vol.36, No.1, Article 2, March 2009.

[34] Manolis I. A. Lourakis, Antonis A. Argyros, *The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm*, FORTH-ICS, August 2004.

[35] SIFT Keypoint detector, *http://www.cs.ubc.ca/∼lowe/keypoints/*, David G. Lowe , 2009.

[36] David G. Lowe, *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision, pp. 91-110, 2004.

[37] David G. Lowe, *Local feature view clustering for 3D object recognition*, IEEE Conference on Computer Vision and Pattern Recognition, Kauai, Hawaii, pp. 682-688, December 2001.

[38] David G. Lowe, *Object recognition from local scale-invariant features*, International Conference on Computer Vision, Corfu, Greece, pp. 1150-1157, September 1999.

[39] K. Madsen, H.B. Nielsen, O. Tingleff, *Methods for Non-linear Least Squares Problems*, Informatics and Mathematical Modelling, Technical University of Denmark, 2nd Edition, April 2004.

[40] Larry Matthies, Mark Maimone, Andrew Johnson, Yang Cheng, Reg Willson, Carlos Villalpando, Steve Goldberg, Andres Huertas, Andrew Stein, and Anelia Angelova, *Computer Vision on Mars*, International Journal of Computer Vision (IJCV), 2007.

[41] Luis Merino, Johan Wiklund, Fernando Caballero, Anders Moe, José Ramiro Martínez de Dios, Per-Erik Forssén, Klas Nordberg, Aníbal Ollero, *Vision-Based Multi-UAV Position Estimation*, IEEE Robotics and Automation Magazine, pp. 53-62, September 2006.

[42] Luis Merino, Fernando Caballero, J.R. Martínez-de Dios, J. Ferruz, A. Ollero, *A cooperative perception system for multiple UAVs: Application to automatic detection of forest fires*, Journal of Field Robotics, Volume 23 Issue 3-4, pp. 165-184, March-April 2006.

[43] Meshlab, *http://meshlab.sourceforge.net/* , Meshlab Project, Visual Computing Lab, Italian National Research Council, 2010.

[44] Photosynth, *http://livelabs.com/photosynth/*, Microsoft Research, 2009.

[45] Sigpac, *http://sigpac.mapa.es/fega/visor/*, Ministerio de Medio Ambiente y Medio Rural y Marino, 2010.

[46] David M. Mount, Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, Maryland, *ANN Programming Manual*, Version 1.1, 2010.

[47] David M. Mount, Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, Maryland, *http://www.cs.umd.edu/∼mount/ANN*, Web de ANN, 2010.

[48] UAV Helios, *http://www.nasa.gov/centers/dryden/history/pastprojects/Helios/index.html*, NASA, 2010.

[49] GloPac 2010 *http://www.nasa.gov/centers/dryden/research/GloPac/index.html*, NASA, 2010.

[50] D. Nistér, *An Efficient Solution to the Five-Point Relative Pose Problem*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(6):756-770, 2004.

[51] Web de OpenCV, *http://opencv.willowgarage.com/wiki/*, OpenCV, 2010.

[52] VGG MultiView Compute Library, *http://www.robots.ox.ac.uk/∼vgg/hzbook/code/*, Oxford Robotics Research Group, 2010.

[53] Picasa, *http://picasaweb.google.es*, 2010.

[54] Marc Pollefeys, *Tutorial on 3D Modelling from Images*,Katholieke Universiteit Leuven, Bélgica, June 2000.

[55] Ananth Ranganathan, *The Levenberg-Marquardt Algorithm*, June 2004.

[56] Sivakumar Rathinam, ZuWhan Kim, Raja Sengupta, *Vision-Based Following of Structures Using an Unmanned Aerial Vehicle (UAV)* Institute of Transportation Studies, University of California at Berkeley, Research Report UCB-ITS-RR-2006-1, March 2006.

[57] Structure From Motion Toolkit for Matlab, *http://vision.ucsd.edu/∼vrabaud/toolbox/doc/*, Vincent Rabaud, 2010.

[58] RQ-4 Global Hawk, *http://www.as.northropgrumman.com/products/ghrq4a/index.html*, Northrop Grumman, 2010.

[59] M. Scaioni, L. Barazetti, R. Brumana, B. Cuca, F. Fassi, F. Prandi, *RC-Heli and Structure and Motion techniques for the 3-D Reconstruction of a Milan Dome spire*, 3rd International Workshop 3D-ARCH, 2009.

[60] Web de Scanalyze, *http://graphics.stanford.edu/software/scanalyze/*, Stanford Computer Graphics Laboratory, 2010.

[61] S. Se, P. Firoozfam, N. Goldstein, L. Wu, M. Dutkiewicz, P. Pace, P. Naud, *Automated UAV-based mapping for airborne reconaissance and video exploitation*, Airborne Intelligence, Surveillance, Reconnaisance (ISR) Systems and Applications VI, Proceedings of the SPIE, Vol 7307, 2009.

[62] Stephen Se, Ho-Kong Ng, Piotr Jasiobedzki, Tai-Jing Moyung, *Vision Based Modelling and Localization for Planetary Exploration Rovers*, 55th International Astronautical Congress, Vancouver, Canada, 2004.

[63] Steve Seitz, Brian Curless, James Diebel, Daniel Scharstein, Richard Szeliski, *A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms*, CVPR, vol. 1, pages 519-526, 2006.

[64] A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms, Middlebury University, *http://vision.middlebury.edu/mview/*, Steve Seitz, Brian Curless, James Diebel, Daniel Scharstein, Richard Szeliski, 2006.

[65] Sudipta N. Sinha, Jan-Michel Frahm, Marc Pollefeys, Yakup Genc, *GPU-based Video Feature Tracking and Matching*, Workshop on Edge Computing Using New Commodity Architectures, May 2006.

[66] Silver Fox, *http://www.acrtucson.com/UAV/silverfox/index.htm*, BAE Systems, 2010.

[67] Web de Bundler, *http://phototour.cs.washington.edu/*, Noah Snavely, Feb. 2009.

[68] Noah Snavely, *Scene Reconstruction and Visualization from Internet Photo Collections*, University of Washington, 2008.

[69] Noah Snavely, Rahul Garg, Steven M. Seitz, Richard Szeliski, *Finding Paths through the World's Photos*, SIGGRAPH Conf. Proc., 2008.

[70] Noah Snavely, Steven M. Seitz, Richard Szeliski, *Modeling the World from Internet Photo Collections*, International Journal of Computer Vision, 2007.

[71] Noah Snavely, Steven M. Seitz, Richard Szeliski, *Photo Tourism: Exploring image collections in 3D*, SIGGRAPH Conf. Proc., 2006.

[72] H. Stewénius, C. Engels, D. Nistér, *Recent Developments on Direct Relative Orientation*, ISPRS Journal of Photogrammetry and Remote Sensing, Vol. 60, Issue 4, pp. 284-294. June 2006.

[73] Structure and Motion Toolkit in Matlab, *http://www.mathworks.com/matlabcentral/fileexchange/4576*, Philip Torr, March 2004.

[74] Structure and Motion Toolkit in Matlab, *http://cms.brookes.ac.uk/staff/PhilipTorr/Code/code.htm*, Philip Torr, 2010.

[75] UAVNET *European Civil Unmanned Air Vehicle Roadmap. Volume 1 Overview* December 2005.

[76] UAVNET *European Civil Unmanned Air Vehicle Roadmap. Volume 2 Action Plan* December 2005.

[77] UAVNET *European Civil Unmanned Air Vehicle Roadmap. Volume 3 Strategic Research Agenda* December 2005.

[78] Web de UNVEX'10, *http://www.unvex10.es*, UNVEX'10, 2010.

[79] Varios autores, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, Second Edition, 1992.

[80] Maarten Vergauwen, Luc Van Gool, *Web-based 3D reconstruction service*, Machine Vision Applications, 17, pp. 411-426, 2006.

[81] Web de VLC, *http://www.videolan.org/vlc/*, VideoLAN Project, 2010.

[82] Web de Virtual Dub, *http://www.virtualdub.org/*, Virtual Dub Group, 2010.

[83] Photocity, *http://photocity.cs.washington.edu*, Washington University, 2010.

[84] 2d3 *http://www.2d3.com* Web 2d3, 2010.