Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Aplicación de Técnicas de Generación de Mosaicos a la Inspección de Infraestructuras Lineales mediante UAS

Autor: Raúl Tapia López

Tutor: José Ramiro Martínez de Dios

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Aplicación de Técnicas de Generación de Mosaicos a la Inspección de Infraestructuras Lineales mediante UAS

Autor:
Raúl Tapia López

Tutor:
José Ramiro Martínez de Dios
Catedrático de Universidad

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Trabajo Fin de Grado: Aplicación de Técnicas de Generación de Mosaicos a la Inspección de Infraestructuras Lineales mediante UAS

Autor: Raúl Tapia López
Tutor: José Ramiro Martínez de Dios

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

A mis padres, por enseñarme a aprender;
a mi hermana, con quien he aprendido a enseñar,
y a Sara, por permitirme aprender a su lado.

A Ramiro, por toda su dedicación,
y a Julio, Juanma, María y Víctor, por su ayuda.

Resumen

La inspección y mantenimiento de infraestructuras es uno de los ámbitos en los que el uso de la robótica ha experimentado un notable crecimiento durante los últimos años. Este trabajo de fin de grado presenta un método de generación de mosaicos para aplicaciones de inspección de infraestructuras lineales a partir de imágenes capturadas por sistemas aéreos no tripulados.

El algoritmo propuesto comprende tres etapas: detección y descripción de características, asociación de características y cálculo de la transformación que relaciona las imágenes. El método ha sido diseñado utilizando hipótesis derivadas de las condiciones de vuelo para reducir su coste computacional, sin sacrificar por ello un comportamiento eficiente y robusto. La estimación del desplazamiento entre imágenes consecutivas permite seleccionar zonas de interés para evitar detecciones y asociaciones en toda la imagen, reduciendo el tiempo requerido y simplificando la optimización para el cálculo de la transformación.

Abstract

Inspection and maintenance of infrastructures is one of the areas where robotics has experienced a remarkable development during the last years. This Degree Final Project presents a mosaicking method for linear infrastructures inspection applications from images taken by unmanned aerial systems.

The proposed algorithm comprises three stages: detection and description of features, feature matching and calculation of the transformation that relates the images. The method has been designed using hypotheses derived from the flight conditions in order to reduce their computational cost, without sacrificing an efficient and robust behavior. The estimation of the displacement between consecutive images allows select regions of interest to avoid detections and associations in the whole image, reducing the required time and simplifying the optimization for calculation of the transformation.

Índice Abreviado

Índice

Notación

$\mathbb{R}$	Conjunto de los números reales
$\exists$	Existe
$\nexists$	No existe
$\mid$	Tal que
$\in$	Pertenece al conjunto
$\cap$	Intersección de conjuntos
$\cup$	Unión de conjuntos
$\otimes$	Disyunción exclusiva de conjuntos
$\lvert x \rvert$	Valor absoluto de x
$\sum$	Sumatorio
$\prod$	Productorio
$\lVert \mathbf{v} \rVert$	Norma del vector $\mathbf{v}$
$\det(\mathbf{A})$	Determinante de la matriz $\mathbf{A}$
$\mathrm{tr}(\mathbf{A})$	Traza de la matriz $\mathbf{A}$
$\mathbf{A}^\top$	Transpuesta de la matriz $\mathbf{A}$
$\mathbf{A}^{-1}$	Inversa de la matriz $\mathbf{A}$
$\mathbf{I}$	Matriz identidad
e	Número de Euler
cos	Función coseno
sin	Función seno
tan	Función tangente
arctan	Función arcotangente
Δ	Incremento
$\frac{\partial y}{\partial x}$	Derivada parcial de y respecto a x
μ_x	Media de x
σ_x	Desviación típica o estándar de x
σ_x^2	Varianza de x
$\mathcal{N}(\mu_x, \sigma_x)$	Distribución gaussiana de media μ_x y desviación típica σ_x

1 Introducción

En los últimos años, el uso de la robótica en tareas de inspección y mantenimiento de infraestructuras y entornos industriales ha crecido de forma considerable. El empleo de robots, sean o no teleoperados, permite realizar inspecciones de una manera mucho más rápida y segura, evitando la intervención humana en zonas de alta peligrosidad o de difícil acceso. En consecuencia, se reducen las horas de personal necesarias para operaciones de este tipo, lo que repercute directamente en una reducción del coste de la inspección.

El presente trabajo de fin de grado presenta un método de generación de mosaicos para su aplicación a la inspección de infraestructuras mediante el uso de UAS (*unmanned aircraft systems* o sistemas aéreos no tripulados). La generación de mosaicos es una técnica que permite combinar varias imágenes para formar una de mayor tamaño, centralizándose así toda la información recogida en una única imagen final donde poder visualizar todo el espacio fotografiado.

En concreto, el algoritmo propuesto en este documento se centra en el estudio de infraestructuras de naturaleza lineal, es decir, instalaciones que ocupen un espacio en el que una de las dimensiones sea predominante respecto a las otras. Por lo general, estas infraestructuras se extienden durante varios kilómetros. Ejemplos de infraestructuras lineales son los gasoductos, los oleoductos, las autopistas y autovías, las redes ferroviarias o el tendido eléctrico.

Figura 1.1 Oleoducto, ejemplo de estructura lineal.

1.1 Objetivo

El objetivo del método propuesto en este documento es generar un mosaico a partir de la unión de imágenes capturadas por un robot aéreo. El algoritmo diseñado será utilizado en operaciones de inspección de gasoductos.

Todas las imágenes que compongan el mosaico deben estar georeferenciadas para que, en caso de detectar algún fallo en la instalación, el operario encargado de la inspección pueda localizar el lugar donde efectuar la reparación. Además, el robot lleva incorporado un sensor para medir la concentración de gases hidrocarburos en el aire. Con el fin de detectar posibles fugas, se deben incluir en el mosaico indicadores que permitan visualizar las medidas obtenidas.

1.2 Contribución

En la literatura asociada a la generación de mosaicos existe un elevado número de algoritmos diferentes, sin embargo, la mayor parte de estos posee un carácter general y son pocos los métodos desarrollados con un objetivo específico. Es por esto que la mayoría de estas técnicas conlleva un coste computacional elevado en comparación con el coste temporal requerido por las operaciones de inspección, las cuales deben tener, por lo general, un funcionamiento en condiciones cercanas al tiempo real.

El método que se presenta en este documento ha sido diseñado para poseer un comportamiento eficiente y robusto. Su principal fin es el de generar resultados en un tiempo relativamente corto en comparación con el de muchas de las técnicas existentes. Para lograr este objetivo, la implementación del algoritmo realizará una serie de particularizaciones al caso concreto presentado anteriormente basadas en las hipótesis que se detallan a continuación.

1.2.1 Hipótesis

Las condiciones de vuelo y demás hipótesis empleadas para particularizar el método de generación de mosaicos son las siguientes:

- Se realizarán inspecciones de estructuras lineales, por lo que una de las dos dimensiones del mosaico será mucho mayor que la otra.
- El vuelo se realizará a una distancia del suelo lo suficientemente elevada como para poder despreciar las diferencias de altura de los objetos contenidos en las imágenes.
- El vuelo será a altura constante, por lo que no se esperan grandes variaciones de escala.
- El robot aéreo se desplazará con *pitch* y *roll* constantes, existiendo variaciones únicamente del *yaw*, por lo que las imágenes adquiridas no presentarán cambios significativos de perspectiva. En la Figura 1.2 se detalla la notación empleada para designar los ángulos de giro.
- No existe ningún requisito relativo a la velocidad de desplazamiento o a la tasa de adquisición de imágenes, aunque deberá garantizarse que dos imágenes consecutivas posean un número de puntos en común suficiente como para poder realizar la unión.

Con todo lo anterior, la trayectoria que describirá el robot puede ser simplificada como un movimiento coplanario paralelo al suelo. Los grados de libertad que definen este tipo de movimiento (Figura 1.3) son dos traslaciones y una rotación.

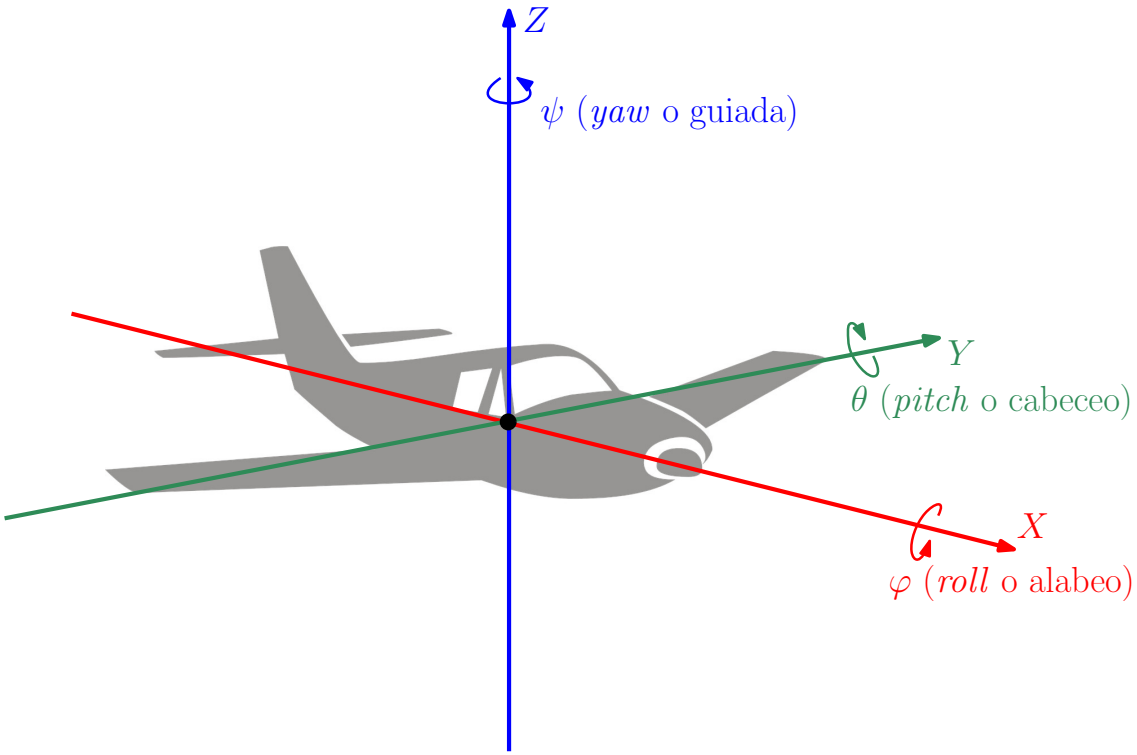

Figura 1.2 Nomenclatura empleada para ángulos de giro.

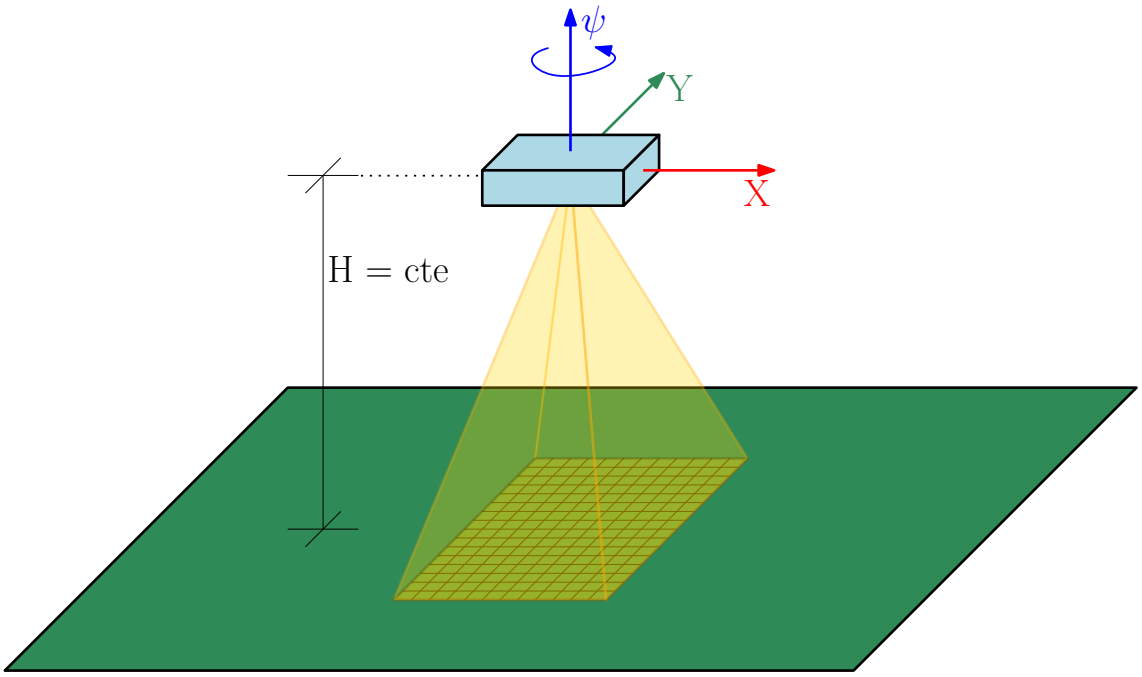

Figura 1.3 Grados de libertad del movimiento simplificado del robot.

1.3 Marco de realización

Este trabajo de fin de grado se enmarca dentro del proyecto INSPECTOR (*INdustrial inSPECtion and mainTenance of complex OR unattended facilities*, en español, *Inspección y mantenimiento industrial de instalaciones complejas o desatendidas*) desarrollado bajo el Programa Estratégico de Consorcios de Investigación Empresarial Nacional (CIEN) del Centro para el Desarrollo Tecnológico Industrial (CDTI), en el que participa el Grupo de Robótica, Visión y Control (GRVC) de la Universidad de Sevilla. El proyecto INSPECTOR busca la "transference to industry of several technologies, tools and research regarding to the automation of inspection processes and industrial maintenance. Particularly, great improvements in efficiency are expected on challenging scenarios (dangerous or difficult to access)" [transferencia a la industria de varias tecnologías, herramientas e investigación relacionadas con la automatización de los procesos de inspección y mantenimiento industrial. En particular, se esperan grandes mejoras en eficiencia en escenarios que puedan suponer un reto (peligrosos o de difícil acceso)] [3].

La inspección se realizará con un robot aéreo equipado con una cámara de alta resolución orientada hacia el suelo, un GPS y un sensor para medir la concentración de gases hidrocarburos en el aire y poder detectar fugas en el gasoducto. Para validar el algoritmo implementado se ha utilizado un conjunto de imágenes adquiridas mediante un robot aéreo DJI Phantom 4 Pro [1].

Versiones preliminares del algoritmo han sido implementadas utilizando el lenguaje de programación Python para el intérprete Python 3.7.3 con la ayuda de las librerías OpenCV 4.1.0 y NumPy 1.10.2. La versión final del algoritmo ha sido implementada utilizando el lenguaje de programación C++ versión 11 para el compilador GCC 7.4.0 con la ayuda de la librería OpenCV 4.1.0 y bajo el framework ROS Kinetic Kame para Ubuntu 16.04 LTS.

1.3.1 ROS

ROS (*Robot Operative System*) [5] es un entorno de trabajo o *framework* para el desarrollo software desarrollado inicialmente en 2007 por el *Stanford Artificial Intelligence Laboratory* y por *Willow Garage* desde 2008. ROS provee los servicios básicos de un sistema operativo y esta basado en una estructura de nodos que pueden recibir y enviar mensajes entre si. Actualmente, es uno de los softwares más empleados a nivel comercial e investigador en la robótica.

Figura 1.4 Logotipo de ROS. Fuente: [5].

1.3.2 OpenCV

OpenCV [4] es una librería desarrollada desde 1999 por Intel utilizada en aplicaciones relacionadas con la visión artificial. Se trata de una librería *open source* y multiplataforma pensada para ofrecer un gran número de funciones relacionadas con la percepción artificial. Permite ser utilizada con C++ y Python.

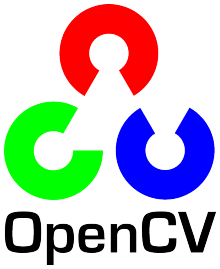

Figura 1.5 Logotipo de OpenCV. Fuente: [4].

1.4 Estructura

Este trabajo de fin de grado se compone de siete capítulos y un anexo. El Capítulo 2 presenta el estado del arte de las técnicas de generación de mosaicos. En el Capítulo 3 se encuentra un esquema general del método propuesto. Los Capítulos 4, 5 y 6 detallan los tres módulos que componen el método: extracción de características, correlación de características y cálculo de la transformación homogénea que relaciona las imágenes. El Capítulo 7 realiza un análisis de los resultados obtenidos al aplicar el método a un conjunto de imágenes para su validación. Por último, en el Capítulo 8, se presentan las conclusiones extraídas durante la realización de este proyecto y sus posibles desarrollos futuros. El Anexo A describe la construcción de un sistema de referencia cartesiano a partir de las medidas ofrecidas por el GPS.

2 Estado del Arte

2.1 Introducción

En este capítulo se describirán y compararán las distintas técnicas de *mosaicking* empleadas actualmente, abarcando con mayor profundidad los conocidos como métodos indirectos.

La generación de mosaicos o *mosaicking* es una técnica de procesamiento de imagen que permite combinar varias imágenes digitales para formar una de mayor tamaño. Se trata de un caso especial de corrección geométrica en el que una imagen se ajusta a otra con la que se relaciona mediante una transformación homogénea [23].

Actualmente, las técnicas de mosaico poseen gran interés por parte de la comunidad investigadora por sus posibles aplicaciones en sistemas que interaccionan con el mundo real.

En la literatura asociada al *mosaicking* puede encontrarse un gran número de soluciones [21] [53] [58] [6] [38], sin embargo, este campo de investigación continúa creciendo con el objetivo de obtener algoritmos cada vez más eficientes. Además, puesto que los sistemas de adquisición de imágenes y los sistemas de posicionamiento no dejan de evolucionar día tras día con más y mejores prestaciones, son numerosos los distintos tipos de datos a los que enfrentarse.

Figura 2.1 Superposición de dos imágenes.

2.2 Técnicas de *mosaicking*

2.2.1 Una visión general de las técnicas de mosaico de imágenes

Pese al creciente número de técnicas, pueden distinguirse dos categorías principales:

- Técnicas sin extracción de *features* (también conocidas como métodos directos o métodos basados en área).
- Técnicas con extracción de *features* (también conocidas como métodos indirectos).

Técnicas sin extracción de *features*

Dentro de las técnicas sin extracción de *features*, predominan aquellas en las que el mosaico es generado a partir de la estimación de los parámetros de posición, obtenida minimizando el error cuadrático de la diferencia de intensidades en las zonas donde las imágenes deben superponerse (Figura 2.2). La calidad del mosaico resultante en este tipo de técnicas depende directamente del resultado de la minimización, siendo en ocasiones un gran problema su sensibilidad ante mínimos locales. Además, el hecho de trabajar con todo el conjunto de puntos de la imagen hace que sean métodos inestables frente a la presencia de objetos móviles o los cambios de luz.

También se incluyen las técnicas en las que las imágenes que forman el mosaico se reconstruyen sobre un cilindro o una esfera para conformar un entorno tridimensional, que será proyectado para obtener la imagen final (Figura 2.3). El resultado entregado por este tipo de algoritmos depende en gran medida de la calidad de las imágenes con las que trabaja.

La principal ventaja de las técnicas que no requieren *features* es la redundancia de información (tienen en cuenta todos los puntos de las imágenes analizadas), mientras que la principal desventaja es la necesidad de un gran porcentaje de puntos en común.

En [60] se llega a la conclusión de que "the motion between two successive images has to be small and an initial estimation of the motion needs to be provided" [el movimiento entre dos imágenes consecutivas tiene que ser pequeño y es necesaria una estimación inicial de ese movimiento] por lo que "this category is generally restricted to the case of video sequences" [esta categoría se restringe, por lo general, al caso de secuencias de video].

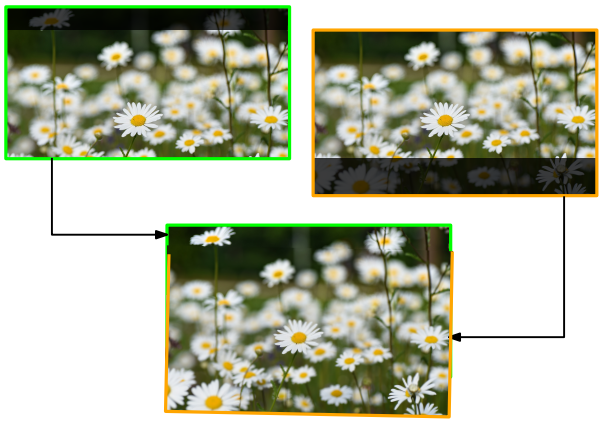

Figura 2.2 Mosaicking mediante estimación de parámetros de posición.

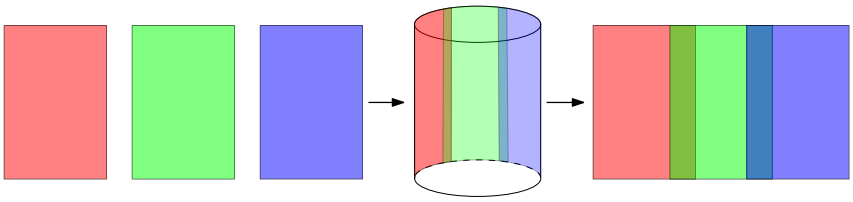

Figura 2.3 Mosaicking mediante proyección en cilindro.

Técnicas con extracción de *features*

Los métodos basados en *features* (Figura 2.4) comprenden tres etapas:

1. Extracción de *features* en las áreas comunes a ambas imágenes.
2. Unión de *features* (*feature matching*).
3. Cálculo de la homografía que relaciona las imágenes.

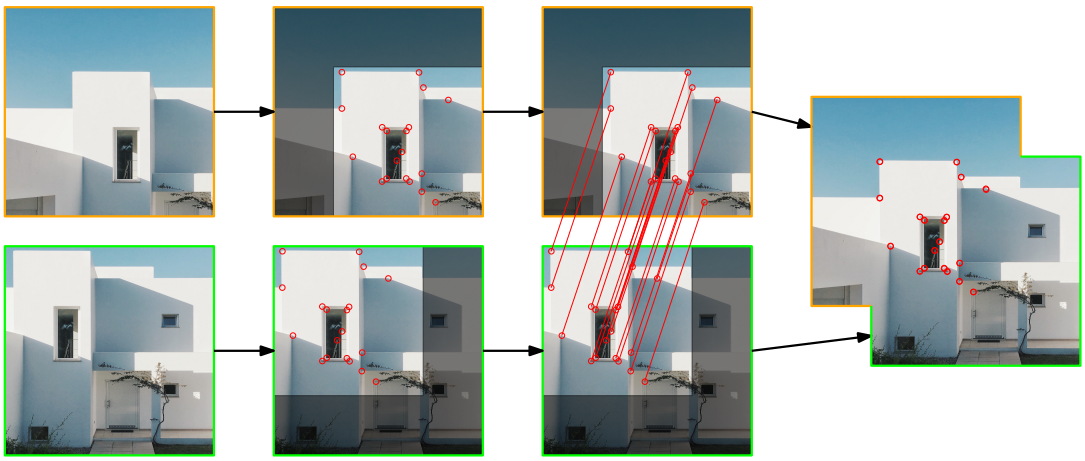

Figura 2.4 Mosaicking mediante técnica basada en *features*.

Los algoritmos que implementan métodos no directos pueden clasificarse según los tipos de transformaciones homogéneas que aplican a las imágenes:

- Algoritmos para imágenes relacionadas por traslación o rotación.
- Algoritmos para imágenes relacionadas por traslación y rotación.
- Algoritmos para imágenes relacionadas por cualquier transformación homogénea (traslación, rotación, perspectiva y/o escalado).

Como es de esperar, la diferencia entre estas tres categorías reside en la complejidad de los algoritmos y en su tiempo de ejecución, siendo necesario estudiar las características de las imágenes antes de elegir el método que más se adecúe a cada caso.

Al trabajar únicamente con algunos puntos de las imágenes, estos métodos son, por lo general, más eficientes que los métodos directos y más estables frente a variaciones de luz u objetos móviles. Tal y como se concluye en [60], son métodos cuya principal ventaja es que "they can handle small overlapping regions" [pueden manejar pequeñas regiones de superposición] por lo que se suelen emplear para conjuntos de imágenes con tasas de adquisición bajas. El principal inconveniente respecto a los métodos sin extracción de *features* es el mayor coste computacional.

Resumen de las Técnicas Mencionadas

A modo de resumen, se presenta la Figura 2.5.

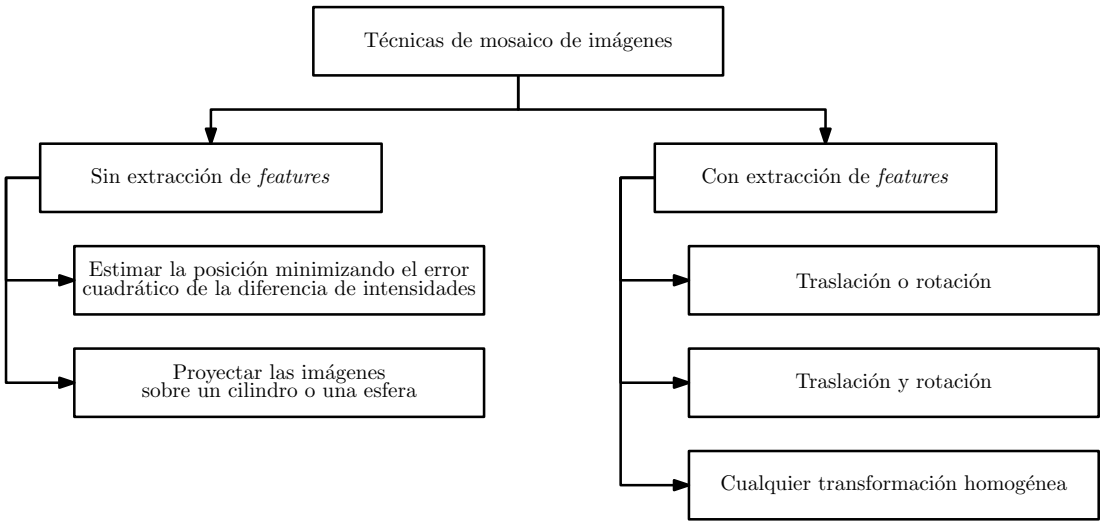

Figura 2.5 Clasificación general de las técnicas de mosaico de imágenes.

2.2.2 Aplicaciones de las técnicas de mosaico de imágenes

Las técnicas de mosaico de imágenes son un campo de investigación que no deja de crecer, abarcando disciplinas tales como la robótica, la fotografía o la biomedicina.

Uno de los campos en los que el *mosaicking* encuentra muchas aplicaciones es el de la robótica aérea, pues permite inspeccionar grandes extensiones o zonas de difícil acceso [7] [34] [27] [35] [31] [14] [32] [57]. Ejemplos de aplicación de las técnicas de mosaico a la inspección y mantenimiento son [7], donde se capturan imágenes de un campo de paneles fotovoltaicos y se procesan en tiempo real; [31], donde se inspeccionan espacios afectados por desastres naturales tales como inundaciones; o [57], donde se utilizan los mosaicos para conseguir una respuesta rápida ante incendios en los bosques.

En el ámbito de la medicina encontramos, entre otros muchos, estudios relacionados con la fetología [51] [15], la cardiología [54] y demás campos [43] [52]. Otro ejemplo del uso de estas técnicas relacionadas con la biología y la ecología es [33], donde los mosaicos permiten a los científicos determinar ciertas características de los arrecifes de coral.

Las técnicas de *mosaicking* también son empleadas para fotografía panorámica [41] [20] [11], video vigilancia [9] o escaneo de documentos [48].

Figura 2.6 Inspección de planta de paneles solares. Fuente: [7].

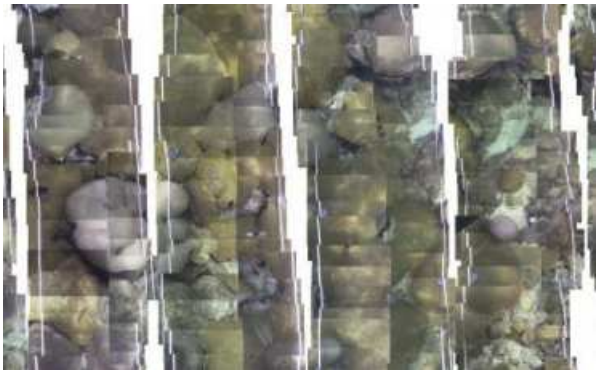

Figura 2.7 Mosaico de un arrecife de coral. Fuente: [33].

Figura 2.8 Panorámica construida con 32 imágenes del Hiranandani Gardens, Mumbai, India. Fuente: [11].

2.3 Técnicas de *mosaicking* basadas en la extracción de *features*

Se conocen como técnicas basadas en la extracción de *features* a todas aquellas técnicas de mosaico que calculan la transformación homogénea que relaciona dos imágenes a partir de la unión de los *features* de ambas.

2.3.1 Extracción de *features*

En el ámbito del procesamiento de imágenes, se conoce como *feature* o característica a una parte de la imagen cuya información es útil para resolver una tarea. Existen distintos tipos de *features*, siendo los más empleados los puntos, los bordes y las regiones (Figura 2.9).

La obtención de los *features* de una imagen se realiza mediante métodos de extracción. Estos métodos están formados por un algoritmo de detección (*corner detector*, *edge detector*, *ridge detector*, *blob detector*, ...) [40] [16] [44] [37] [45], encargado de obtener sus localizaciones, y un algoritmo de descripción [22] [39] [59], encargado de extraer información de interés de los elementos vecinos.

Se debe tener en cuenta que la eficiencia del algoritmo final dependerá en gran medida del tipo de *feature* empleado y de la eficiencia de la extracción.

Figura 2.9 Tipos de *features*.

Harris Corner Detector

El algoritmo *Harris Corner Detector* (HCD) es un detector de *features* ampliamente utilizado en el procesamiento de imágenes desarrollado por Chris Harris y Mike Stephens [19] a partir del *Moravec's Corner Detector* [17].

El detector de esquinas de Harris se encarga de determinar qué ventanas de una imagen producen grandes variaciones cuando se mueven en cualquier dirección. Se trata de un algoritmo invariante ante la rotación y el cambio de perspectiva, pero no ante el escalado [49] [47] [40].

Sea I una imagen en escala de grises y w una ventana de esa imagen, el error cuadrático entre w y una región del mismo tamaño desplazada $(\Delta x,\Delta y)$ viene dado por la expresión:

$$E(w) = \sum_{(x,y)\in w} (I(x,y) - I(x+\Delta x, y+\Delta y))^2 \tag{2.1}$$

Que puede ser reescrita utilizando el desarrollo en serie de Taylor de $I(x+\Delta x, y+\Delta y)$ como:

$$E(w) = \sum_{(x,y)\in w} (\frac{\partial I}{\partial x}(x,y)\Delta x + \frac{\partial I}{\partial y}(x,y)\Delta y)^2 \tag{2.2}$$

Cuya forma matricial es:

$$E(w) = \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \sum_{(x,y)\in w} \begin{bmatrix} (\frac{\partial I}{\partial x}(x,y))^2 & \frac{\partial I}{\partial x}(x,y)\frac{\partial I}{\partial y}(x,y) \\ \frac{\partial I}{\partial x}(x,y)\frac{\partial I}{\partial y}(x,y) & (\frac{\partial I}{\partial y}(x,y))^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{2.3}$$

A partir de la Ecuación 2.3, puede definirse el tensor M:

$$M(w) = \sum_{(x,y)\in w} \begin{bmatrix} (\frac{\partial I}{\partial x}(x,y))^2 & \frac{\partial I}{\partial x}(x,y)\frac{\partial I}{\partial y}(x,y) \\ \frac{\partial I}{\partial x}(x,y)\frac{\partial I}{\partial y}(x,y) & (\frac{\partial I}{\partial y}(x,y))^2 \end{bmatrix} \tag{2.4}$$

Se buscan aquellas ventanas cuyo desplazamiento provoque un gran error, por lo que a cada una se le asocia un número R:

$$R(w) = \det(M(w)) - k(\mathrm{tr}(M(w)))^2 = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2) \tag{2.5}$$

Donde la constante k debe ser determinada de forma empírica.

Toda aquella ventana que posea un valor de R mayor que un valor umbral, contendrá una esquina. Para obtener las esquinas, suele calcularse el máximo local de la Ecuación 2.1 dentro de cada región seleccionada.

Atendiendo al desarrollo anterior, puede definirse el algoritmo de *Harris Corner Detector* de la siguiente forma:

Código 2.1 Algoritmo Harris Corner Detector.

```
PARA CADA w EN I
  Calcular derivadas espaciales
  Calcular el tensor M
  Calcular el valor R
  SI R > umbral
    w contiene una esquina
  FIN SI
FIN PARA CADA
PARA CADA w que contenga una esquina
  Calcular valor que maximice el error cuadrático dentro de w
FIN PARA CADA
```

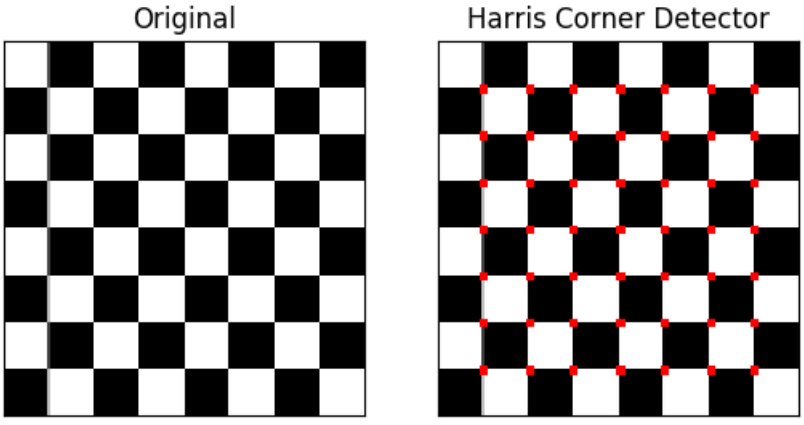

Figura 2.10 Resultado de implementar el algoritmo *Harris Corner Detector*.

Pese a ser un detector muy utilizado, presenta ciertos inconvenientes al no ser invariante ante escalado, por lo que muchas aplicaciones harán uso de detectores más complejos como pueden ser SIFT, SURF u ORB [26] [50] [24].

SIFT

SIFT (*Scale Invariant Feature Transform*) es un detector y descriptor de *features* invariante respecto a la rotación, la perspectiva, el escalado y los cambios de intensidad de las imágenes a analizar [36]. El algoritmo SIFT se compone de cuatro pasos:

1. Estimación de extremos en la escala-espacio aplicando la diferencia gaussiana, es decir, buscar máximos locales a lo largo del espacio para distintas escalas (Figura 2.19).
2. Localización de puntos de interés (*keypoints*), descartando aquellos puntos cuyo valor esté por debajo de un valor umbral.
3. Asignación de una orientación a cada punto de interés, garantizándose así la invarianza respecto a la rotación.
4. Generación del descriptor para cada punto de interés.

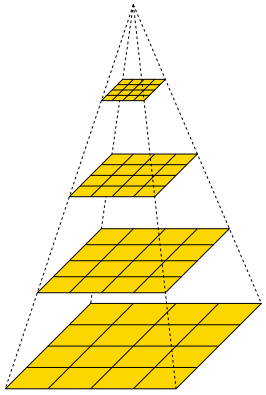

Figura 2.11 Generación de escala-espacio.

SURF

SURF (*Speed Up Robust Features*) es un detector y descriptor de *features* basado en el algoritmo SIFT [10]. Las principales diferencias con su predecesor son:

- En general, provee mejores resultados en menor tiempo.
- Posee un comportamiento más robusto.
- Aproxima la diferencia gaussiana que utiliza SIFT mediante la convolución con un filtro cuadrado.
- Utiliza un detector BLOB (*Binary Large Object*) basado en el uso de la matriz Hessiana, reduciendo así el tiempo de computación.
- No almacena información de escala y orientación para cada punto (solo existe un único punto de interés por cada posición (x,y)).

ORB

ORB (*Oriented FAST and Rotated BRIEF*) es un detector y descriptor de *features* nacido de la unión del detector FAST (*Features from Accelerated Segment Test*) y el descriptor BRIEF (*Binary Robust Independent Elementary Features*). Es un método invariante respecto a la rotación, la perspectiva, el escalado y los cambios de intensidad, así como resistente al ruido.

Se trata de un algoritmo presentado en [46] cuyo objetivo es proporcionar una alternativa rápida y eficiente a SIFT y SURF. Tanto FAST como BRIEF son algoritmos incapaces de trabajar con imágenes relacionadas mediante una rotación, por lo que [46] incorpora una componente de orientación para el detector (oFAST, *Oriented FAST*) y una expansión del descriptor para tener en cuenta la rotación (rBRIEF, *Rotated BRIEF*).

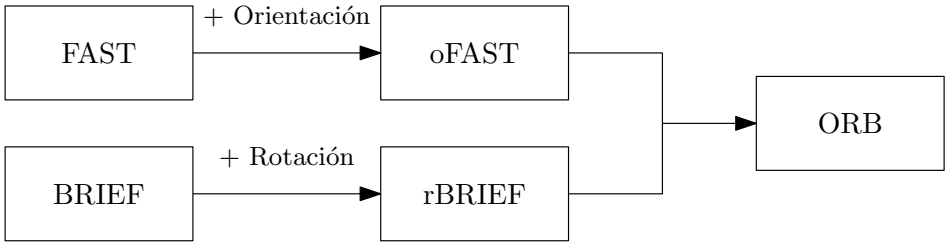

Figura 2.12 Composición del algoritmo ORB.

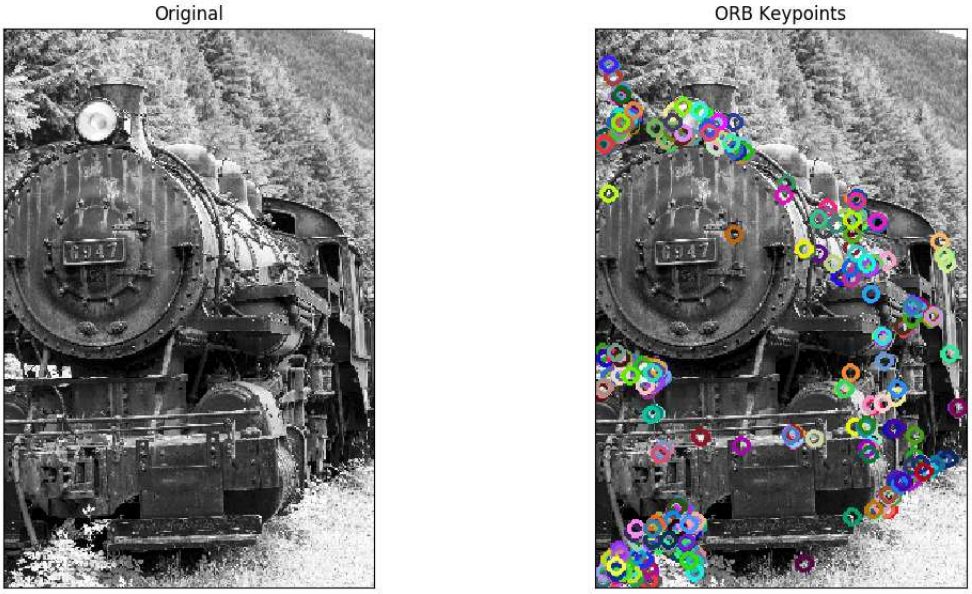

Figura 2.13 Resultado de implementar el algoritmo ORB.

Comparativa entre SIFT, SURF y ORB

Todos los datos presentados a continuación han sido extraídos del documento [26].

Tabla 2.1 Comparativa para imágenes con variación de intensidad.

Algoritmo	Tiempo (s)	Features 1	Features 2	Matches	Porcentaje de matches (%)
SIFT	0.13	248	229	183	76.7
SURF	0.04	162	166	119	72.6
ORB	0.03	261	267	168	63.6

Tabla 2.2 Comparativa para imágenes relacionadas por una rotación.

Algoritmo	Tiempo (s)	Features 1	Features 2	Matches	Porcentaje de matches (%)
SIFT	0.16	248	260	166	65.4
SURF	0.03	162	271	110	50.8
ORB	0.03	261	423	158	46.2

Tabla 2.3 Comparativa para imágenes con variación de escala.

Algoritmo	Tiempo (s)	Features 1	Features 2	Matches	Porcentaje de matches (%)
SIFT	0.25	248	1210	232	31.8
SURF	0.08	162	581	136	36.6
ORB	0.02	261	471	181	49.5

Tabla 2.4 Comparativa para imágenes con variación de perspectiva.

Algoritmo	Tiempo (s)	Features 1	Features 2	Matches	Porcentaje de matches (%)
SIFT	0.13	248	229	150	62.9
SURF	0.05	162	214	111	59.0
ORB	0.03	261	298	145	51.9

Extracción de *features* en región común

La principal ventaja de las técnicas de mosaico de imágenes basadas en *features* es el hecho de que trabajan únicamente con los puntos comunes a ambas imágenes, lo que aumenta su eficiencia. Para conocer el área común a dos imágenes es necesario conocer la posición de la cámara en el momento de la captura (lo que permite conocer la distancia desplazada entre una adquisición y la siguiente). Cuanto más precisos sean los datos de posición, más preciso será el cálculo de las regiones comunes.

Si los datos de posición de la cámara presentan un error que provoque identificar como área común una región de la imagen que no lo es, puede extraerse un *feature* en una imagen que no pueda ser relacionado con ningún *feature* de la otra imagen. La solución a este problema será tratada en el Apartado 2.3.4.

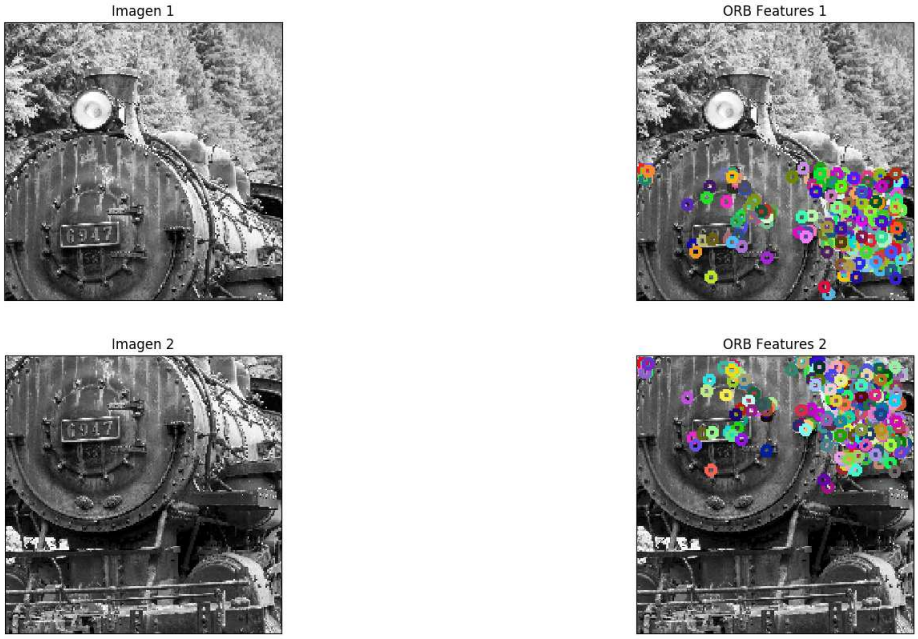

Figura 2.14 Resultado de implementar el algoritmo ORB.

2.3.2 *Feature matching*

Una vez obtenidos los *features* de las dos imágenes a superponer, el siguiente paso es relacionarlos mediante técnicas conocidas como *feature matching*. Esto permite emparejar aquellos píxeles que correspondan al mismo punto proyectado en distintas imágenes.

Brute force matching

Una de las técnicas más utilizadas a la hora de correlacionar *features* es la unión por fuerza bruta (*brute force matching*) [25] [30] [12]. Los algoritmos de fuerza bruta son aquellos que prueban todas las combinaciones posibles para obtener el resultado esperado. El *brute force matcher* une cada *feature* del primer conjunto con todos los *features* del segundo (Figura 2.15), comparando sus descriptores y devolviendo las correlaciones o *matches* que mejor se ajusten (Figura 2.16). Se obtiene así una relación biyectiva entre los dos conjuntos (cada *feature* del conjunto A está relacionado con un solo *feature* del conjunto B, y viceversa).

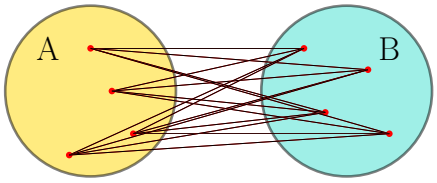

Figura 2.15 Algoritmo de fuerza bruta.

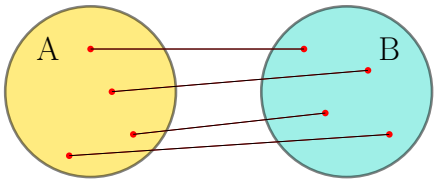

Figura 2.16 Resultado del algoritmo de fuerza bruta.

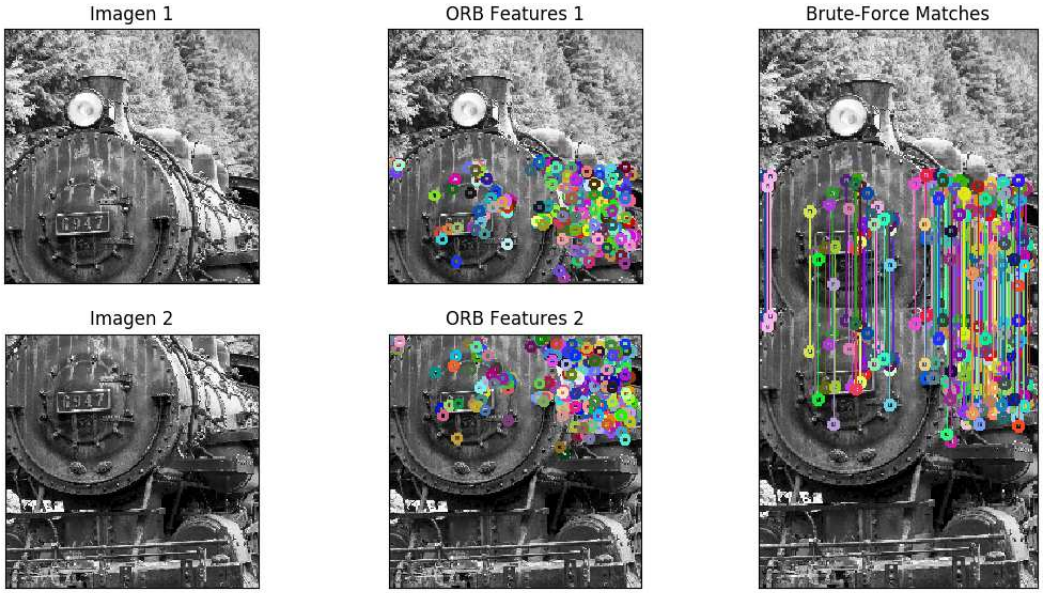

Figura 2.17 Resultado de implementar el algoritmo *Brute Force Matcher*.

Matching utilizando medidas de posición

Las técnicas de *matching* pueden ver elevada su eficiencia si se conoce el desplazamiento entre una imagen y otra, pues no es necesario efectuar la unión de todos los *features* de A con todos *features* de B, sino que puede trabajarse con ventanas de búsqueda en torno a cada punto de interés que contemplen los errores de posición.

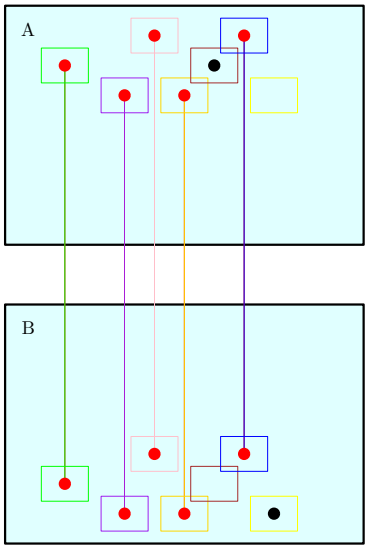

Figura 2.18 Matching con medidas de posición.

2.3.3 Cálculo de la homografía que relaciona las imágenes

A partir de los *matches* o correlaciones, puede calcularse la matriz de transformación homogénea que relaciona las dos imágenes. Para ello, se deben obtener los parámetros que proporcionen el mejor ajuste del modelo mediante la minimización del error cuadrático.

Coordenadas homogéneas

Las coordenadas homogéneas permiten describir un punto en el espacio proyectivo. La representación mediante coordenadas homogéneas en un espacio n-dimensional se realiza a través de coordenadas de un espacio (n+1)-dimensional.

Por ejemplo, sea un punto P representado en el espacio euclídeo:

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.6}$$

Puede expresarse en coordenadas homogéneas introduciendo una cuarta coordenada (que representa un factor de escala):

$$P = \begin{bmatrix} sx \\ sy \\ sz \\ s \end{bmatrix} \tag{2.7}$$

Matriz de transformación homogénea

La matriz de transformación homogénea permite representar la transformación de un vector expresado en coordenadas homogéneas de un sistema de referencia a otro. Para el caso tridimensional, la matriz de transformación T viene dada por:

$$T = \begin{bmatrix} R_{11} & R_{12} & R_{13} & p_1 \\ R_{21} & R_{22} & R_{23} & p_2 \\ R_{31} & R_{32} & R_{33} & p_3 \\ f_1 & f_2 & f_3 & w \end{bmatrix} = \begin{bmatrix} R_{(3x3)} & p_{(3x1)} \\ f_{(1x3)} & w_{(1x1)} \end{bmatrix} \tag{2.8}$$

Donde las matrices $R_{(3x3)}$, $p_{(3x1)}$, $f_{(1x3)}$, $w_{(1x1)}$ representan respectivamente la rotación, la traslación, la perspectiva y el escalado del punto transformado.

Por tanto, los puntos (x,y,z) y (u,v,w) estarán relacionados por una transformación homogénea T si se cumple que:

$$\begin{bmatrix} \alpha u \\ \alpha v \\ \alpha w \\ \alpha \end{bmatrix} = T \begin{bmatrix} \beta x \\ \beta y \\ \beta z \\ \beta \end{bmatrix} \tag{2.9}$$

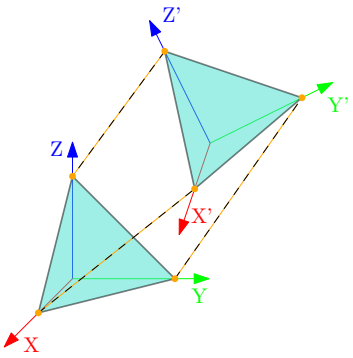

Figura 2.19 Transformación homogénea.

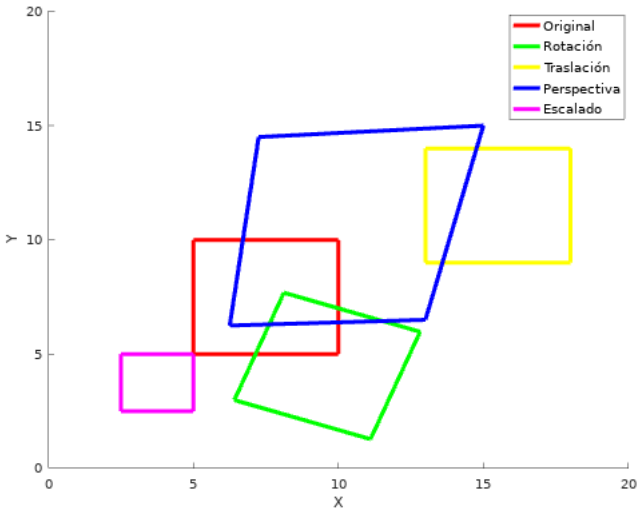

Figura 2.20 Ejemplos de transformaciones homogéneas.

Ajuste de los parámetros del modelo

Se conoce como modelo matemático al conjunto de formulas que expresan relaciones entre variables y/o parámetros que permiten analizar un sistema.

Los *matches* obtenidos en el apartado 2.3.2 dan como resultado dos conjuntos de puntos relacionados entre sí. Abordando una situación no ideal, no existe ninguna transformación homogénea que relacione los dos conjuntos a la perfección, por lo que será necesario encontrar aquella que mejor resultado provea. Entiéndase por situación no ideal todas aquellas en las que existan perturbaciones (objetos en movimiento, cambios de intensidad, errores en posicionamiento que supongan fallos en el cálculo de las áreas comunes a ambas imágenes, ...) que provoquen *features* que aparecen solo en una imagen o *feature matchings* erróneos.

El ajuste de los parámetros del modelo consiste, por tanto, en encontrar aquellos parámetros que minimicen el error de la transformación de un conjunto en otro. Sea (X_k,Y_k) con $k = 1,...,N$ un conjunto de N puntos en $\mathbb{R}^n$ se desea encontrar $F(X_k) \approx Y_k$ tal que minimice el error cuadrático medio, dado por la expresión:

$$E_{cm}(F) = \sqrt{\frac{\sum_{k=1}^{N} e_k^2}{N}} \tag{2.10}$$

$$e_k = Y_k - F(X_k) \tag{2.11}$$

La minimización del error cuadrático medio equivale a la minimización del error cuadrático.

$$\arg\min_F \{\sqrt{\frac{\sum_{k=1}^{N} e_k^2}{N}}\} = \arg\min_F \{\sum_{k=1}^{N} e_k^2\} \tag{2.12}$$

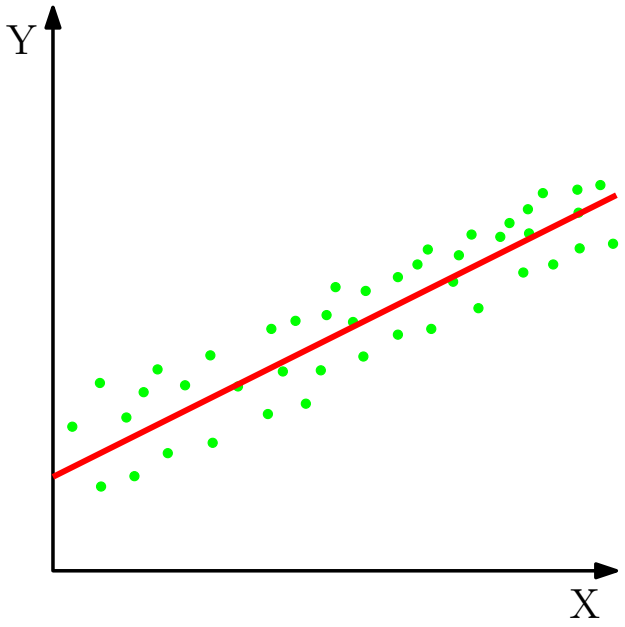

Figura 2.21 Recta de mejor ajuste a una nube de puntos en $\mathbb{R}^2$.

Particularizando el planteamiento para el caso de estudio y suponiendo variaciones unicamente en (x, y, θ), sean los conjuntos de puntos (x,y) y (u,v) relacionados en la fase de *matching* se debería cumplir que:

$$\begin{bmatrix} u_k \\ v_k \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & p_x \\ \sin\theta & \cos\theta & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ 1 \end{bmatrix} \tag{2.13}$$

Desarrollando:

$$\begin{aligned} u_k &= x_k\cos\theta - y_k\sin\theta + p_x \\ v_k &= x_k\sin\theta + y_k\cos\theta + p_y \end{aligned} \tag{2.14}$$

Despejando:

$$\begin{aligned} u_k - x_k\cos\theta + y_k\sin\theta - p_x &= 0 \\ v_k - x_k\sin\theta - y_k\cos\theta - p_y &= 0 \end{aligned} \tag{2.15}$$

Al no existir (x, y, θ) tales que la expresión 2.15 se cumpla $\forall k = 1,...,N$ se ajustarán los parámetros para minimizar el error cuadrático, dado por:

$$E_c = \sum_{k=1}^{N} (e_k)^2 = \sum_{k=1}^{N} ((u_k - x_k\cos\theta + y_k\sin\theta - p_x)^2 + (v_k - x_k\sin\theta - y_k\cos\theta - p_y)^2) \tag{2.16}$$

2.3.4 Problema asociado a los *outliers*

Como se indicó en el apartado 2.3.3 existen distintos elementos a tener en cuenta que afectan al desarrollo del mosaico:

- Objetos en movimiento.
- Cambios de intensidad de la luz.
- Inexactitud en el cálculo de las áreas comunes, derivada de errores en la medida de la posición.

Todas estas perturbaciones pueden suponer la existencia de correlaciones erróneas entre *features* por lo que, para evitar efectos indeseados, es necesario tratar estos puntos como *outliers*.

RANSAC

RANSAC (RANdom SAmple Consensus) es un algoritmo iterativo que permite calcular los parámetros de un modelo matemático para lograr el mejor ajuste de un conjunto de datos entre los que se incluyen valores atípicos (*outliers*).

Comparativa entre mínimos cuadrados y RANSAC

El método de los mínimos cuadrados es muy utilizado para el cálculo de los parámetros de un modelo matemático que produzcan el mejor ajuste de todos los datos, sin embargo, no discrimina entre *inliers* y *outliers*.

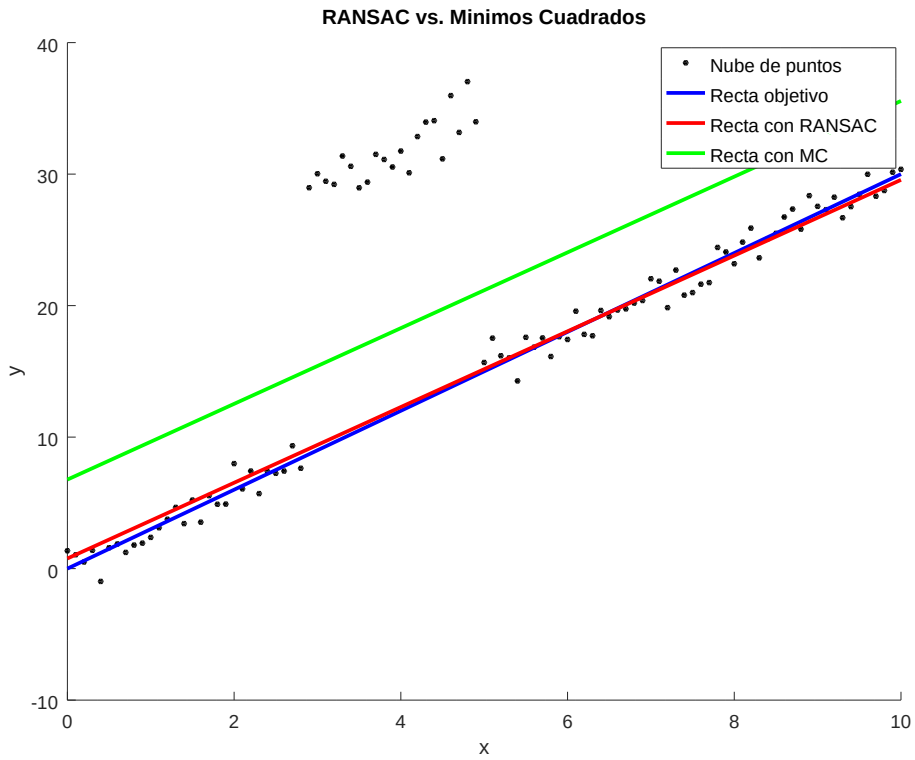

Figura 2.22 Comparativa entre mínimos cuadrados y RANSAC.

Algoritmo RANSAC

Sea k el número de iteraciones realizadas, N el número de iteraciones máximas a realizar, n el número de datos a escoger, D el conjunto de datos conocidos y U un valor umbral, el algoritmo RANSAC puede ser descrito como sigue:

Código 2.2 Algoritmo RANSAC.

```
MIENTRAS k < N
  Escoger n datos de forma aleatoria
  Calcular el modelo que se ajuste a esos datos
    PARA CADA dato d en D (sin contar los n escogidos)
      SI la distancia de d al modelo es menor que U
        Incluir d en el conjunto de inliers
      FIN SI
    FIN PARA CADA
    SI el numero de inliers es el mayor hasta el momento
      Guardar numero de inliers
      Guardar los n datos escogidos
    FIN SI
FIN MIENTRAS
Calcular el modelo que se ajuste a los datos, excluyendo outliers
```

Número de iteraciones

El número de iteraciones necesarias puede ser calculado a partir del siguiente razonamiento: sean p la probabilidad de éxito del algoritmo, t el porcentaje de *inliers* en los datos, n el número de datos escogidos y N el número de iteraciones necesarias, la probabilidad de que al menos uno de los n puntos sea un *outlier* vendrá dada por $1-t^n$, por lo que la probabilidad de que el algoritmo nunca seleccione n puntos sin *outliers* será $(1-t^n)^N$, por tanto:

$$1-p=(1-t^n)^N \tag{2.17}$$

Aplicando logaritmos y despejando N:

$$N=\frac{\log(1-p)}{\log(1-t^n)} \tag{2.18}$$

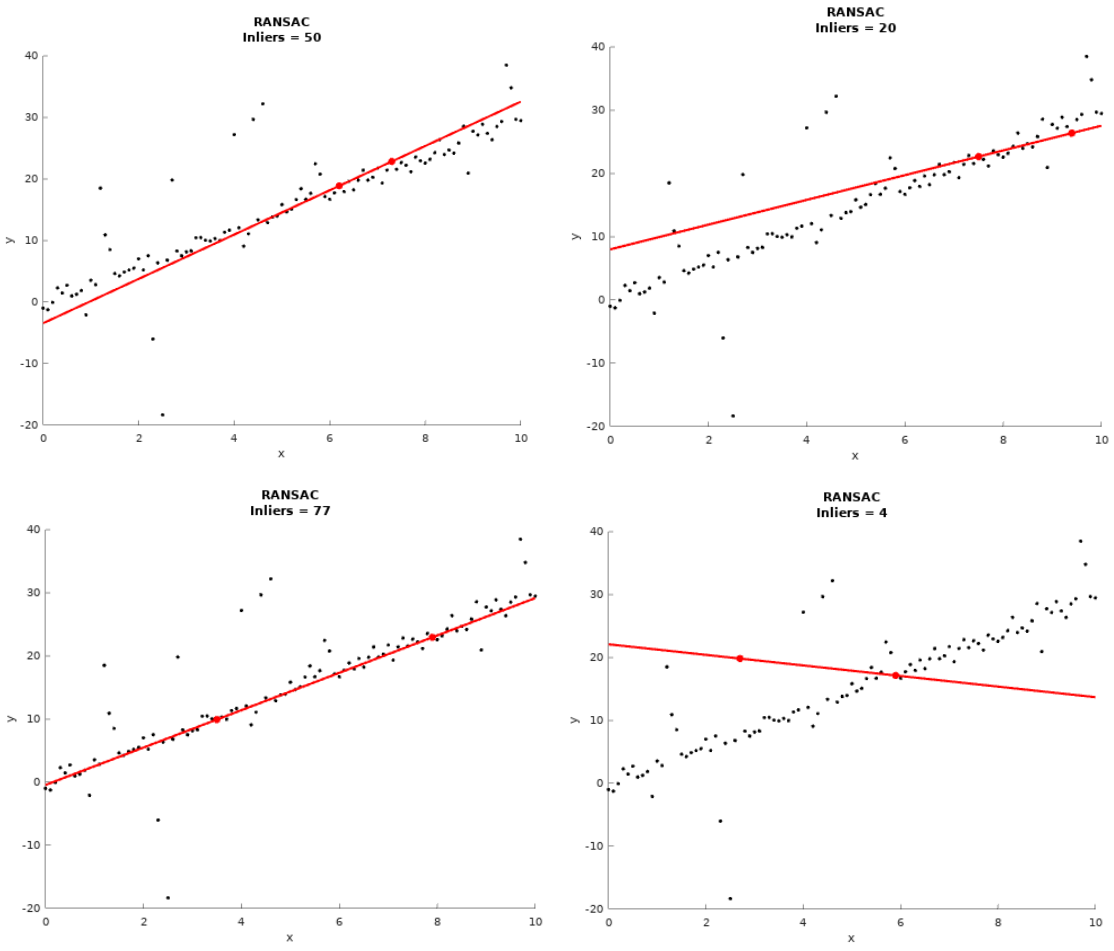

Figura 2.23 Algunos resultados intermedios obtenidos al implementar el algoritmo RANSAC.

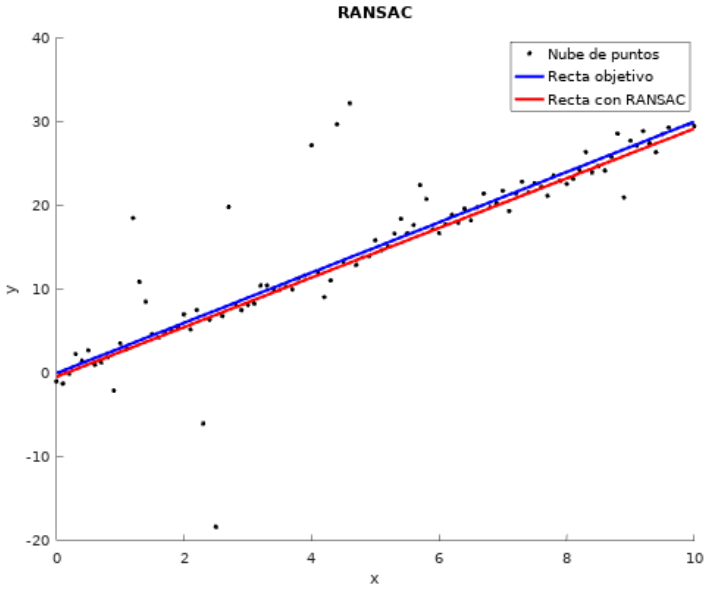

Figura 2.24 Resultado final obtenido al implementar el algoritmo RANSAC.

2.4 Conclusiones

Atendiendo a las especificaciones descritas en el Capítulo 1, se ha decidido escoger un método basado en *features* como solución al problema. La elección de este tipo de técnica se basa, principalmente, en el hecho de trabajar con conjuntos de imágenes con una tasa de adquisición no muy elevada y una zona de superposición de entre el 60 % y el 70 % del total de puntos. Además, puesto que se intentará garantizar un vuelo a altura constante evitando variaciones en *pitch* y *roll*, se ha optado por implementar un algoritmo que solo contemple variaciones en el plano paralelo al suelo, es decir, desplazamiento según eje X, desplazamiento según eje Y y giro en *yaw*.

Por otra parte, en lo referido a algoritmos de extracción, se ha decidido utilizar el algoritmo ORB para lograr el objetivo descrito en el Apartado 1.1 por las siguientes razones:

- Es el algoritmo que presenta resultados en el menor tiempo, por lo que es el más apropiado para aplicaciones de tiempo real.
- Pese a ser el que menos tiempo de ejecución necesita, no por ello deja de otorgar buenos resultados.
- Es el que mayor porcentaje de *matches* presenta al variar la escala, útil si aparecen ligeras variaciones en la altura de vuelo del UAV (*unmanned aerial vehicle* o vehículo aéreo no tripulado) durante su trayectoria mientras adquiere las imágenes.

Otro detalle a tener en cuenta es que SIFT y SURF son algoritmos patentados por lo que debe adquirirse una licencia para utilizarlos con propósitos comerciales, sin embargo, en lo relativo a propósitos académicos o de investigación, su uso es gratuito.

3 Esquema General del Método Propuesto

3.1 Introducción

En este capítulo se presenta un esquema general del método diseñado para la generación de mosaicos y se describen los módulos que lo componen.

Como se detalló en el capítulo anterior, se ha decidido implementar un método indirecto puesto que, por lo general, estos presentan mayor eficiencia y robustez que los directos. La estructura del algoritmo implementado coincide con las tres etapas características de este tipo de técnica: extracción de *features* en la zona común a dos imágenes consecutivas, correlación o asociación de los *features* obtenidos y cálculo de la transformación homogénea que provoca el mejor ajuste de las dos imágenes a partir de los puntos asociados.

3.2 Extracción de *features*

El primer paso del método diseñado es extraer *features* en las dos imágenes a unir. Un *feature* o característica es una parte de la imagen que proporciona información útil para realizar una cierta tarea. En concreto, el método propuesto utiliza puntos de interés como características y realiza la extracción mediante el algoritmo ORB (*Oriented FAST and Rotated BRIEF*).

Para aumentar la eficiencia de la extracción, y reducir su coste computacional, se realizará únicamente en las zonas comunes a las dos imágenes a unir. Para ello, se utilizará una estimación del desplazamiento obtenida a partir de las medidas del GPS.

Además, el algoritmo no extraerá puntos en regiones cercanas a los bordes, evitando así posibles fallos provocados por la distorsión de la lente de la cámara e intentará homogeneizar la distribución de las detecciones para mejorar la calidad de los resultados ofrecidos.

3.3 Correlación de *features*

Una vez localizados y descritos los puntos de interés, el siguiente paso es emparejar aquellos que correspondan al mismo punto del mundo real. Gracias a utilizar vectores de descripción binarios, el cálculo de la distancia de Hamming entre dos descriptores es una operación cuya ejecución requiere un coste computacional muy bajo.

Además de emplear la distancia de Hamming como medida de la similitud entre dos puntos, se utilizarán ventanas de búsqueda (*windowing*) cuya posición será estimada a partir de las medidas del GPS, para reducir aun más el tiempo requerido.

3.4 Cálculo de la transformación homogénea

El último paso es obtener la transformación homogénea que provoque el mejor ajuste de los puntos asociados anteriormente. El cálculo de los parámetros de la transformación que conlleve el menor error posible se corresponde con la resolución de un problema de mínimos cuadrados.

Se particularizará la transformación para reducir el número de incógnitas, disminuyendo el tiempo necesario para resolver la minimización. Además, el algoritmo propuesto hace uso de la descomposición en valores simples (SVD) para obtener los parámetros mencionados, por ser este un método eficiente y rápido.

Una vez calculada la transformación, será aplicada a todos los puntos de una imagen para que quede superpuesta sobre la que le precede en la sucesión de imágenes capturadas por el robot.

3.5 Conclusiones

Sea I_k la imagen correspondiente a la adquisición k-ésima y N el número de adquisiciones realizadas en la inspección, el algoritmo diseñado puede ser descrito mediante el diagrama de flujo de la Figura 3.1 o mediante el Pseudocódigo 3.1.

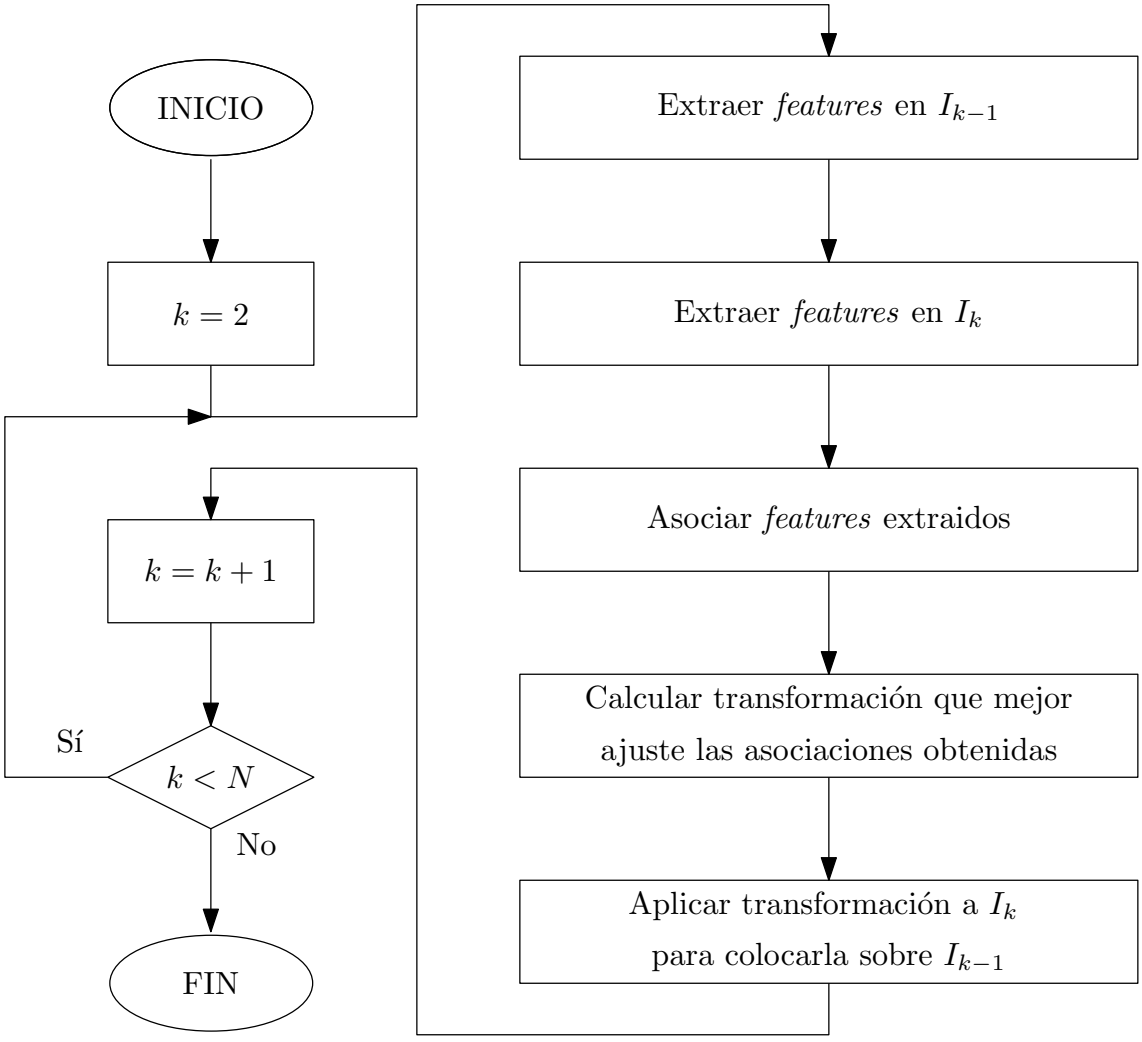

Figura 3.1 Algoritmo implementado para la generación de mosaicos.

Código 3.1 Algoritmo implementado para la generación de mosaicos.

```
i_anterior = Primera imagen del conjunto de imágenes capturadas
PARA CADA i EN el conjunto de imágenes capturadas (sin contar la primera)
  Extraer features en i_anterior
  Extraer features en i
  Asociar features extraídos
  Calcular transformación que mejor ajuste las asociaciones obtenidas
  Aplicar transformación a i para colocarla sobre i_anterior
  Almacenar en i_anterior la imagen i
FIN PARA CADA
```

4 Aplicación del Algoritmo ORB para la Extracción de *Features*

4.1 Introducción

En este capítulo se describirá el comportamiento del algoritmo ORB como extractor de puntos de interés y la relación entre los resultados obtenidos y los parámetros característicos de esta técnica de extracción.

ORB (*Oriented FAST and Rotated BRIEF*) es un detector y descriptor de puntos de interés (en ocasiones también llamados *keypoints* o puntos singulares) que combina el detector FAST (*Features from Accelerated Segment Test*) y el descriptor BRIEF (*Binary Robust Independent Elementary Features*).

El algoritmo ORB es utilizado en muchas ocasiones como una alternativa a los algoritmos SIFT y SURF (dos de los métodos más utilizados para la extracción de *features*) que, por lo general, presenta resultados en menos tiempo y sin una perdida de eficiencia considerable. Una comparativa entre los tres métodos mencionados ha sido ya presentada en el Apartado 2.3.1.

4.2 Algoritmo ORB como detector de *features*

Se conoce como detección al proceso mediante el cual un extractor obtiene la localización de los puntos de interés en una imagen. Los puntos detectados son aquellos con menor autosimilutud, o lo que es lo mismo, con mayor singularidad, es decir, puntos cuyas características difieran de las de sus vecindades.

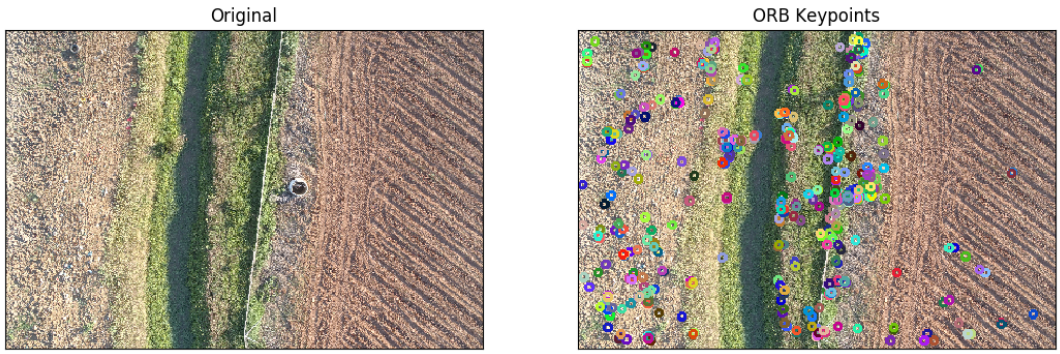

Figura 4.1 Extracción de *features* en una de las imágenes del set.

El detector utilizado por ORB es una modificación del algoritmo de detección FAST , conocido como oFAST (*Oriented FAST*). La ventaja de oFAST sobre FAST es que incorpora un mecanismo para compensar la orientación, convirtiéndolo en un método invariante frente a la rotación.

4.2.1 Método para la compensación de la orientación

Se define el momento de orden p,q de una sección $(i,j) \in S$ de la imagen I como:

$$m_{pq} = \sum_{i,j} i^p j^q I_{i,j}, (i,j) \in S \tag{4.1}$$

A partir de los momentos m_{00}, m_{01} y m_{10} se puede calcular el centroide de la sección S como:

$$C = \left[\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right]^\top \tag{4.2}$$

Identificado el centro de la sección, O, puede construirse el vector $\overrightarrow{OC}$, cuya orientación vendrá dada por la expresión:

$$\theta = \arctan \frac{m_{01}}{m_{10}} \tag{4.3}$$

Conocida la orientación θ, puede rotarse la sección S para colocarla según una orientación canónica, θ_0, por lo que el algoritmo ofrecerá los mismos resultados $\forall \theta$.

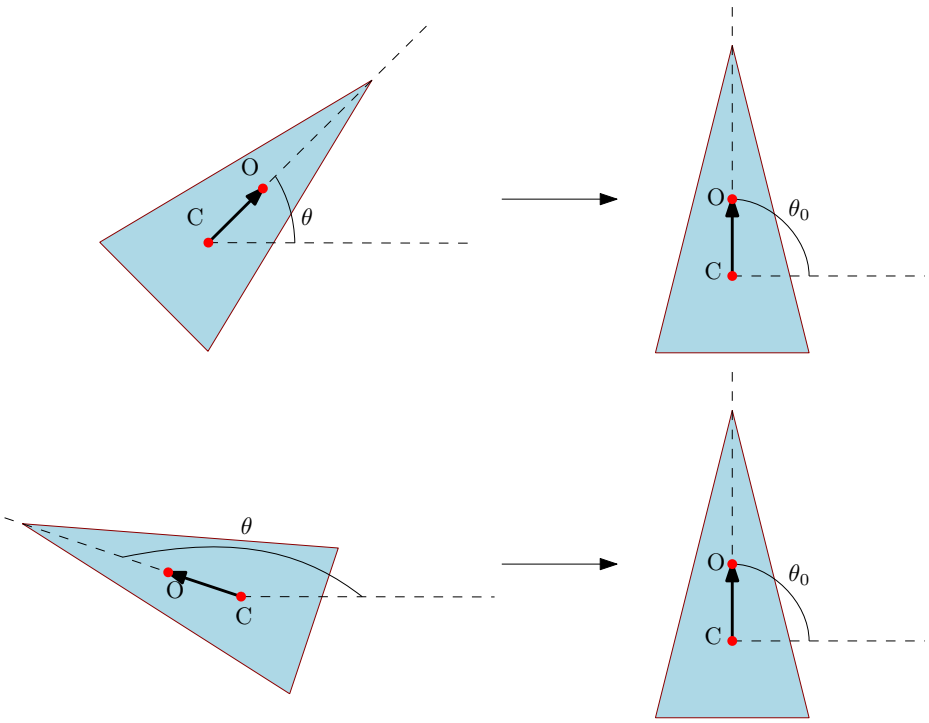

Figura 4.2 Compensación de la rotación.

4.2.2 Número de *features*

Uno de los parámetros más importante del algoritmo ORB, así como de cualquier algoritmo de detección de *features*, es el número máximo de puntos a encontrar. Un número de puntos muy elevado supondrá un coste computacional mayor durante la fase de correlación, mientras que un número muy bajo podría ofrecer como resultado un conjunto de puntos insuficiente para obtener las correlaciones necesarias que permitan realizar con éxito el *matching* de las imágenes.

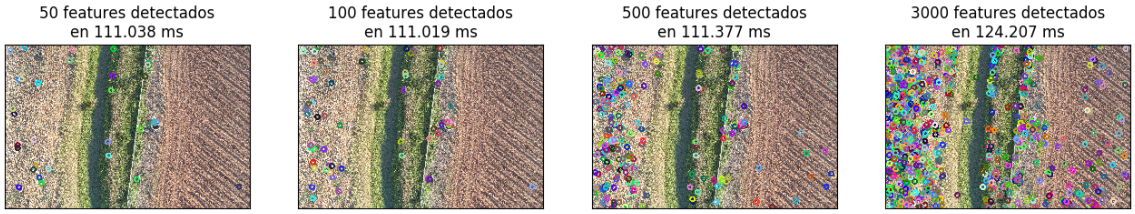

Figura 4.3 Comportamiento de ORB al variar el número de *features* a extraer.

Como se puede comprobar en la Figura 4.3, aumentar el número de extracciones no aumenta en exceso el tiempo de ejecución, pero sí afectará de forma significativa al coste temporal de la correlación de los puntos detectados.

4.2.3 *Edge threshold*

El *edge threshold* o umbral de borde es otro de los parámetros a tener en cuenta a la hora de configurar ORB. Se trata del tamaño del marco de la imagen donde no habrá extracción de *features*. La existencia de este umbral permite evitar posibles problemas asociados a la extracción de puntos en zonas cercanas a los límites de la imagen.

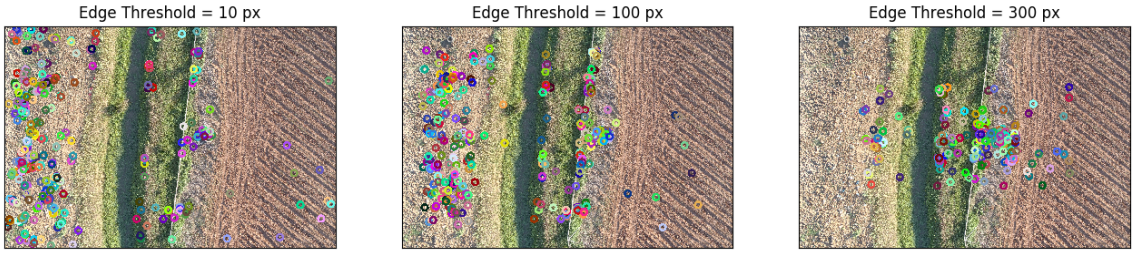

Figura 4.4 Comportamiento de ORB al variar el *edge threshold*.

Otro motivo por el que podría preferirse no buscar puntos de interés en entornos próximos a los límites de la imagen es evitar realizar una calibración interna del sistema de adquisición (Figura 4.5). La distorsión provocada por la lente de la cámara suele ser mayor en los bordes de la imagen, como ilustra la Figura 4.6, por lo que ignorar esta zona puede ser en muchas ocasiones una solución rápida y efectiva a este problema.

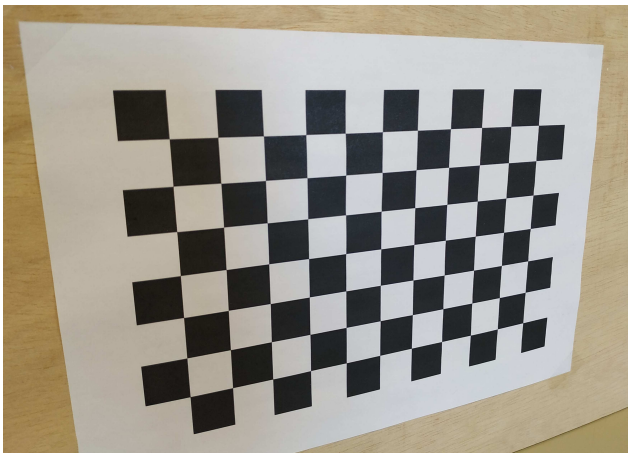

Figura 4.5 Patrón de calibración utilizado para obtener los parámetros intrínsecos de una cámara..

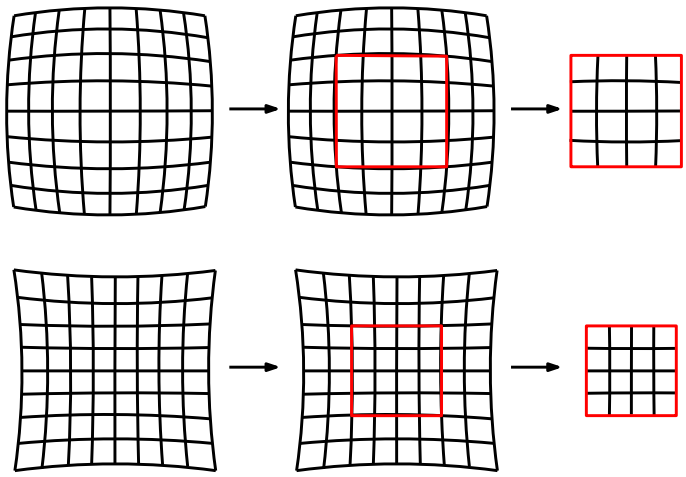

Figura 4.6 Efectos de la distorsión de una lente y su solución con *edge threshold*.

4.2.4 Máscara

En ocasiones es necesario ignorar parte de una imagen tal y como se ha hecho con los bordes. La solución a este problema supone la creación de una máscara, es decir, una matriz (cuyas dimensiones coincidan con las de la imagen a analizar) formada por un único canal, cuyos elementos solo comprendan los valores lógicos 0 o 1 (255 si se representa con 8 bits). Sea M la máscara de la imagen I, podrán ser puntos de interés únicamente aquellos puntos $I_{i,j}$ tal que $M_{i,j} = 1$.

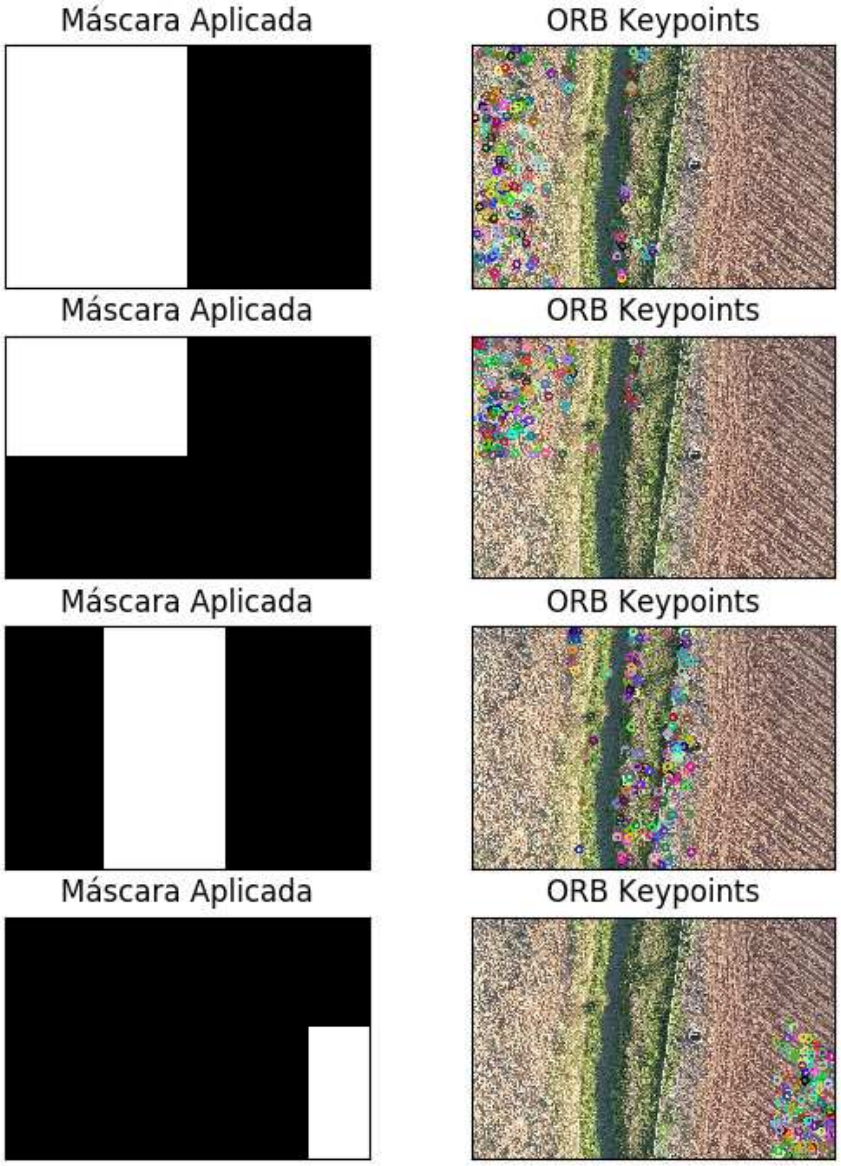

Figura 4.7 Extracción de *features* con distintas máscaras aplicadas.

Evitar el estudio de una región de la imagen es un hecho a tener en cuenta si se conoce de antemano que esta introducirá valores atípicos (*outliers*). Por ejemplo, si en las imágenes capturadas para inspeccionar una superficie se muestra una carretera por la que circulan coches, aplicar una máscara adecuada restringirá la extracción a aquellas zonas en las que no aparezcan objetos en movimiento.

El algoritmo de *mosaicking* que ha sido implementado hace uso de máscaras para evitar la extracción de *features* en puntos no comunes a dos imágenes consecutivas. Los puntos no comunes han sido calculados a partir de una estimación del desplazamiento, obtenida con las coordenadas GPS del UAV asociadas a cada captura. Sean $\Delta x = x_k - x_{k-1}$ e $\Delta y = y_k - y_{k-1}$ obtenidas a partir de las posiciones (x_k, y_k) y (x_{k-1}, y_{k-1}) calculadas con las medidas del GPS (véase Anexo A), el desplazamiento en línea recta efectuado entre el instante $k-1$ y k puede ser calculado como:

$$d = \sqrt{\Delta x^2 + \Delta y^2} \tag{4.4}$$

Una alternativa al uso de máscaras binarias es trabajar con regiones de interés o ROIs (*Regions Of Interest*), es decir, establecer como entrada al algoritmo ORB una *subimagen* contenida en la imagen original. El inconveniente es que las coordenadas de los *keypoints* calculados estarán referidas a la *subimagen*, por lo que será preciso sumar a cada localización la posición relativa de la ROI respecto a la imagen original. Por otro lado, la ventaja de este método frente al uso de máscaras es que no es necesario recorrer todos los puntos de la imagen, reduciendo así el coste computacional.

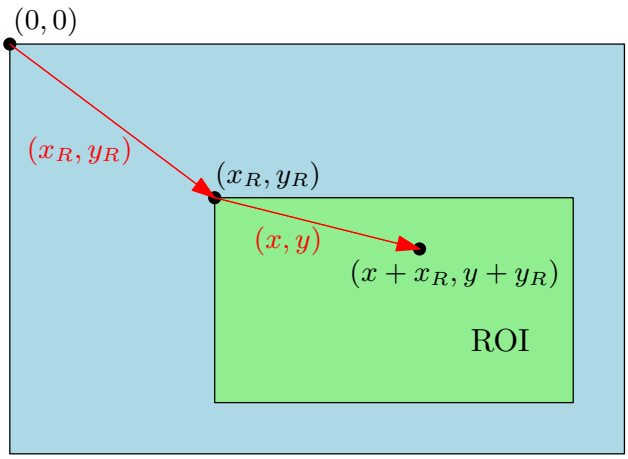

Figura 4.8 Coordenadas relativas de un *feature* en una ROI.

4.2.5 Distribución de los *features*

Ciertas imágenes pueden contener regiones con características claramente diferenciadas (por ejemplo, a distintas alturas), lo que, en ocasiones, puede significar que en una o varias de ellas no aparezca ningún *feature* tras la extracción. Como consecuencia, se realizará una buena unión únicamente en aquellas zonas con mayor número de puntos singulares detectados.

La solución más directa a este problema es aumentar el número de *features* de ORB hasta conseguir extracciones en todas las regiones de la imagen. Sin embargo, se trata de una solución ineficiente que aumenta el coste computacional del algoritmo. Una solución mucho más eficiente se basa en la división de estas zonas mediante máscaras o ROIs, utilizando ORB de manera independiente en cada una.

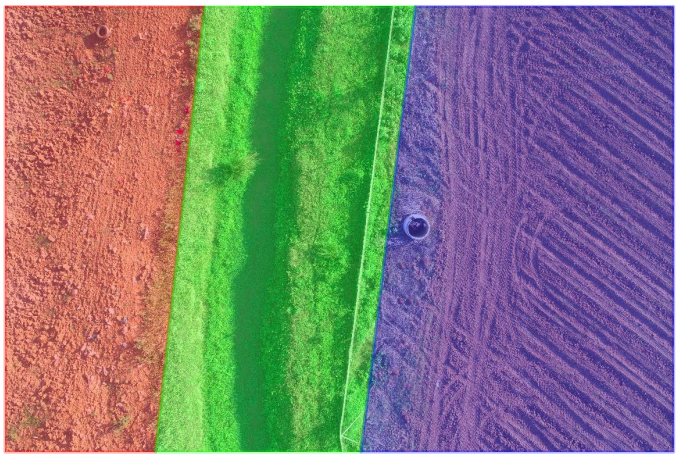

Figura 4.9 Descomposición de una imagen en tres regiones de interés.

Puesto que las imágenes tratadas por el algoritmo presentan un elemento central que divide el terreno en dos, se ha decidido extraer *features* en dos ROIs (una para la parte izquierda y otra para la derecha).

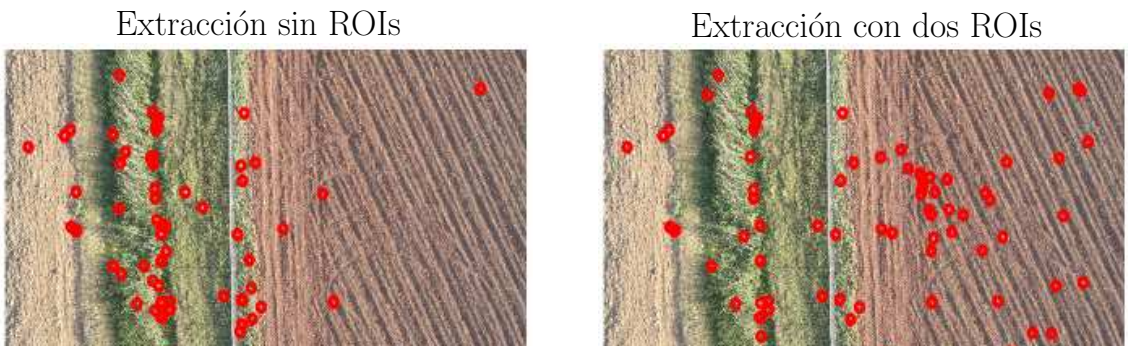

Figura 4.10 Efecto de la distribución homogénea de *features*.

4.2.6 Invarianzas del detector

Una de las ventajas del algoritmo ORB frente a otros extractores es su invarianza respecto a la rotación, la perspectiva y el escalado, es decir, que presenta resultados aproximadamente iguales ante tales transformaciones. La Figura 4.11 muestra el resultado del algoritmo frente a distintas transformaciones. Las invarianzas serán útiles para solucionar los siguientes problemas:

- La invarianza ante rotación permitirá relacionar dos imágenes consecutivas cuando exista una variación del *yaw*, es decir, cuando el UAV efectúe un giro en torno al eje z.
- La invarianza ante escalado hará al algoritmo estable frente a pequeñas variaciones de altura.
- La invarianza ante perspectiva permitirá al algoritmo ser estable frente a pequeñas variaciones de *pitch* y de *roll*.

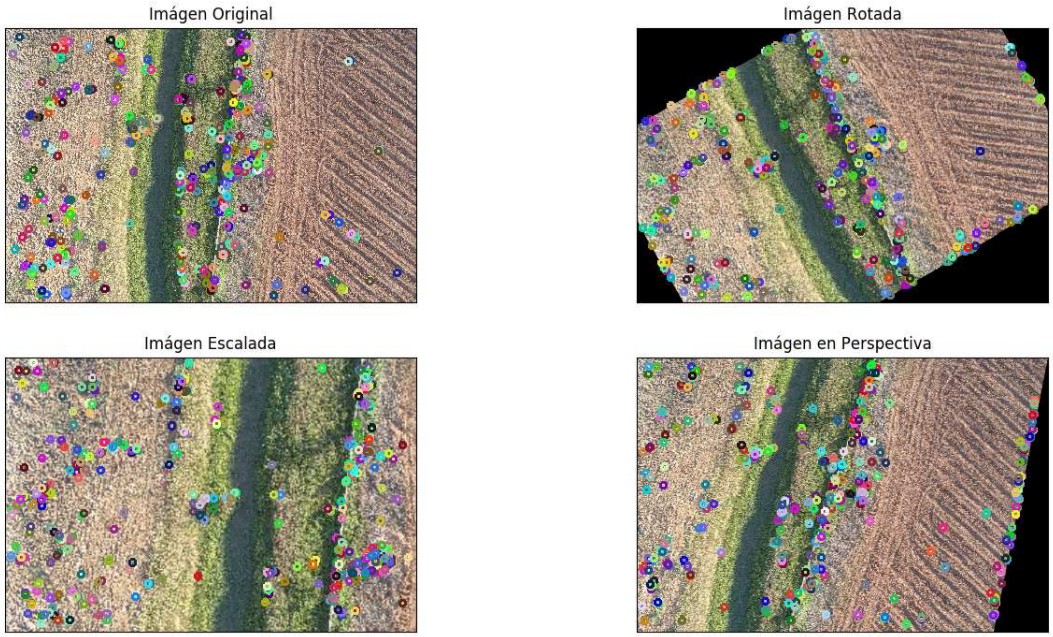

Figura 4.11 Comportamiento de ORB ante distintas transformaciones.

Nótese que, para comparar los resultados obtenidos de la Figura 4.11, no se deben tener en cuenta los *features* extraídos en los nuevos bordes diagonales que aparecen en los casos de rotación y perspectiva, pues en una imagen capturada por una cámara no existirían tales límites.

4.2.7 Comportamiento frente a cambios de intensidad y ruido en las imágenes

En la Figura 4.12 se presenta el comportamiento del extractor frente a variaciones en la intensidad de la luz. Se trata de un análisis importante a tener en cuenta si en la inspección a realizar se prevén cambios de luz (por ejemplo, si existe algún elemento que arroje sombra en una parte del recorrido o si se vuela el suficiente tiempo como para que el cambio del ángulo de incidencia de la luz del Sol deba tenerse en cuenta). Puede comprobarse como el algoritmo obtiene aproximadamente el mismo resultado en las tres imágenes.

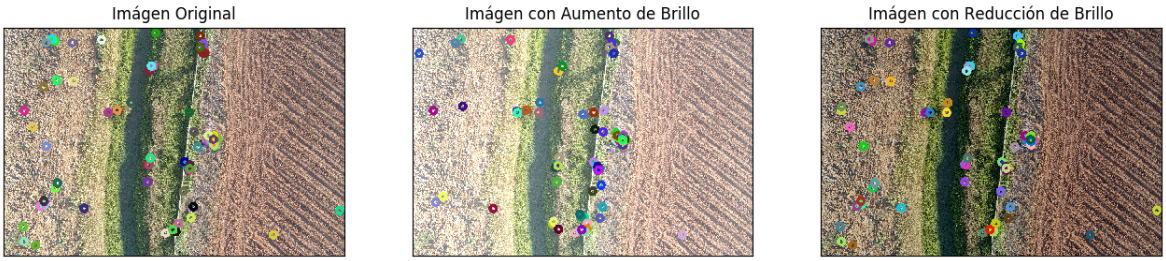

Figura 4.12 Comportamiento de ORB ante variaciones en la intensidad.

En la Figura 4.13 se presenta el comportamiento del extractor frente a imágenes sometidas a ruido gaussiano y a ruido *sal y pimienta*. El resultado ofrecido por el algoritmo es diferente en cada caso, pero con un porcentaje significante de puntos extraídos en la misma posición en las tres imágenes, por lo que se podría afirmar que ORB es un método robusto ante ruido digital.

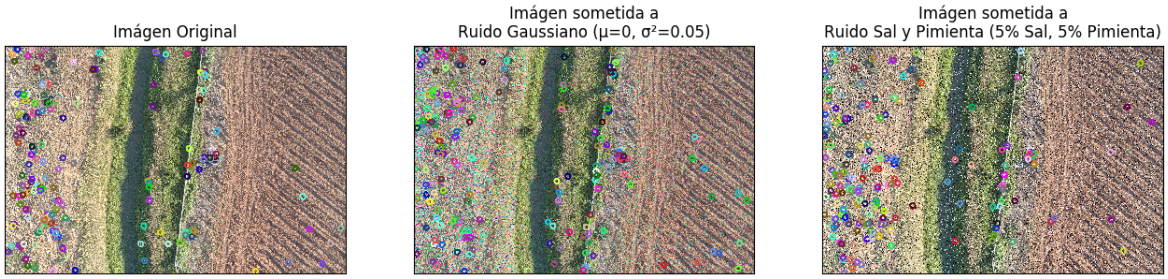

Figura 4.13 Comportamiento de ORB ante imágenes sometidas a ruido.

El ruido gaussiano es un tipo de ruido según el cual todos los píxeles que componen la imagen cambian su valor teórico de acuerdo a una distribución normal. Una distribución normal, también conocida como gaussiana, $\mathcal{N}(\mu, \sigma)$ es una distribución de probabilidad con función de densidad definida según la Ecuación 4.5, con media μ, desviación estándar σ y varianza σ^2. En este caso, además, se trata de un proceso estocástico de ruido blanco, pues las variables aleatorias que lo conforman (los píxeles de la imagen) no están correlacionadas entre sí, por lo que la media del proceso debe ser igual a cero.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4.5}$$

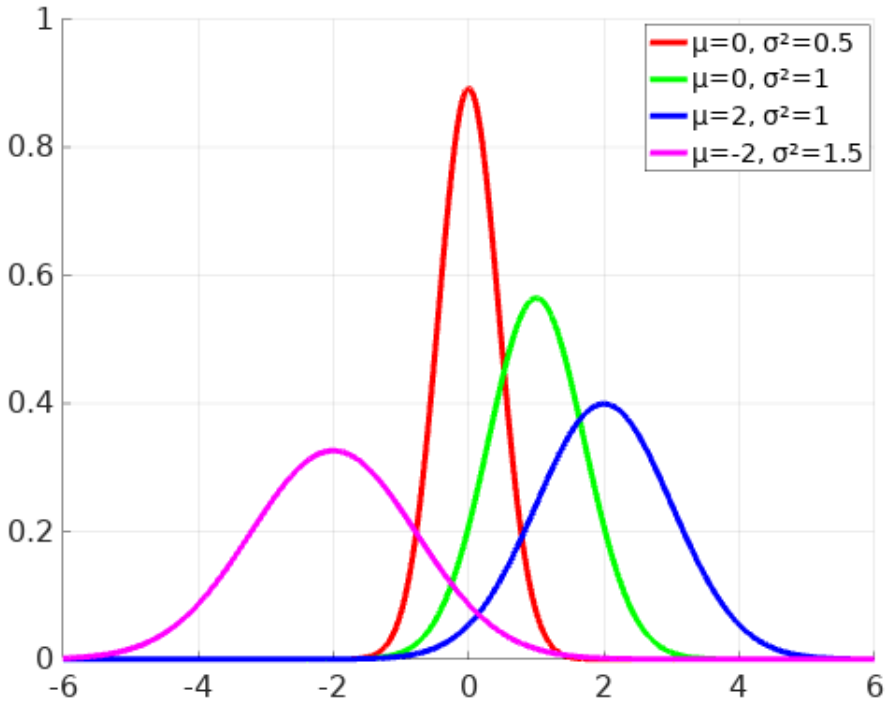

Figura 4.14 Distribuciones normales.

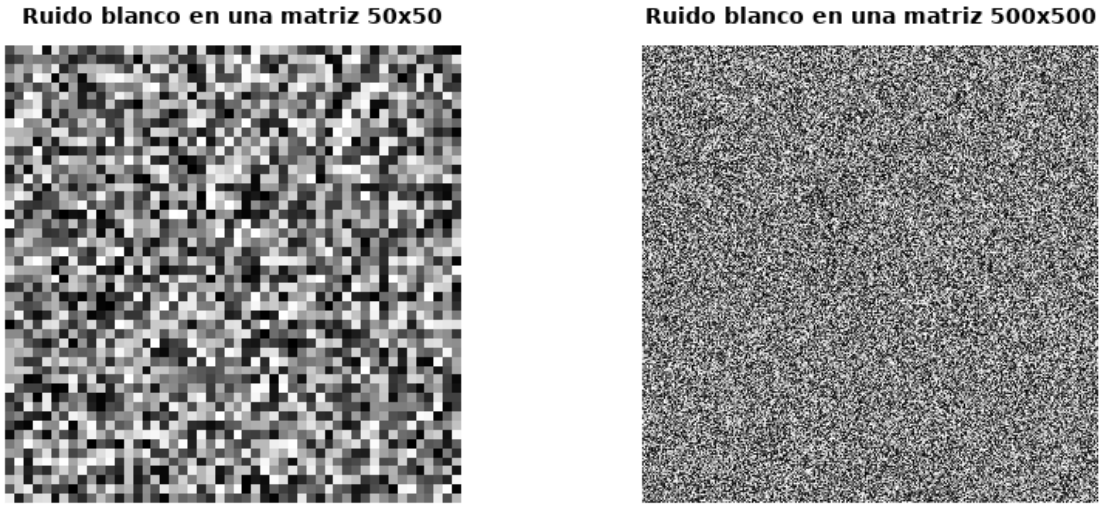

Figura 4.15 Imágenes B/N de ruido blanco.

El ruido *sal y pimienta* es un tipo de ruido digital según el cual algunos pixeles al azar de la imagen cambian su valor real por el valor más bajo posible y otros (también al azar) por el valor más alto posible. El número de píxeles que cambian su valor a 0 frente al total de píxeles que componen la imagen se conoce como densidad de *pimienta*, mientras que el número de píxeles que cambian a un valor alto (255 si se utilizan 8 bits para representar la intensidad) frente al total se conoce como densidad de *sal*. Este ruido pretende simular manchas de polvo en la óptica de la cámara o componentes defectuosos.

Figura 4.16 Imagen sometida a ruido gaussiano y a ruido *sal y pimienta*.

4.3 Algoritmo ORB como descriptor de *features*

Se conoce como descripción al proceso mediante el cual un extractor genera un vector de datos que describe la vecindad en torno a cada uno de los *features* detectados. De este modo, cada punto de interés tendrá asociado un *identificador* que lo diferencia de los demás. Idealmente, los descriptores son invariantes ante transformaciones, lo que permite identificar al mismo *feature* cuando la imagen haya sido transformada.

El vector generado por el descriptor suele ser un vector de números reales, sin embargo, ORB utiliza un vector de valores binarios con un tamaño de 256 bits. Sea $d(p_1,p_2)$ una componente del vector de descripción del *feature* f, S una sección de la imagen y $S(p)$ la intensidad de S en el punto $p = (x,y)^\top$:

$$d(p_1,p_2) := \begin{cases} 1 & si \quad S(p_1) < S(p_2) \\ 0 & si \quad S(p_1) \geq S(p_2) \end{cases} \tag{4.6}$$

El vector de descripción de f vendrá dado por:

$$D_f = \sum_{1 \leq k \leq 256} 2^{k-1} d(a_k,b_k) \tag{4.7}$$

Donde los valores a_k y b_k se escogen según una distribución gaussiana en torno al centro de S.

El descriptor utilizado por ORB es una modificación del algoritmo de descripción BRIEF , conocido como rBRIEF (*Rotated BRIEF*). La ventaja de rBRIEF sobre BRIEF es que incorpora un método para trabajar con la orientación de los *keypoints* a describir.

4.3.1 Método para dotar de dirección a BRIEF

Para cada uno de los puntos $p \in \{a_k \cup b_k\}$ se define la matriz M como:

$$M = \begin{bmatrix} x_1, x_2, ..., x_n \\ y_1, y_2, ..., y_n \end{bmatrix} \tag{4.8}$$

Conocida la orientación de S, θ, puede construirse la matriz de rotación R_θ:

$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{4.9}$$

Aplicando esta rotación a se consigue una versión *dirigida* de M:

$$M_\theta = R_\theta M \tag{4.10}$$

Por tanto, para describir a f, se debe aplicar el operador definido en la Ecuación 4.7 $\forall p \mid (x,y) \in M_\theta$.

4.4 Validación de los resultados

Sea I_k la imagen obtenida en adquisición k-ésima, I'_k la imagen obtenida en adquisición k-ésima transformada y situada en el mosaico y $ROI_{(k-1)\leftrightarrow k}$ la región común de I_k e I'_{k-1} (calculada con una estimación del desplazamiento a partir de las coordenadas GPS del UAV y del *yaw* de la plataforma referido a una referencia), el algoritmo que ha sido implementado puede ser descrito mediante el diagrama de flujo de la Figura 4.17 o mediante el Pseudocódigo 4.1.

Para la implementación, se ha decidido hacer uso de la clase *cv::ORB* incluida en la librería *OpenCV 4.1.0* y de sus métodos *detect()* y *compute()* para detectar y describir, respectivamente, los *features* de cada una de las imágenes que compondrán el mosaico.

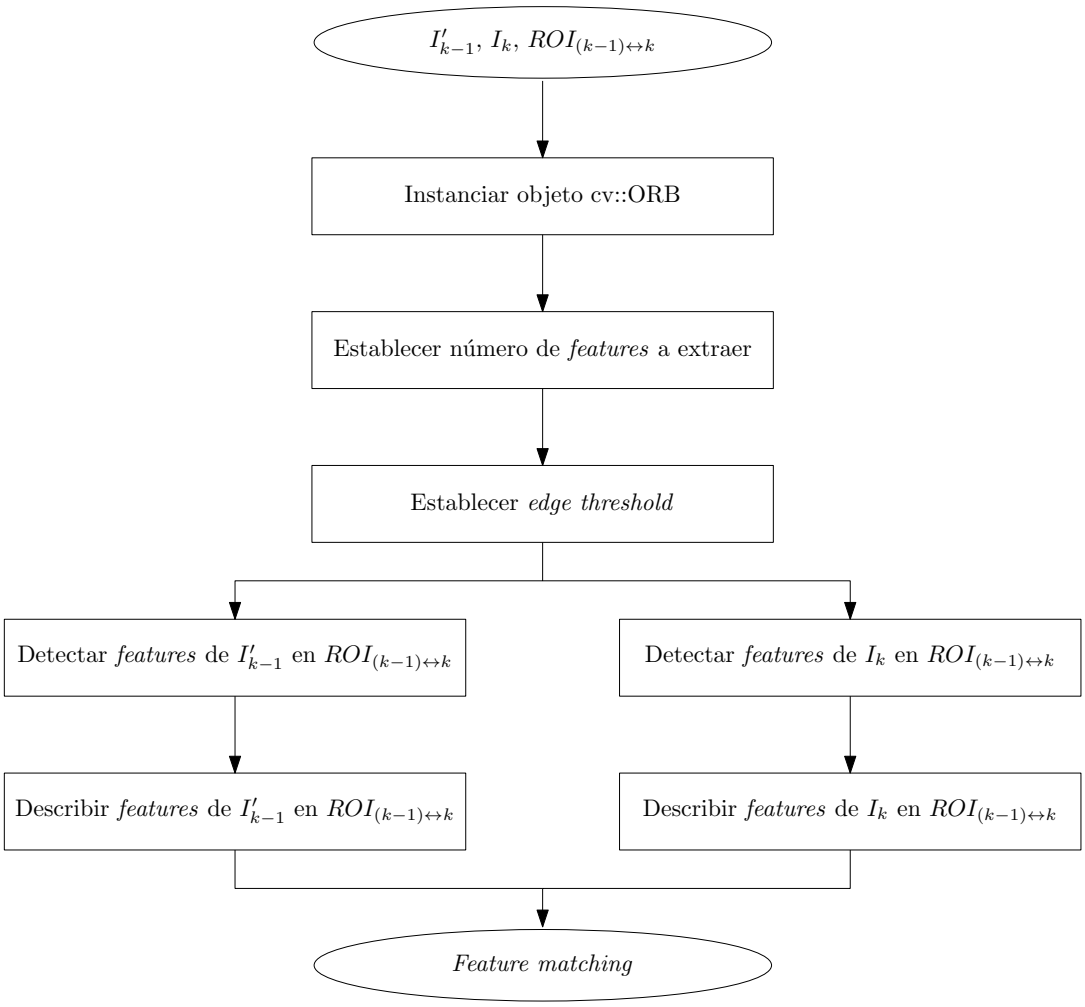

Figura 4.17 Algoritmo implementado para la detección y descripción de *features* con ORB.

Código 4.1 Algoritmo de extracción de *features*.

```
Instanciar objeto cv::ORB
Establecer número de features a extraer
Establecer edge threshold
Detectar features de I'(k-1) y features de I(k) en ROI(k-1,k)
Describir features de I'(k-1) y features de I(k) en ROI(k-1,k)
```

Para poner a prueba todo lo implementado hasta el momento, se utilizará un conjunto de siete imágenes. Los resultados presentados a continuación avalan el correcto funcionamiento del algoritmo.

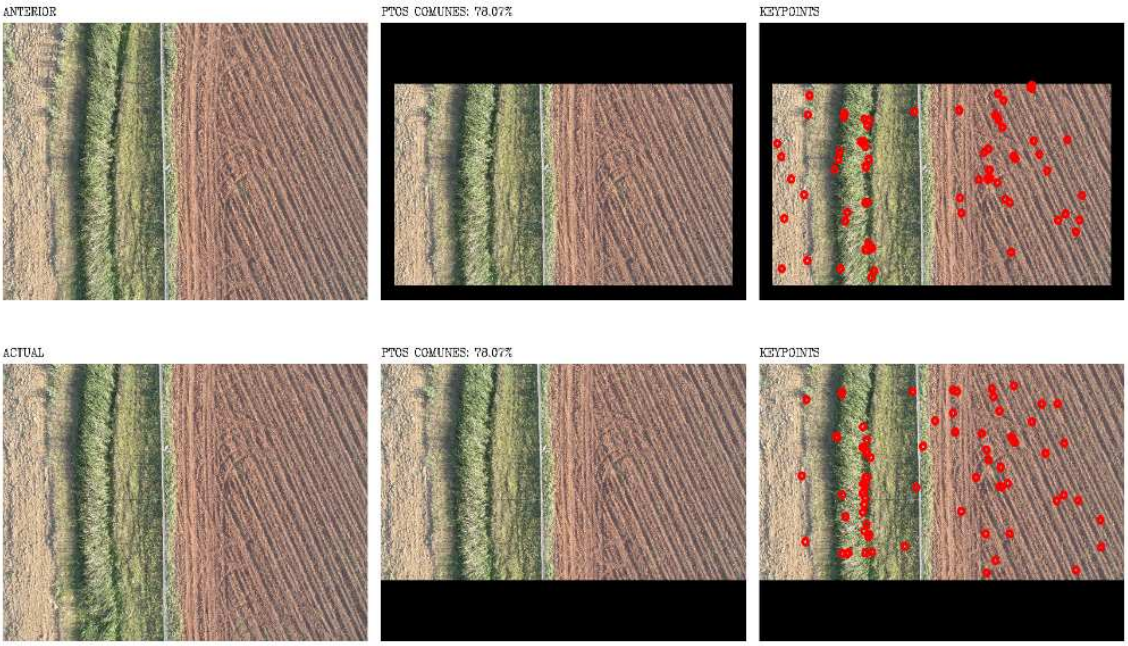

Figura 4.18 Resultado del algoritmo para las imágenes I_1 e I_2.

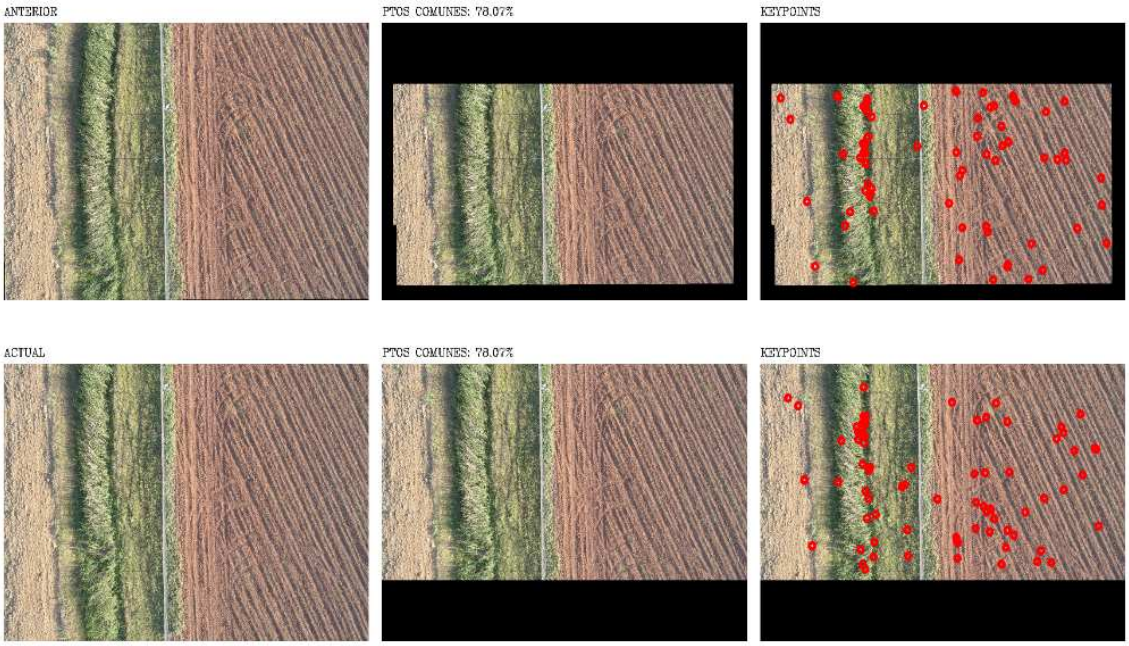

Figura 4.19 Resultado del algoritmo para las imágenes I_2 e I_3.

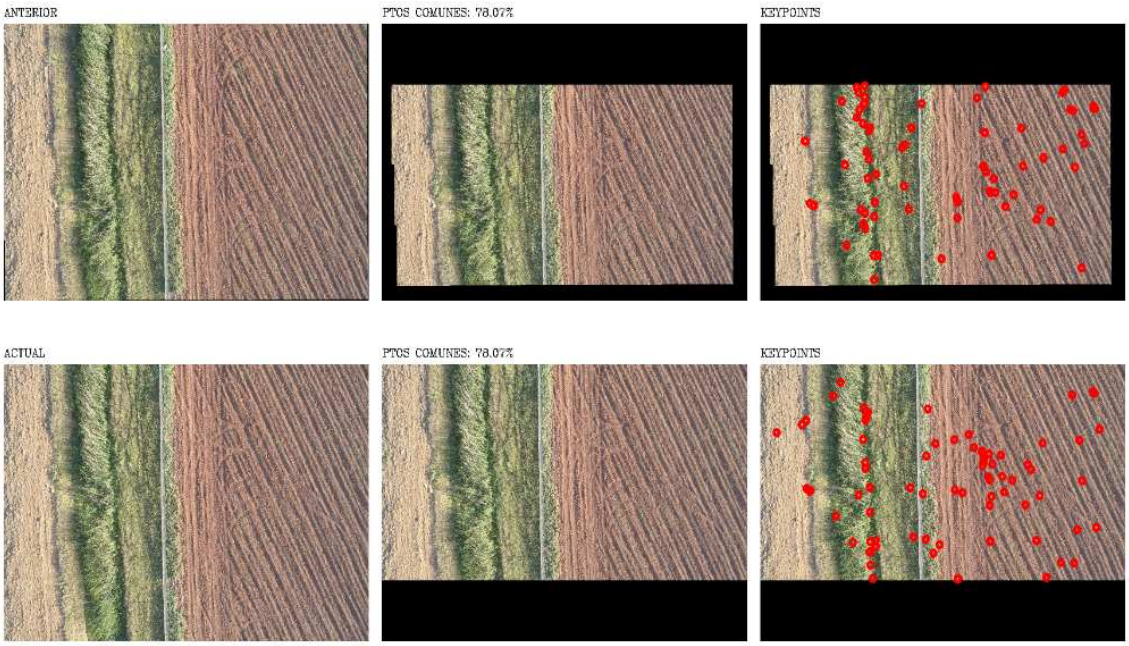

Figura 4.20 Resultado del algoritmo para las imágenes I_3 e I_4.

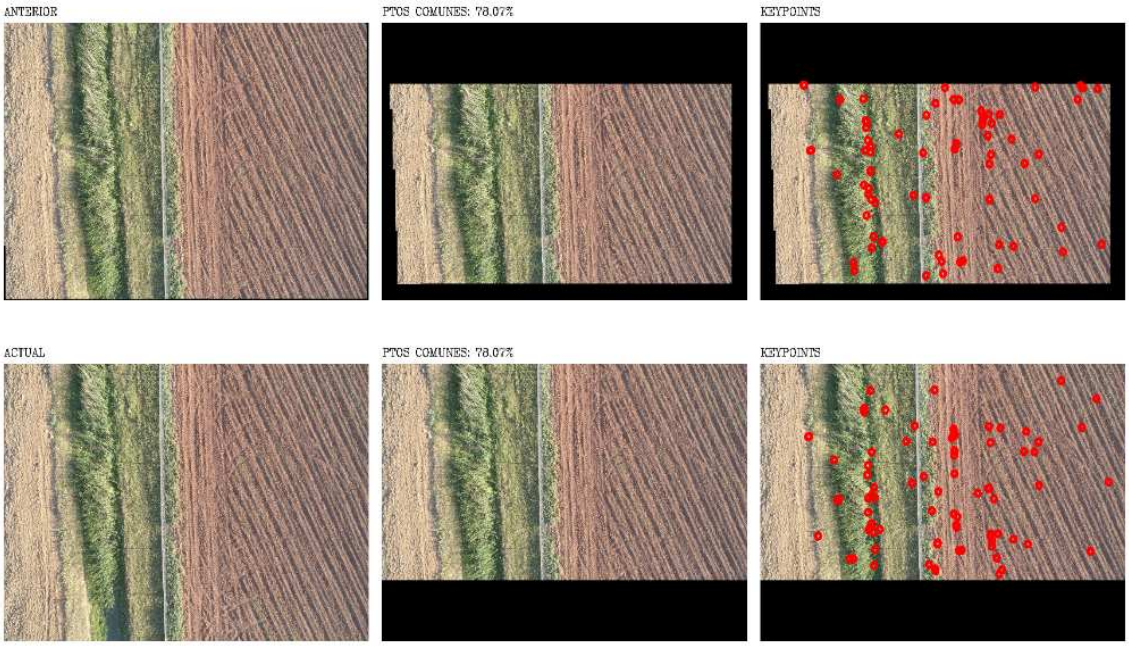

Figura 4.21 Resultado del algoritmo para las imágenes I_4 e I_5.

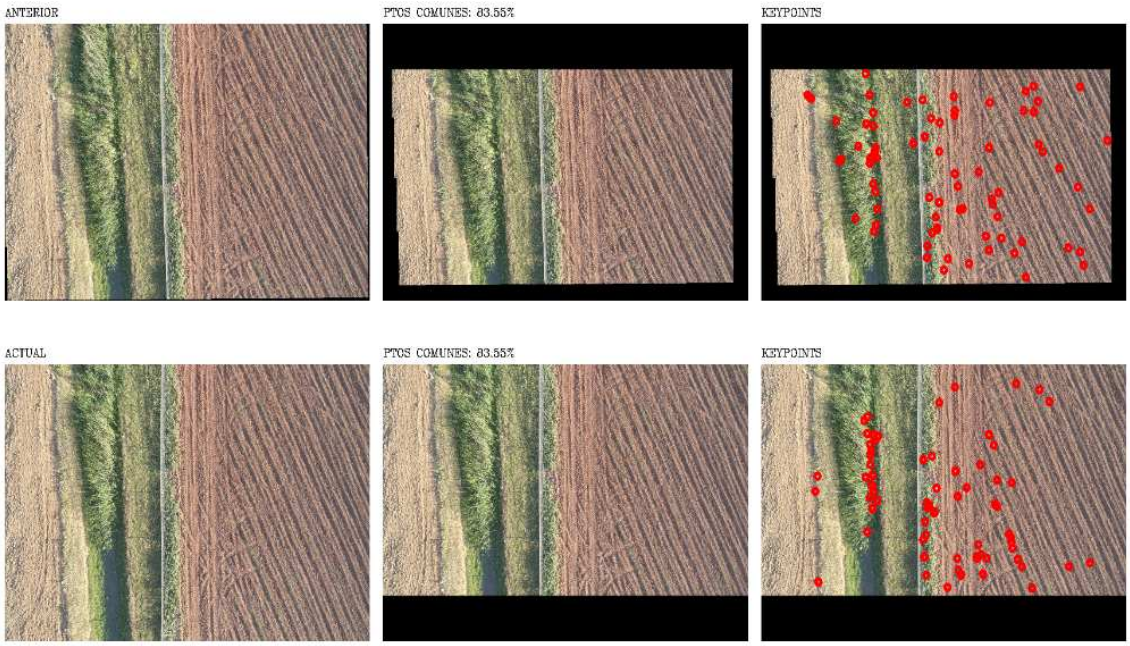

Figura 4.22 Resultado del algoritmo para las imágenes I_5 e I_6.

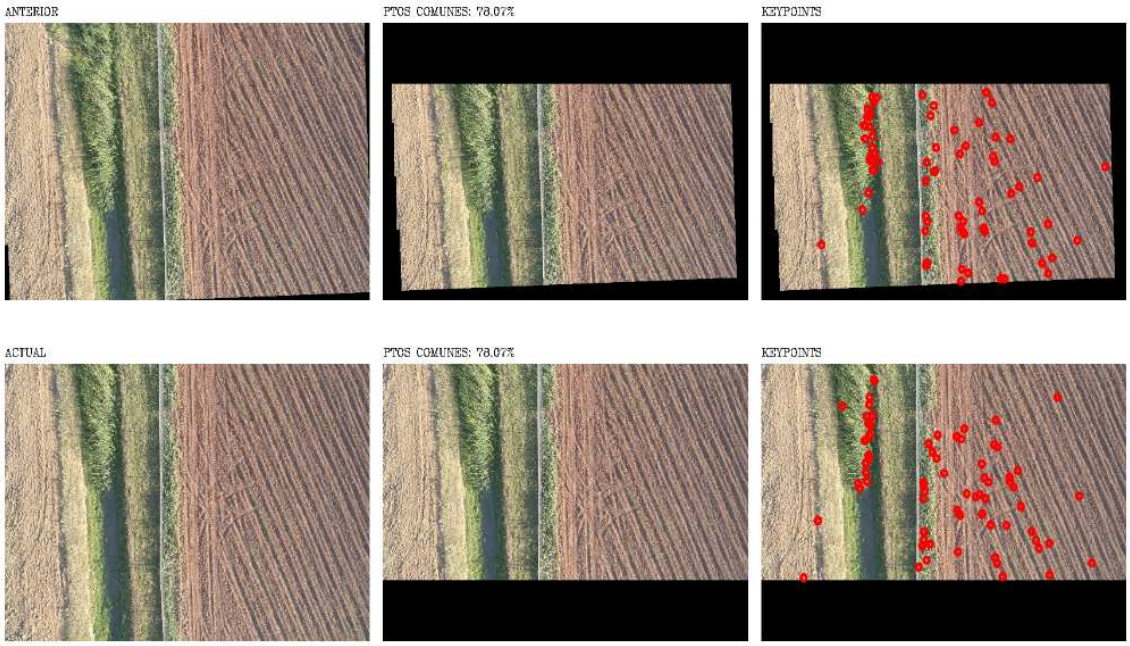

Figura 4.23 Resultado del algoritmo para las imágenes I_6 e I_7.

4.5 Conclusiones

Atendiendo a todo lo expuesto durante el desarrollo de este capítulo, puede afirmarse que el algoritmo implementado es bastante robusto frente a transformaciones, cambios de intensidad, ruido y distorsiones de las imágenes de entrada. El coste computacional de la detección y descripción es relativamente bajo en comparación con otros algoritmos y los resultados que ofrece poseen una calidad adecuada para una posterior correlación.

El número de *features* y el umbral de borde son parámetros que han sido mínimamente sobredimensionados, para asegurar un número de *matchings* suficientes, evitando el fallo del algoritmo en zonas donde exista una menor correlación entre imágenes. Esto no ha supuesto una pérdida de eficiencia ni un aumento del coste computacional considerables.

5 Aplicación de la Distancia de Hamming para la Correlación de *Features*

5.1 Introducción

En este capítulo se describirá el fundamento y el comportamiento del algoritmo implementado para obtener la correlación entre los *features* de dos imágenes. En un paso previo, el algoritmo ORB ha localizado los puntos de interés de cada una de las imágenes y los ha descrito, es decir, existe un vector descriptor asociado a cada *feature*.

Muchas aplicaciones obtienen esta correlación empleando *brute force matching* o correlación por fuerza bruta, consistente en comparar los descriptores de todos los puntos singulares de una imagen con todos los puntos singulares de la otra imagen. Sin embargo, este tipo de técnica requiere un gran coste computacional, sobre todo para relacionar un gran número de puntos. Por ello, se ha optado por implementar un *matcher* más eficiente, que obtenga la correlación mediante el cálculo de la distancia de Hamming de los descriptores de los puntos de interés y que tenga en cuenta la estimación del desplazamiento del UAV.

5.2 *Feature matching* mediante cálculo de la distancia de Hamming

Se conoce como *feature matching* o correlación de *features* al proceso mediante el cual se relacionan entre sí los puntos de interés de dos imágenes, emparejándose aquellos que correspondan al mismo punto del mundo real.

El descriptor rBRIEF (utilizado por ORB para describir los *features* detectados) es un descriptor binario, es decir, asocia a cada punto de interés un vector cuyos elementos son 0 o 1. Hacer uso de un descriptor binario permite implementar un *matching* muy eficiente, pues los descriptores pueden ser comparados utilizando la distancia de Hamming, una operación que puede ser interpretada como medida de la similitud entre dos puntos.

5.2.1 Cálculo de la distancia de Hamming

Se define la distancia de Hamming entre la cadena a y la cadena b como el número de bits que se deben cambiar en a para transformarla en b. Por ejemplo, la distancia de Hamming entre las dos cadenas de la Figura 5.1 es igual a 4.

Cadena A: 10101110

Cadena B: 10001001

Figura 5.1 Cadenas de 8 bits.

Para el caso binario, la distancia de Hamming puede ser definida como:

$$d(a,b) = \sum_{k=1}^{n} |a_k - b_k| \tag{5.1}$$

Sin embargo, se trata de un concepto extrapolable a cualquier tipo de cadena. Por ejemplo, la distancia de las cadenas de la Figura 5.2 es igual a 2.

Cadena A: tener
Cadena B: beber

Figura 5.2 Cadenas de caracteres.

La distancia de Hamming posee las siguientes propiedades:

$$d(a,b) = d(b,a) \tag{5.2}$$

$$d(a,b) = 0 \Leftrightarrow a = b \tag{5.3}$$

$$d(a,b) + d(b,c) \geq d(a,c) \tag{5.4}$$

Como puede comprobarse, esta distancia puede ser interpretada como un operador que permite medir la similitud entre dos descriptores.

5.2.2 Ventaja del uso de la distancia de Hamming

La ecuación 5.1 puede ser reescrita utilizando la operación XOR como:

$$d(a,b) = \text{bit\_count}(a \oplus b) \tag{5.5}$$

La disyunción exclusiva (XOR) de los conjuntos A y B es un conjunto $A \oplus B$ que comprende todos los elementos que pertenecen a A o a B, pero no a A y a B, es decir, $A \oplus B = (A \cup B) \setminus (A \cap B) = (A \cup B) \cap \overline{(A \cap B)}$.

Tabla 5.1 Tabla de verdad de la operación XOR.

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

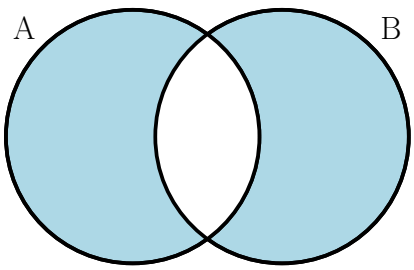

Figura 5.3 Resultado de la operación $A \oplus B$.

El hecho de poder calcular la distancia de Hamming de dos descriptores con esta operación es una gran ventaja en cuanto a coste computacional, ya que XOR es una de las operaciones más rápidas que un procesador puede llevar a cabo. Actualmente todas las CPUs soportan esta operación y son capaces de ejecutarla en menos de un ciclo de reloj. Además, muchos procesadores poseen la capacidad de ejecutar varios XOR en paralelo. Por otra parte, la operación *bit_count()* también pertenece al set de instrucciones de los procesadores modernos.

5.2.3 Umbral de la distancia de Hamming

Una vez obtenida la distancia entre los descriptores de dos *features*, es necesario determinar si esa unión debe ser considerada como *buena* o como *mala*. Para ello, se considerarán *buenas* uniones todas aquellas cuya distancia sea inferior a un valor umbral preestablecido y se desecharán las demás. Determinar este umbral de manera correcta es muy importante, pues un valor demasiado bajo podría provocar que se desechen uniones en exceso, mientras que un valor elevado supone la introducción de uniones que deteriorarán la calidad del mosaico.

Otra opción posible es escoger las n mejores correlaciones, es decir, aquellas n que presenten la menor distancia. Esta alternativa equivaldría a utilizar un umbral dinámico que se adapte a cada situación, lo que confiere al algoritmo una mayor robustez.

Escoger aquellas correlaciones con menor distancia hace innecesario un tratamiento de posibles *outliers*, evitándose todo el coste computacional requerido por algoritmos de eliminación de valores atípicos como RANSAC. Por ejemplo, en la Figura 5.4 puede comprobarse como utilizar una máscara para excluir de la extracción los puntos no comunes es innecesario, pues todos los puntos de interés extraídos en esas zonas quedarán sin correlación escogiendo el umbral adecuado. Aun así, el uso de estas máscaras se justifica por el hecho de aumentar la eficiencia y disminuir el tiempo de la extracción.

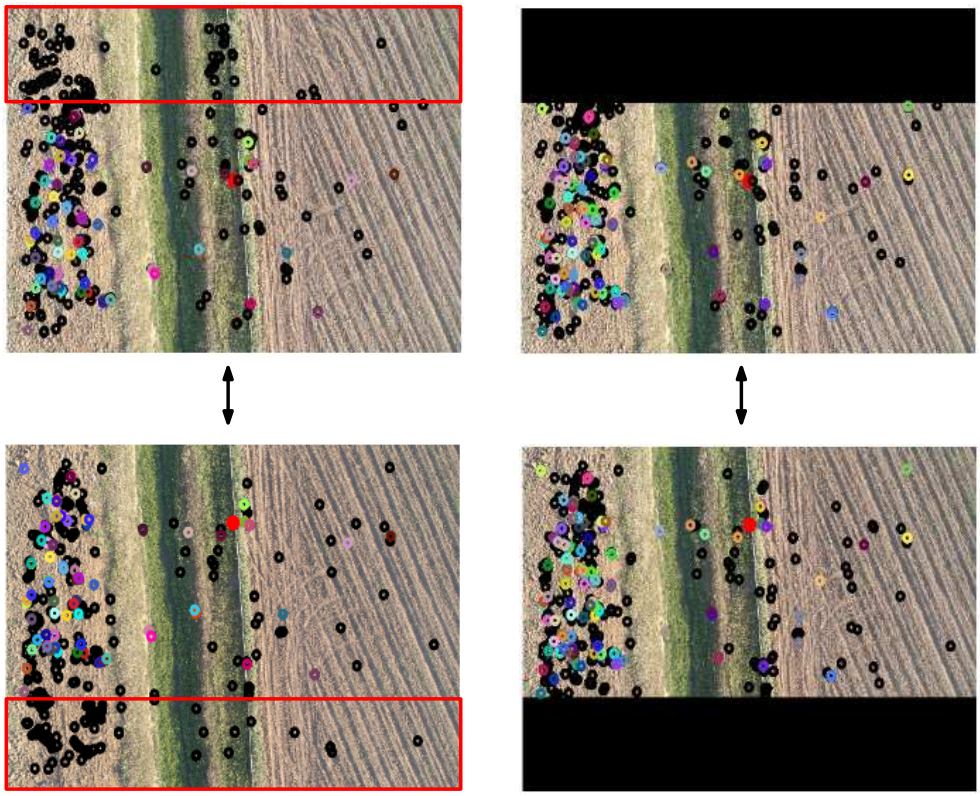

Figura 5.4 Comparativa de la correlación con y sin máscara (en color los puntos con correlación).

5.3 Reducción del tiempo de *matching* a partir de la estimación del desplazamiento

Una vez obtenida la forma de comparar dos descriptores, la correlación de los puntos singulares puede hallarse comparando todos los descriptores de las dos imágenes. Sin embargo, el coste computacional que se requiere para efectuar todas las comparaciones es relativamente elevado (por ejemplo, comparar 100 *features* de una imagen con 100 *features* de otra supone 10000 comparaciones y comparar 500 *features* con otros 500 supondría 250000 comparaciones).

Para reducir el tiempo de ejecución del algoritmo de *matching* se reducirá el número de comparaciones a realizar, para lo cual se hará uso de la estimación de la posición ofrecida por el GPS y de la medida de *yaw* del UAV. Sean $\Delta x = x_k - x_{k-1}$ e $\Delta y = y_k - y_{k-1}$ obtenidas a partir de las posiciones (x_k,y_k) y (x_{k-1},y_{k-1}) calculadas con las medidas del GPS (véase Anexo A), el desplazamiento en línea recta efectuado por el UAV entre el instante $k-1$ y k puede ser calculado como:

$$d = \sqrt{\Delta x^2 + \Delta y^2} \tag{5.6}$$

Conocida la medida de *yaw* respecto a una referencia, θ, y conocida la posición de un punto, (x_1,y_1), en la imagen I_k puede estimarse la posición que ocupó ese punto, (x_0,y_0) en la imagen I_{k-1}, tal y como se muestra en la Figura 5.7, de la siguiente forma:

$$(x_0,y_0) \approx (x_1 - d\sin\theta, y_1 - d\cos\theta) \tag{5.7}$$

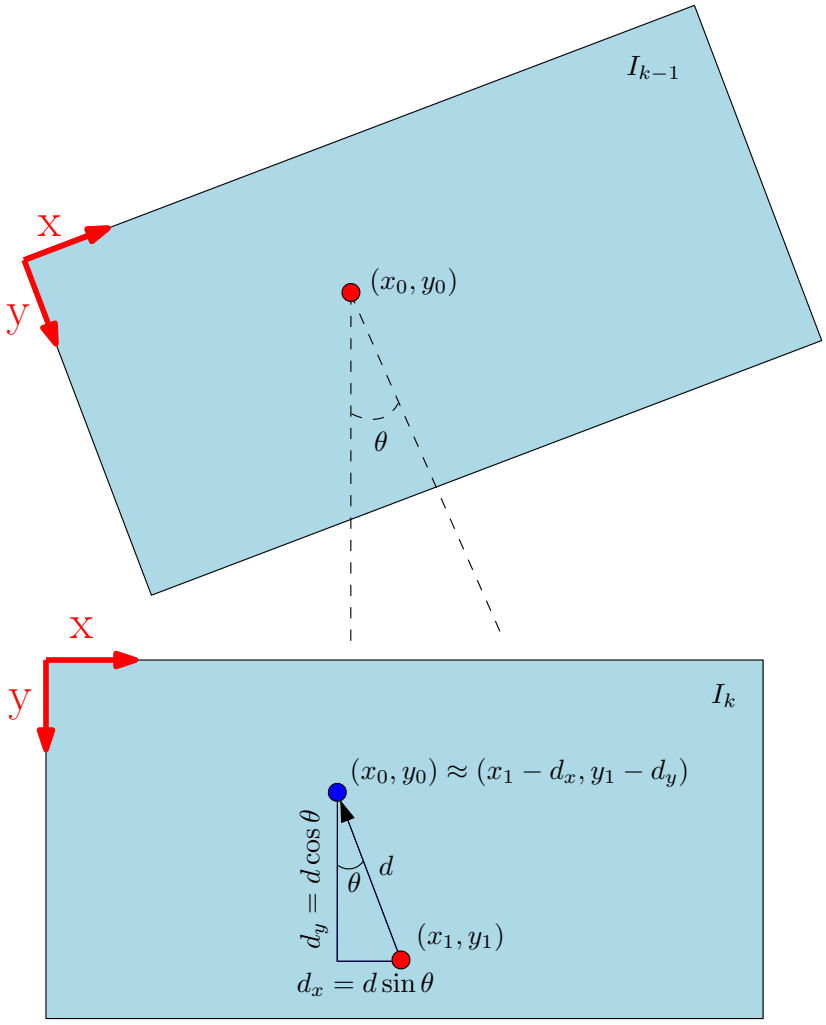

Figura 5.5 Estimación de desplazamiento a partir de GPS y *yaw*.

Al ser las medidas de posición y *yaw* muy erróneas, cada *features* se comparará con los *features* contenidos en una ventana cuya amplitud permita evitar que los errores de la estimación del desplazamiento afecten a la calidad del mosaico.

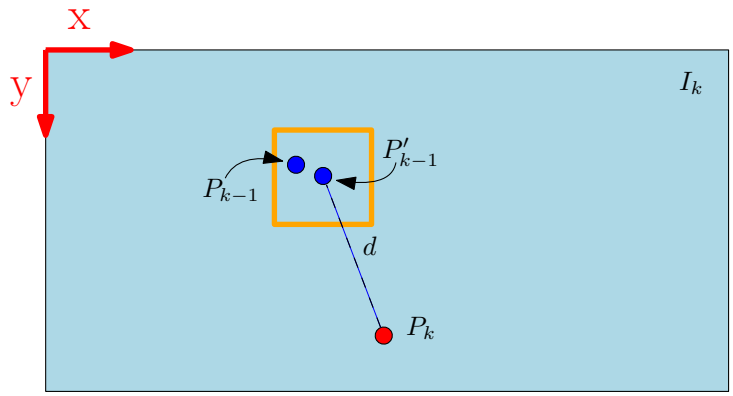

Figura 5.6 Comparación con puntos contenidos en una ventana (P_{k-1} es la posición real y P'_{k-1} la estimada).

Una alternativa al uso de la medida del *yaw*, que suele presentar un error muy significativo, es estimar la guiñada del UAV a partir del giro realizado en la última transformación. Con ello, no solo se obtendrán mejores resultados, sino que se evitará el tiempo dedicado a la lectura de esta medida.

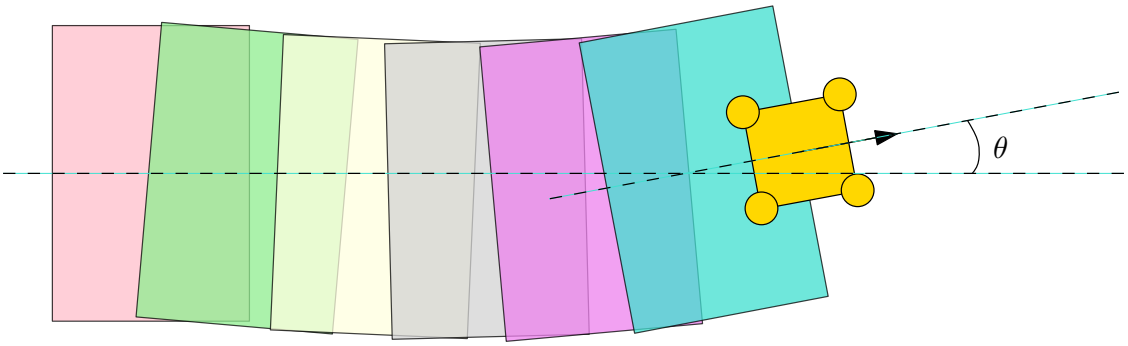

Figura 5.7 Estimación de *yaw* a partir de transformación.

5.4 Validación de los resultados

Sea I_k la imagen obtenida en adquisición k-ésima, I'_k la imagen obtenida en adquisición k-ésima transformada y situada en el mosaico, D^k el conjunto de descriptores de los puntos de interés de I_k, D'^k el conjunto de descriptores de I'_k, P_i^k las coordenadas del descriptor D_i^k, $\hat{P}_i^k$ las coordenadas estimadas a partir del GPS del descriptor D_i^k en I_{k+1}, W_i la ventana entorno al punto P_i y d un vector de distancias de Hamming, el algoritmo que ha sido implementado puede ser descrito mediante el diagrama de flujo de la Figura 5.8 o mediante el Pseudocódigo 5.1.

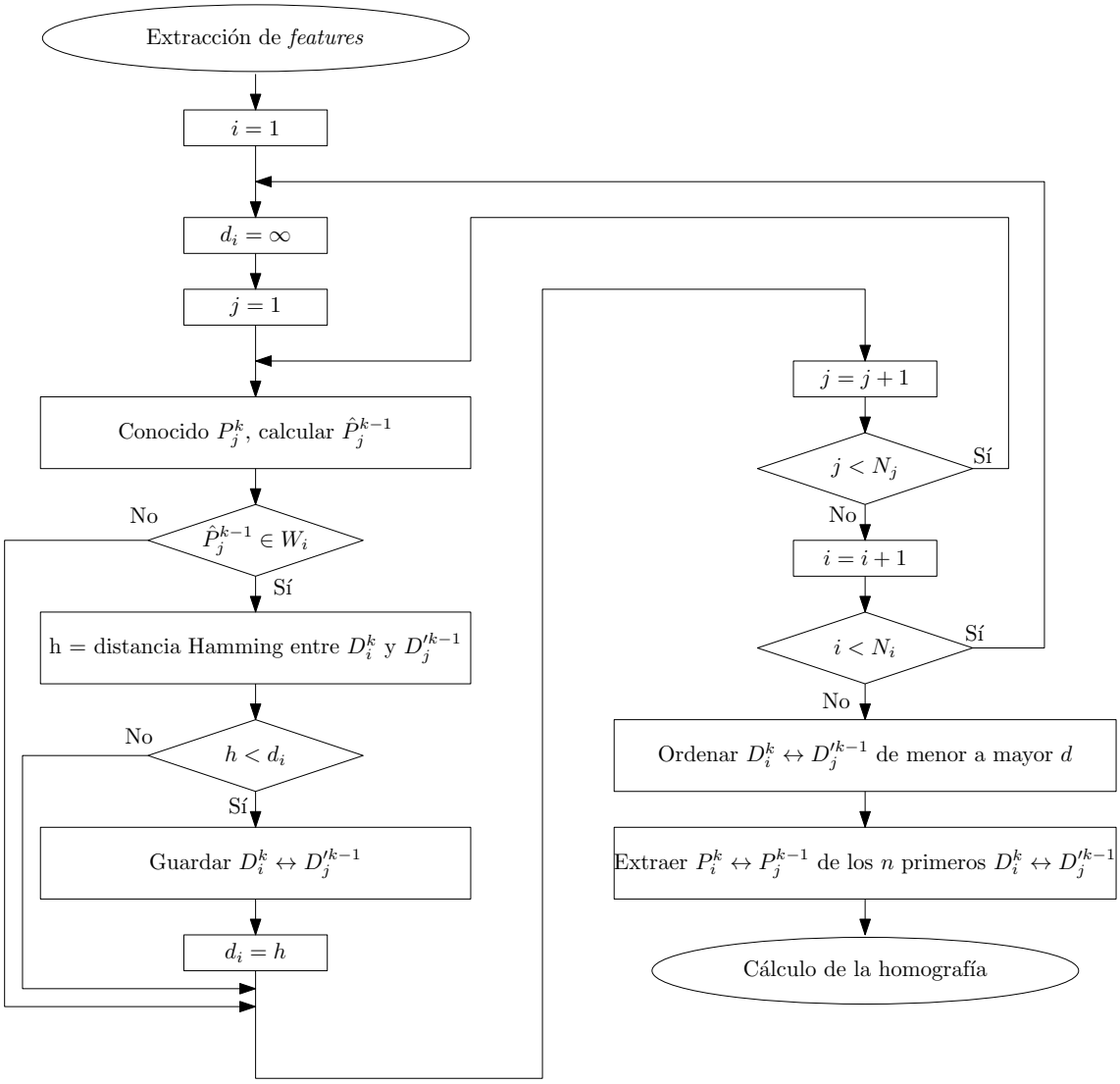

Figura 5.8 Algoritmo implementado para la correlación de *features*.

Código 5.1 Algoritmo de correlación de *features*.

```
PARA CADA i EN descriptores de I(k)
  Guardar en d(i) un valor muy elevado
  PARA CADA j EN descriptores de I'(k-1)
    Conocido P(j) en I(k), calcular P(j) estimada en I'(k-1)
    SI P(j) estimada está contenida en W(i)
      h = distancia de hamming entre i y j
      SI h es menor que d(i)
        Recordar que la mejor unión de i se da con j
        Guardar en d(i) h
      FIN SI
    FIN SI
  FIN PARA CADA
FIN PARA CADA
Ordenar las correlaciones recordadas de menor a mayor d
Extraer las coordenadas de las n primeras correlaciones
```

Para poner a prueba todo lo implementado hasta el momento, se utilizará un conjunto de siete imágenes. Los resultados presentados a continuación avalan el correcto funcionamiento del algoritmo. En todas las figuras:

- Los *features* de color negro son puntos sin correlación, o con una correlación cuya distancia de hamming es mayor que el umbral.
- Los *features* que no son de color negro son puntos sin correlación.
- La cruz roja de I_{k-1} es un punto tomado como referencia.
- La cruz azul de I_k es un punto con las mismas coordenadas que el punto de referencia de I_{k-1}.
- La cruz roja de I_k es la estimación de la posición del punto de referencia de I_{k-1}.
- El marco rojo de I_k es un ejemplo de ventana de búsqueda usada en cada correlación.

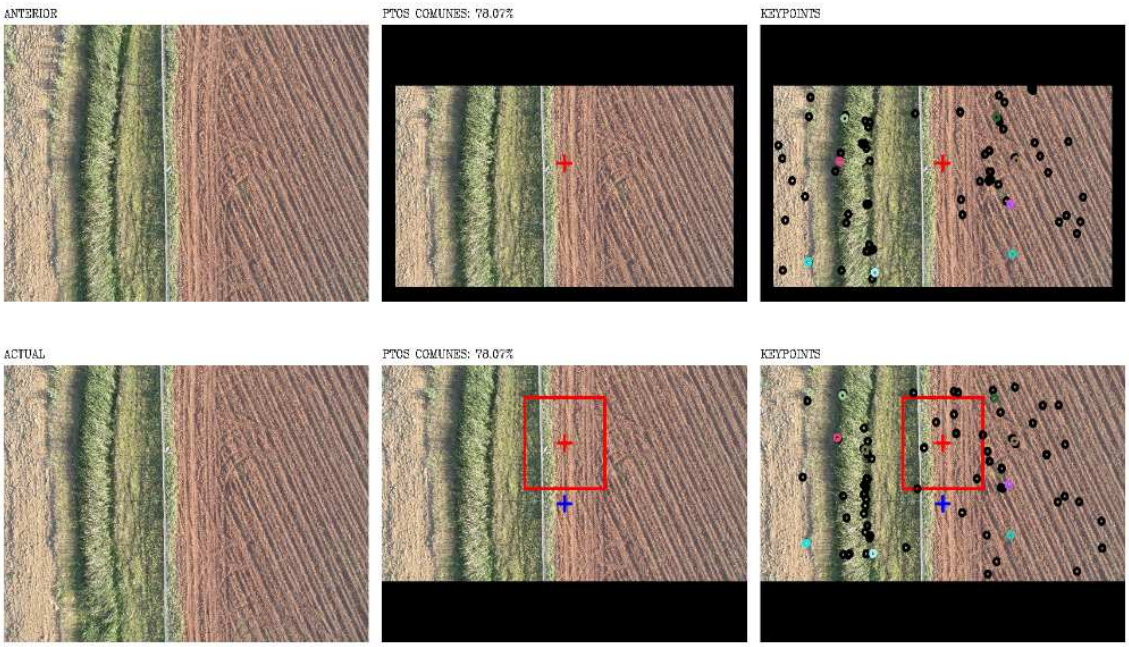

Figura 5.9 Resultado del algoritmo para las imágenes I_1 e I_2.

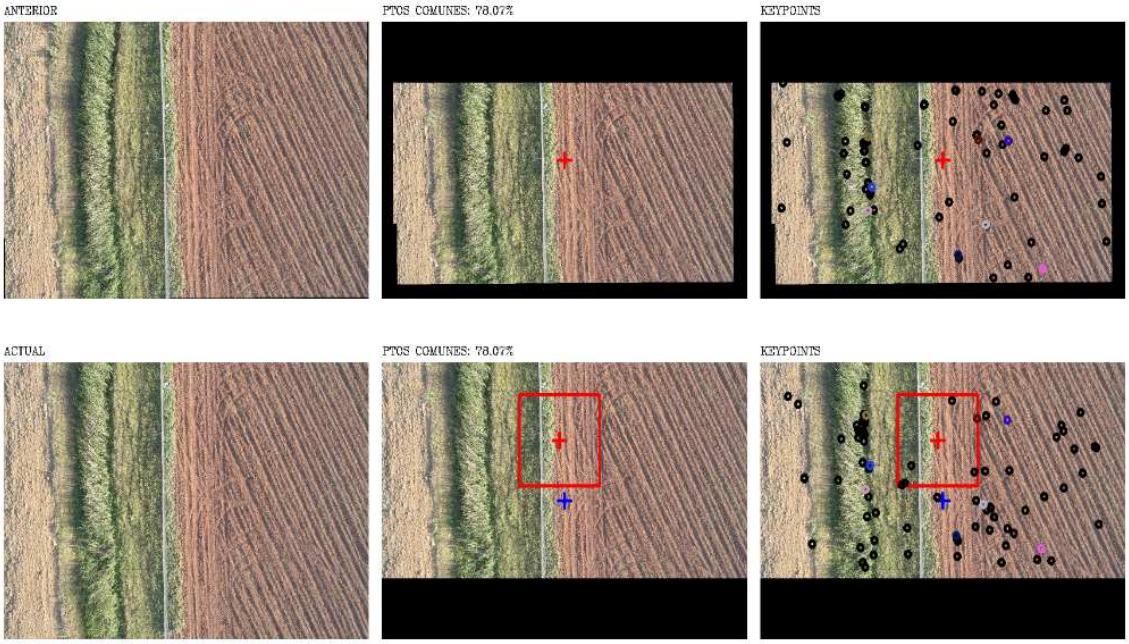

Figura 5.10 Resultado del algoritmo para las imágenes I_2 e I_3.

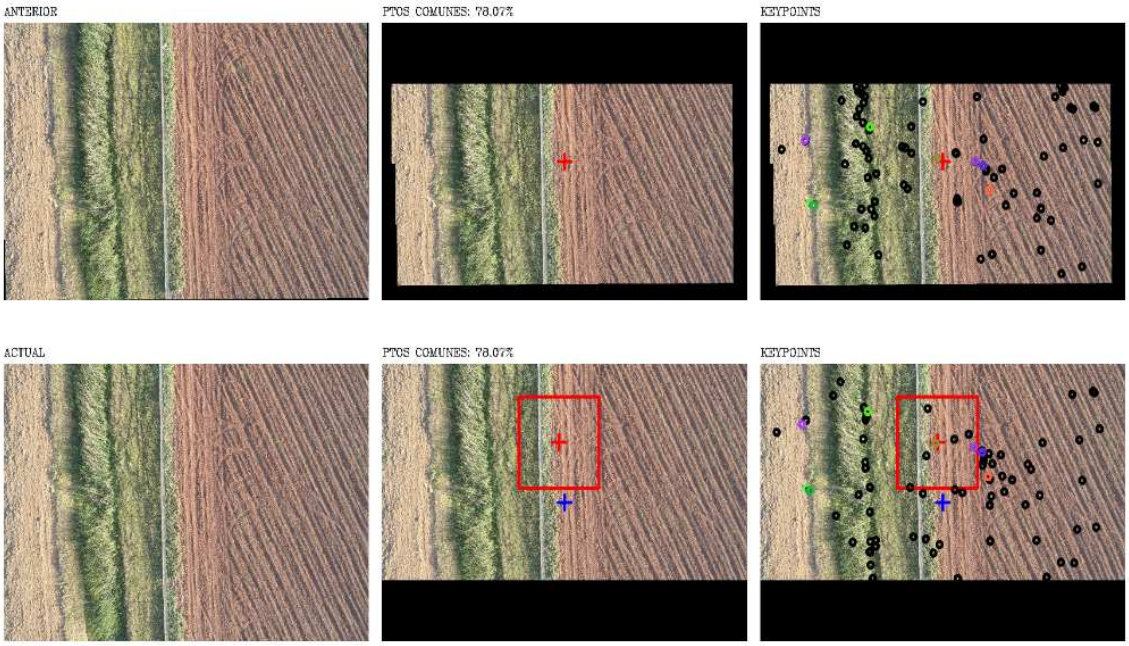

Figura 5.11 Resultado del algoritmo para las imágenes I_3 e I_4.

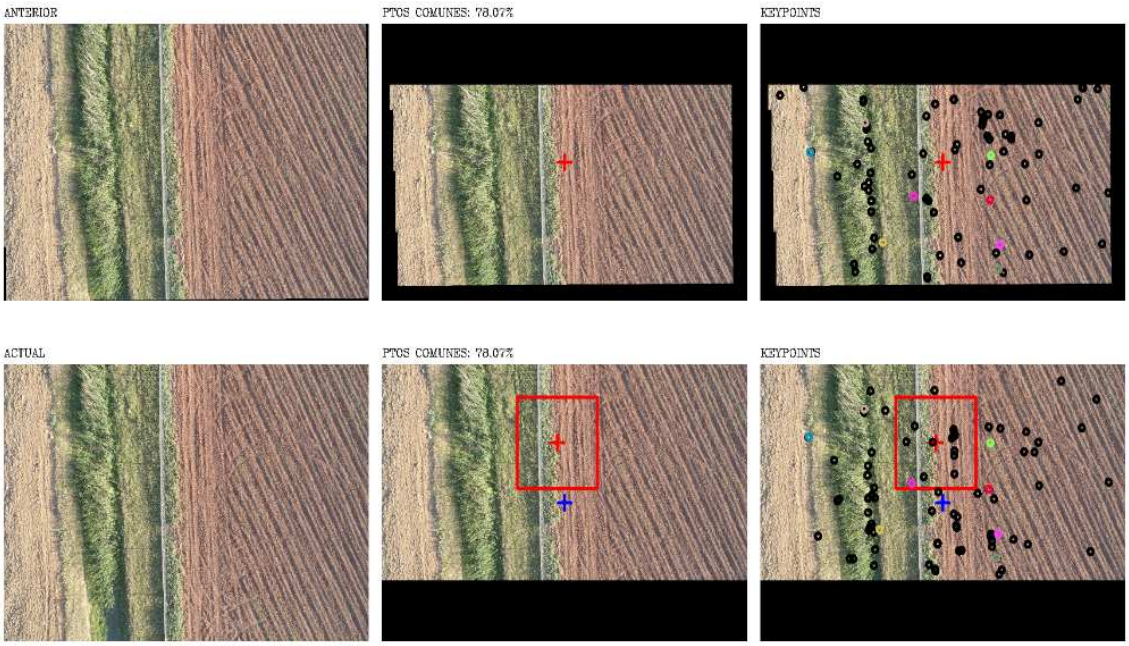

Figura 5.12 Resultado del algoritmo para las imágenes I_4 e I_5.

Figura 5.13 Resultado del algoritmo para las imágenes I_5 e I_6.

Figura 5.14 Resultado del algoritmo para las imágenes I_6 e I_7.

5.5 Conclusiones

Una de las ventajas de utilizar descriptores binarios es, como se ha comprobado en este capítulo, poder implementar la distancia de Hamming como una operación XOR, disminuyendo el tiempo de correlación. Si la descripción se realizase con algoritmos como SIFT o SURF, los vectores descriptores estarían compuestos por números reales, dejando de ser esta una opción rápida y eficiente.

El hecho de obtener en todo momentos las mejores correlaciones posibles no solo beneficia a la fase de *matching*, sino también repercute en las demás etapas:

- Pueden extraerse puntos de interés en una imagen que no existan en la otra, ya que serán descartados por no estar contenidos en ninguna ventana de correlación o por tener un distancia de Hamming muy elevada.
- Los errores del cálculo de la zona no común a dos imágenes no afectan a la unión, como se comprobó en la Figura 5.4.
- El algoritmo de correlación implementado supone un filtro de outliers, por lo que el cálculo de la transformación homogénea se realizará con conjuntos de puntos que permitirán obtener un muy buen ajuste.

Otra ventaja a tener en cuenta es que la amplitud de la ventana es proporcional al error de las medidas de posición admitido, por lo que el algoritmo puede ser adaptado en función de la calidad del GPS utilizado. Incluso podría ser adaptado para trabajar sin medidas GPS, como puede comprobarse en las figuras presentadas a continuación, obteniéndose una calidad similar pero siendo mucho más propenso a fallos.

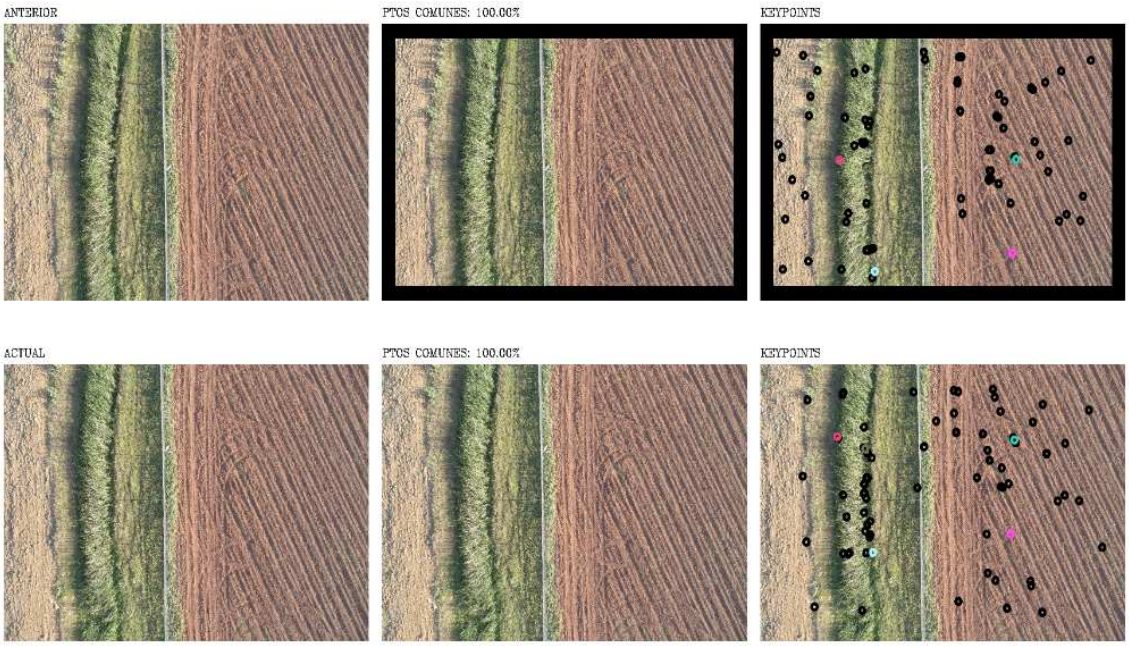

Figura 5.15 Resultado del algoritmo para las imágenes I_1 e I_2 sin medidas GPS.

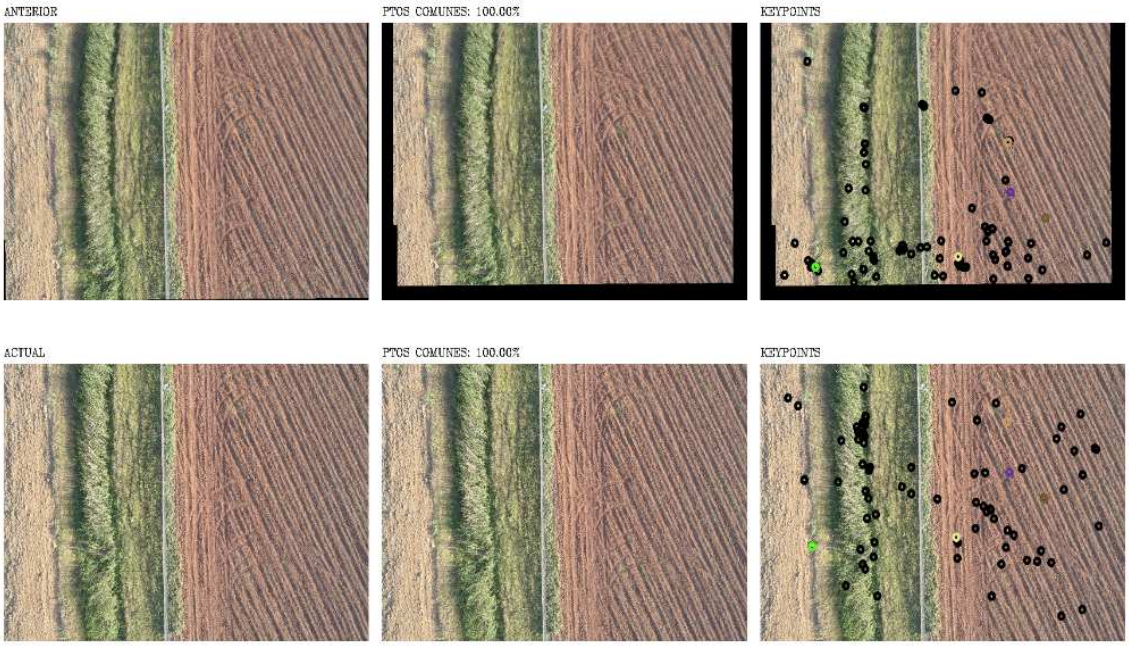

Figura 5.16 Resultado del algoritmo para las imágenes I_2 e I_3 sin medidas GPS.

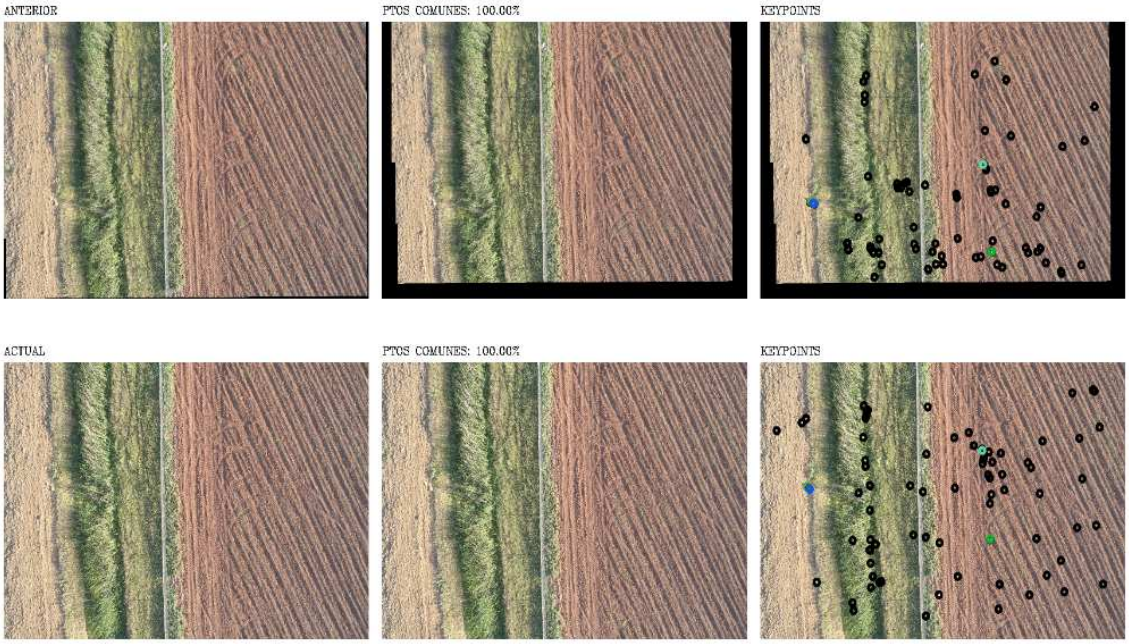

Figura 5.17 Resultado del algoritmo para las imágenes I_3 e I_4 sin medidas GPS.

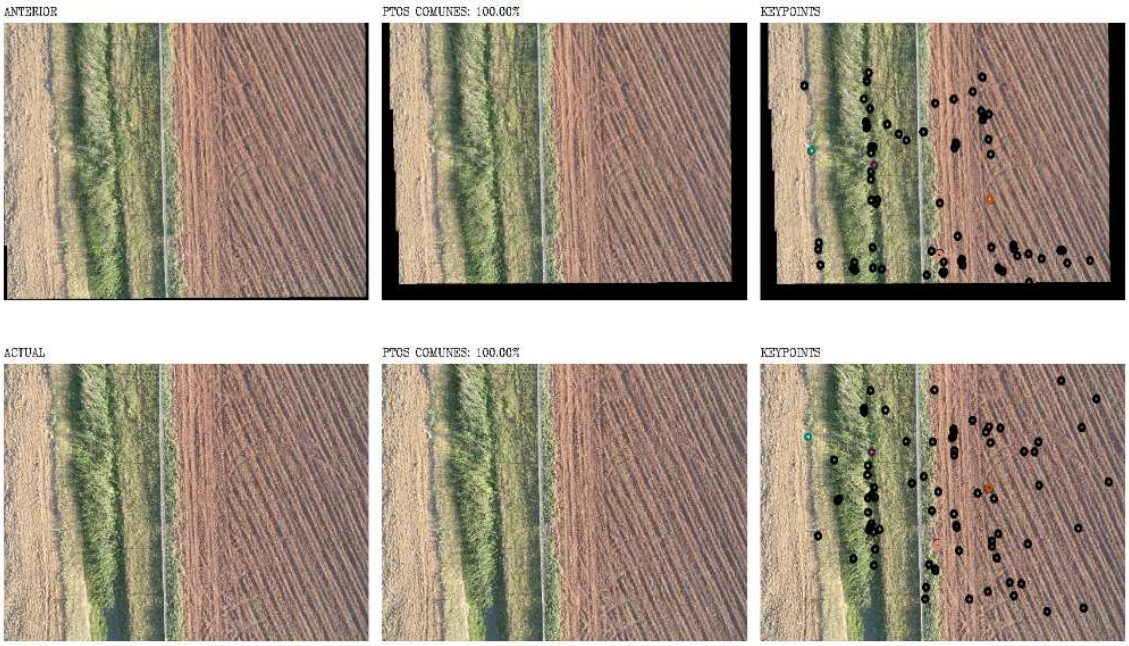

Figura 5.18 Resultado del algoritmo para las imágenes I_4 e I_5 sin medidas GPS.

Figura 5.19 Resultado del algoritmo para las imágenes I_5 e I_6 sin medidas GPS.

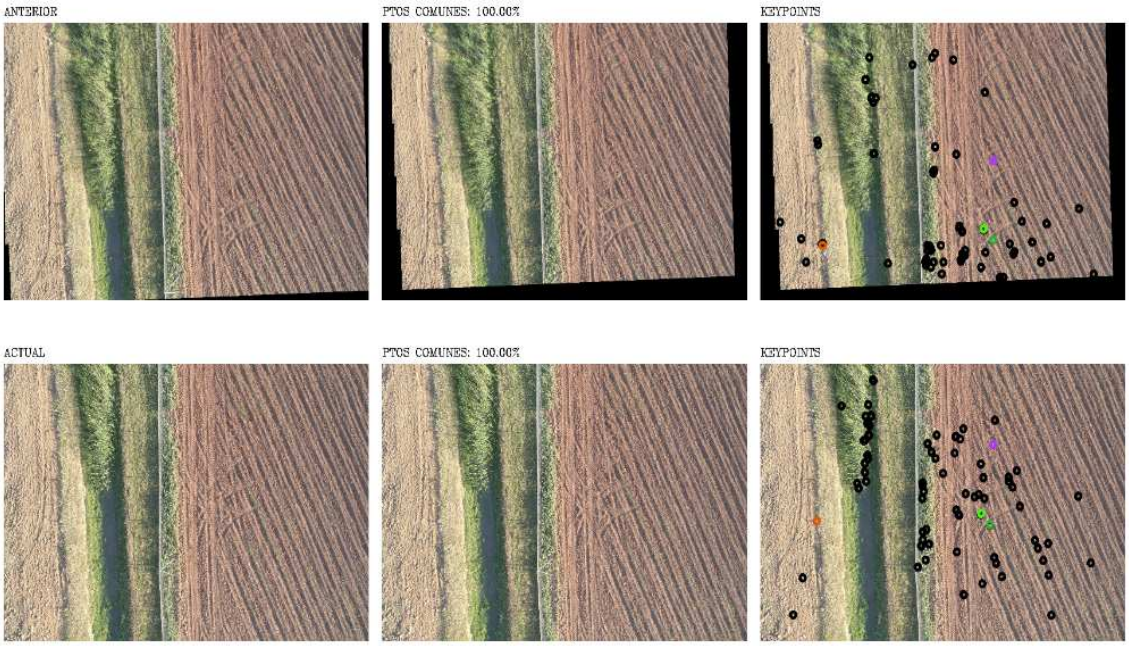

Figura 5.20 Resultado del algoritmo para las imágenes I_6 e I_7 sin medidas GPS.

6 Cálculo de la Transformación Homogénea que Relaciona las Imágenes

6.1 Introducción

El objeto de este capítulo es encontrar la transformación que relaciona los dos conjuntos de puntos correlacionados en el capítulo anterior, calculando para ello los parámetros que proporcionen el mejor ajuste del modelo mediante la minimización del error cuadrático. La estimación de la homografía planteará un problema sobredeterminado de la forma $Ax = 0$, que será resuelto en el sentido de mínimos cuadrados mediante la descomposición en valores simples de A. Aplicando la transformación obtenida, se conseguirá el mejor encaje posible entre los dos conjuntos de puntos y, por tanto, la unión de las dos imágenes.

6.2 Coordenadas homogéneas y matriz de transformación homogénea

En $\mathbb{R}^2$, un punto $P = (x,y)^\top$ puede ser expresado en coordenadas homogéneas (añadiendo una tercera componente) como $P = (sx,sy,s)^\top$, donde la nueva coordenada representa un factor de escala. En lo sucesivo, se supondrá $s = 1$ sin perder generalidad.
Empleando coordenadas homogéneas, la transformación entre el punto $x_1 \in \mathbb{R}^2$ y $x_2 \in \mathbb{R}^2$ vendrá dada por $x_2 = Tx_1$, donde T es la matriz de transformación :

$$T = \begin{bmatrix} R_{11} & R_{12} & p_x \\ R_{21} & R_{22} & p_y \\ ff_1 & f_2 & w \end{bmatrix} = \begin{bmatrix} R_{(2x2)} & p_{(2x1)} \\ f_{(1x2)} & w_{(1x1)} \end{bmatrix} \tag{6.1}$$

Siendo $R_{(2x2)}$, $p_{(2x1)}$, $f_{(1x2)}$, $w_{(1x1)}$ respectivamente la matriz de rotación, de traslación, de perspectiva y de escalado del punto transformado.

Las condiciones de vuelo y demás hipótesis detalladas en el Apartado 1.2.1, permiten aproximar la matriz de transformación de la forma:

$$T = \begin{bmatrix} R_{11} & R_{12} & p_x \\ R_{21} & R_{22} & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{6.2}$$

Lo que simplifica la resolución del problema y disminuye el coste computacional de su implementación.

6.3 Solución al problema de mínimos cuadrados

El ajuste por mínimos cuadrados es una técnica de análisis numérico que permite obtener los parámetros que proveen el mejor ajuste de un modelo a un conjunto de datos, entendiéndose como *mejor ajuste* aquel que minimice el error cuadrático medio.

Sea $Ax = b$ un sistema sobredeterminado en el que $\nexists x \mid Ax - b = 0$, se define el error o residuo de $Ax \approx b$ como $e = b - Ax$. El error cuadrático medio viene dado por la expresión:

$$E_{cm} = \sqrt{\frac{\sum_{k=1}^{N} e_k^2}{N}} = \sqrt{\frac{\| e \|}{N}} = \sqrt{\frac{e^\top e}{N}} \tag{6.3}$$

Minimizar la expresión anterior equivale a minimizar $e^\top e$, que a su vez equivale a minimizar $\frac{1}{2}e^\top e$:

$$\frac{1}{2}e^\top e = \frac{1}{2}(b-Ax)^\top(b-Ax) = \frac{1}{2}(b^\top - x^\top A^\top)(b-Ax) = \frac{1}{2}(b^\top b - x^\top A^\top b - b^\top Ax + x^\top A^\top Ax) \tag{6.4}$$

Al ser $A \in \mathbb{R}^{M\times N}$, $x \in \mathbb{R}^{N\times 1}$ y $b \in \mathbb{R}^{M\times 1}$; el producto $x^\top A^\top b \in \mathbb{R}^{1\times 1}$ y el producto $b^\top Ax \in \mathbb{R}^{1\times 1}$. Por tanto, se cumple que $x^\top A^\top b = b^\top Ax$.

$$\frac{1}{2}e^\top e = \frac{1}{2}(x^\top A^\top Ax - 2b^\top Ax + b^\top b) = \frac{1}{2}x^\top A^\top Ax - b^\top Ax + \frac{1}{2}b^\top b \tag{6.5}$$

$$\arg\min_x\{\frac{1}{2}e^\top e\} = \arg\min_x\{\frac{1}{2}x^\top A^\top Ax - b^\top Ax + \frac{1}{2}b^\top b\} = \arg\min_x\{\frac{1}{2}x^\top A^\top Ax - b^\top Ax\} \tag{6.6}$$

Efectuando los cambios de variable $\alpha = A^\top A$ y $\beta = b^\top A$:

$$\arg\min_x\{\frac{1}{2}e^\top e\} = \arg\min_x\{\frac{1}{2}x^\top \alpha x - \beta x\} \tag{6.7}$$

Sea $f = \frac{1}{2}x^\top \alpha x - \beta x$, el valor x^* que minimiza f será aquel que anule su derivada:

$$\frac{df}{dx} = \alpha x - \beta \tag{6.8}$$

$$\alpha x^* - \beta = 0 \rightarrow x^* = \alpha^{-1}\beta \tag{6.9}$$

Deshaciendo los cambios de variable:

$$x^* = (A^\top A)^{-1}b^\top A \tag{6.10}$$

Expresión válida si y solo si $\exists (A^\top A)^{-1}$.

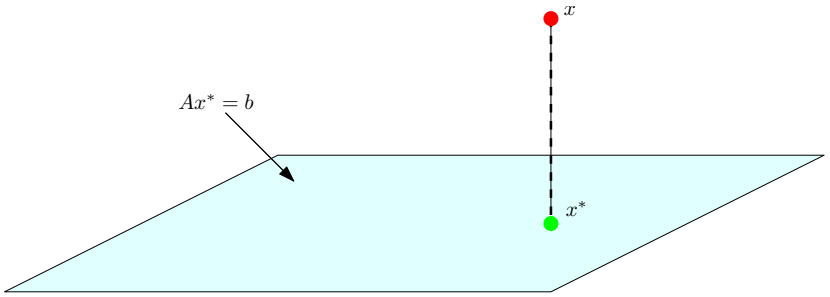

Figura 6.1 Representación del problema de mínimos cuadrados.

La transformación dada por $x_2 = Tx_1$ es una ecuación matricial de la forma $b = Ax$, sin embargo, al ser T la matriz desconocida, será necesario reescribir la expresión de forma que pueda ser resuelta según lo desarrollado en este apartado.

6.4 Estimación de la Homografía

Partiendo de la ecuación:

$$x_2 = Tx_1 \rightarrow \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & p_x \\ R_{21} & R_{22} & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{6.11}$$

Se pueden extraer las expresiones:

$$x_2 = R_{11}x_1 + R_{12}y_1 + p_x \tag{6.12}$$

$$y_2 = R_{21}x_1 + R_{22}y_1 + p_y \tag{6.13}$$

Si se quiere resolver el problema de mínimos cuadrados para obtener los parámetros de T, se necesita reescribir las ecuaciones 6.12 y 6.13 de la forma $Ax = 0$:

$$\begin{aligned} R_{11}x_1 + R_{12}y_1 + p_x - x_2 = 0 \\ R_{21}x_1 + R_{22}y_1 + p_y - y_2 = 0 \end{aligned} \tag{6.14}$$

$$\begin{aligned} [x_1, y_1, 1, -x_2][R_{11}, R_{12}, p_x, 1]^\top = 0 \\ [x_1, y_1, 1, -y_2][R_{21}, R_{22}, p_y, 1]^\top = 0 \end{aligned} \tag{6.15}$$

$$\begin{aligned} [x_1, y_1, 0, 0, 1, 0, -x_2][R_{11}, R_{12}, R_{21}, R_{22}, p_x, p_y, 1]^\top = 0 \\ [0, 0, x_1, y_1, 0, 1, -y_2][R_{11}, R_{12}, R_{21}, R_{22}, p_x, p_y, 1]^\top = 0 \end{aligned} \tag{6.16}$$

$$\begin{bmatrix} x_1 & y_1 & 0 & 0 & 1 & 0 & -x_2 \\ 0 & 0 & x_1 & y_1 & 0 & 1 & -y_2 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{21} & R_{22} & p_x & p_y & 1 \end{bmatrix}^\top = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{6.17}$$

La ecuación 6.17 es un sistema homogéneo sobredeterminado, que será resulto en el sentido de los mínimos cuadrados. Sea $Ax \approx 0$, se define el error como $e = Ax$. Por lo visto en el Apartado 6.3, minimizar el error cuadrático medio equivale a minimizar $\frac{1}{2}e^\top e$:

$$\frac{1}{2}e^\top e = \frac{1}{2}(Ax)^\top (Ax) = \frac{1}{2}x^\top A^\top Ax \tag{6.18}$$

$$\frac{d}{dx}(\frac{1}{2}e^\top e) = A^\top Ax \tag{6.19}$$

$$A^\top Ax^* = 0 \tag{6.20}$$

Por tanto, la solución que provoca el menor error en $Ax \approx 0$ es $x^* \mid A^\top Ax^* = 0$.

6.4.1 Particularización de la matriz de rotación

Atendiendo a la ecuación 6.17, el cálculo de la transformación implica obtener los seis parámetros que conforman la matriz T $(R_{11}, R_{12}, R_{21}, R_{22}, p_x, p_y)$. Sin embargo, pueden reducirse el número de incógnitas suponiendo que la rotación viene dada por un giro en *yaw*:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{6.21}$$

De forma que la matriz de transformación puede ser reescrita como:

$$T = \begin{bmatrix} \cos\theta & -\sin\theta & p_x \\ \sin\theta & \cos\theta & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{6.22}$$

Por tanto:

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & p_x \\ \sin\theta & \cos\theta & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{6.23}$$

$$\begin{aligned} x_2 &= x_1\cos\theta - y_1\sin\theta + p_x \\ y_2 &= x_1\sin\theta + y_1\cos\theta + p_y \end{aligned} \tag{6.24}$$

$$\begin{aligned} x_1\cos\theta - y_1\sin\theta + p_x - x_2 &= 0 \\ x_1\sin\theta + y_1\cos\theta + p_y - y_2 &= 0 \end{aligned} \tag{6.25}$$

$$\begin{aligned} [x_1, -y_1, 1, -x_2][\cos\theta, \sin\theta, p_x, 1]^\top &= 0 \\ [x_1, y_1, 1, -y_2][\sin\theta, \cos\theta, p_y, 1]^\top &= 0 \end{aligned} \tag{6.26}$$

$$\begin{aligned} [x_1, -y_1, 1, 0, -x_2][\cos\theta, \sin\theta, p_x, p_y, 1]^\top &= 0 \\ [y_1, x_1, 0, 1, -y_2][\cos\theta, \sin\theta, p_x, p_y, 1]^\top &= 0 \end{aligned} \tag{6.27}$$

$$\begin{bmatrix} x_1 & -y_1 & 1 & 0 & -x_2 \\ y_1 & x_1 & 0 & 1 & -y_2 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & p_x & p_y & 1 \end{bmatrix}^\top = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{6.28}$$

La ecuación 6.28 tiene la forma $Ax = 0$, por lo que su resolución en el sentido de mínimos cuadrados viene dada por la expresión 6.20. Sin embargo, la minimización del error debe estar sujeta a la restricción $\cos^2\theta + \sin^2\theta = 1$.

6.4.2 Resolución de $A^\top Ax^* = 0$ mediante SVD

El problema de mínimos cuadrados $Ax = b$ puede ser resuelto utilizando la pseudoinversa de A. Sin embargo, esta resolución no puede emplearse para el problema homogéneo $Ax = 0$. Una alternativa para resolverlo es emplear la SVD de A [28] [29]. La descomposición en valores singulares o SVD (*Singular Value Decomposition*) de $A \in \mathbb{R}^{M\times N}$ es una factorización de la forma:

$$A = U\Sigma V^\top \tag{6.29}$$

Con $U \in \mathbb{R}^{M\times M}$ ortogonal, $V \in \mathbb{R}^{N\times N}$ ortogonal y $\Sigma \in \mathbb{R}^{M\times N}$ una matriz diagonal formada por los valores singulares de A (por convención, ordenados de mayor a menor). Si el valor singular más pequeño (σ_{min}) es nulo, el sistema $Ax = 0$ está determinado (existe una solución que une los puntos con total exactitud). Cuando el valor singular más pequeño sea mayor que cero, el sistema estará sobredeterminado y la solución en el sentido de los mínimos cuadrados ofrecerá menos error cuanto más cercano a cero sea. Por otra parte, un σ_{min} menor que cero implica un sistema indeterminado.

Atendiendo a la ecuación a resolver, $A^\top Ax^* = 0$, x^* debe ser igual al autovector de $A^\top A$ que tenga asociado un autovalor igual a cero, o en su defecto, el autovalor más cercano a cero. El autovector de $A^\top A$ buscado coincide con la columna de V de la SVD de A situada más a la derecha (es decir, la columna correspondiente al valor singular más pequeño, por estar ordenados de mayor a menor) por el siguiente motivo: *dada la matriz A con la descomposición en valores singulares $A = U\Sigma V^\top$, las columnas de V se corresponden con los autovectores de $A^\top A$* [29].

6.4.3 Obtención de la matriz de transformación homogénea

La solución $x^* \mid A^\top Ax^* = 0$ contiene los parámetros de la transformación T tales que se obtenga el mejor ajuste entre los puntos. El último paso es acomodar el vector x^* en una matriz 3×3:

$$x^* = \begin{bmatrix} \cos\theta & \sin\theta & p_x & p_y & 1 \end{bmatrix}^\top \rightarrow T = \begin{bmatrix} \cos\theta & -\sin\theta & p_x \\ \sin\theta & \cos\theta & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{6.30}$$

6.5 Uso de la transformación calculada

Una vez calculada la matriz T a partir de los puntos correlacionados de las imágenes I_k e I_{k-1}, se aplica la transformación a todos los puntos de I_k.

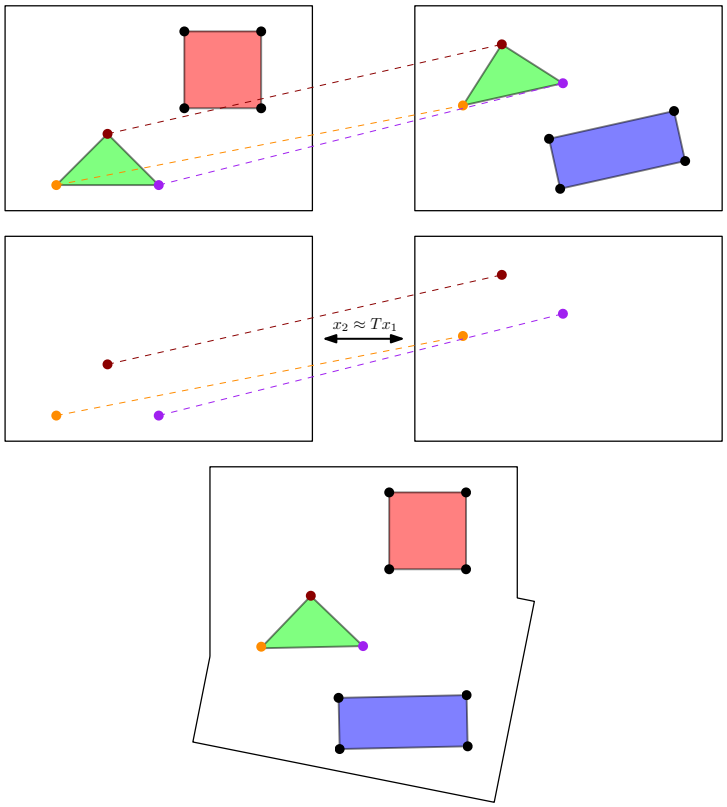

Figura 6.2 Aplicación de la transformación calculada a todos los puntos de una imagen.

En un paso previo a la transformación de los puntos de la imagen, debe ser tenido en cuenta el espacio de memoria requerido. Una solución es ofrecer al mosaico un espacio dinámico que crezca conforme este aumente de tamaño, mientras que otra posible solución es estimar el tamaño final del mosaico (siempre aplicando las tolerancias o sobreestimaciones necesarias) y que este ocupe un espacio estático desde el primer momento.

6.5.1 Interpolación

La composición del mosaico presentada se realiza mediante la superposición de las imágenes. Una posible alternativa es componer el mosaico mediante la interpolación de los puntos coincidentes de las zonas comunes. La ventaja de interpolar es que las variaciones de intensidad luminosa entre imágenes se atenúan, mientras que su principal inconveniente es que acentúa los errores de unión. El hecho de poner de manifiesto el error de los ajustes será utilizado en el Capítulo 7 para analizar la calidad del resultado obtenido.

Existen distintos métodos de interpolación. El escogido para la formación del mosaico, por ser el que provoca menos emborronamiento de la imagen, es el algoritmo de interpolación lineal. Sea $I_k(i,j)$ un punto de la imagen I_k, el resultado de interpolarlo linealmente con el punto $I_{k-1}(i,j)$ viene dado por la siguiente expresión:

$$M = W_1 I_k(i,j) + W_2 I_{k-1}(i,j) \tag{6.31}$$

Con $W_1 \in [0,1]$ y $W_2 \in [0,1]$ pesos de la interpolación.

La interpolación puede suponer una disminución de la robustez del algoritmo y llegar a provocar fallos si en la escena inspeccionada existen objetos en movimiento (por ejemplo, el agua de un rio o plantas movidas por el viento) u objetos a distintas alturas. Pese a realizar las capturas a una altura suficientemente elevada como para poder validar la hipótesis de tierra plana, la interpolación podría provocar efectos indeseados consecuencia del cambio de perspectiva. Por todo ello, la decisión de emplear o no interpolación debe basarse en las condiciones de la zona a inspeccionar.

6.6 Validación de los resultados

Sea I_k la imagen obtenida en adquisición *k*-ésima, I'_k la imagen obtenida en adquisición *k*-ésima transformada y situada en el mosaico, P_k el conjunto de puntos correlacionados con P_{k-1}, $P[i]$ el elemento *i*-ésimo de *P* y N_P el número de correlaciones, el algoritmo que ha sido implementado puede ser descrito mediante el diagrama de flujo de la Figura 6.3 o mediante el Pseudocódigo 6.1.

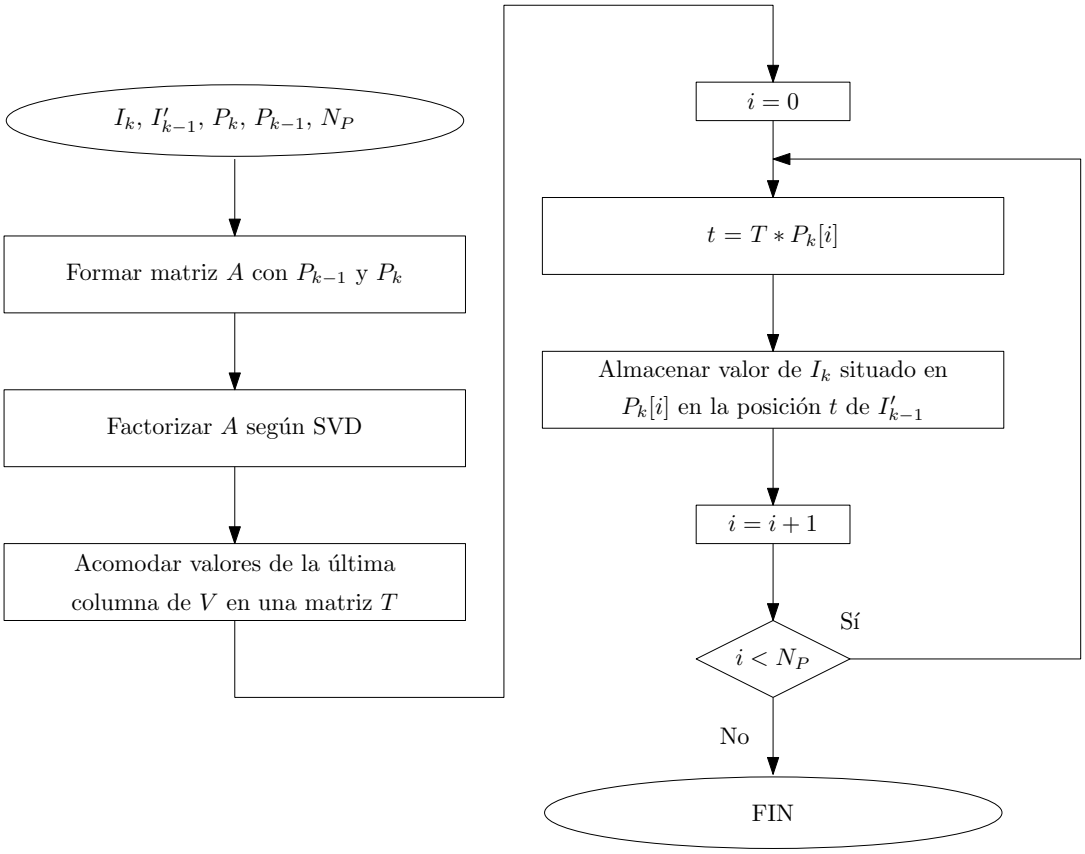

Figura 6.3 Algoritmo implementado para el cálculo y aplicación de la transformación.

Código 6.1 Algoritmo para el cálculo y aplicación de la transformación.

```
Formar matriz A con P(k-1) y P(k)
Calcular SVD de A
Acomodar valores de la última columna de V en T
PARA CADA p EN P(k)
  t = T*p
  I'(k-1) en t = I(k) en p
FIN PARA CADA
```

Para poner a prueba el algoritmo implementado hasta el momento, se utilizará un conjunto de siete imágenes. Los resultados presentados a continuación avalan el correcto funcionamiento del algoritmo.

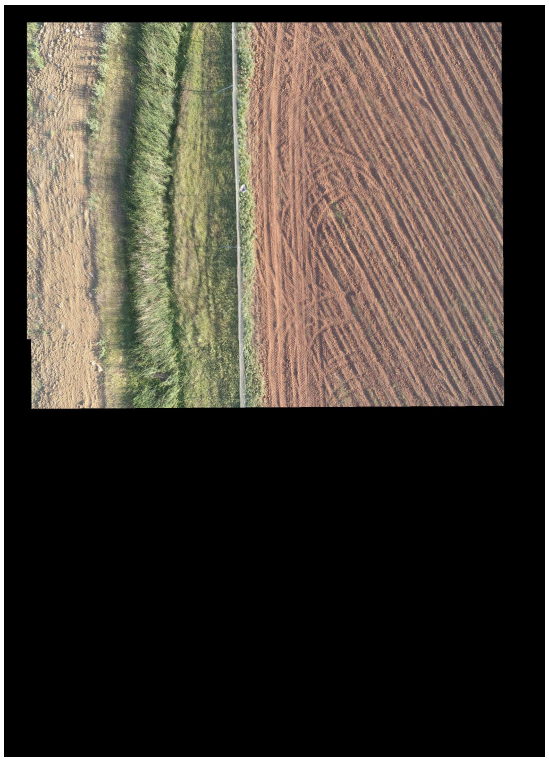

Figura 6.4 Resultado del algoritmo para las imágenes I_1 e I_2.

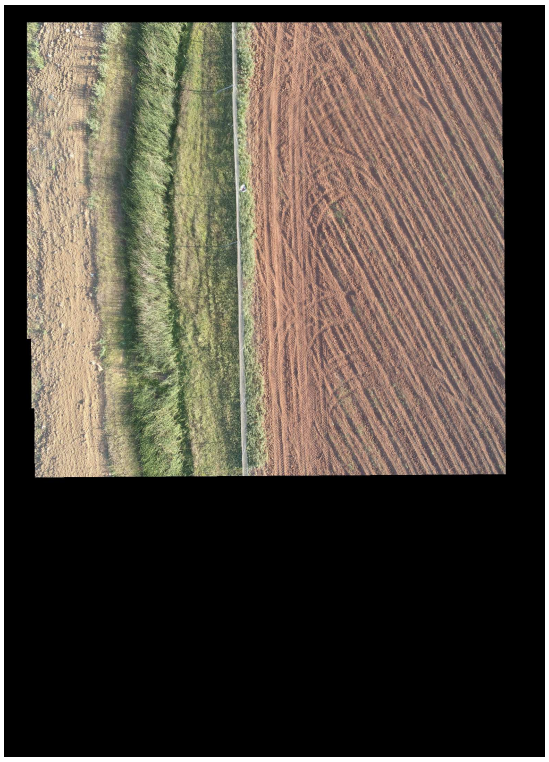

Figura 6.5 Resultado del algoritmo para las imágenes I_1, I_2 e I_3.

Figura 6.6 Resultado del algoritmo para las imágenes I_1, I_2, I_3 e I_4.

Figura 6.7 Resultado del algoritmo para las imágenes I_1, I_2, I_3, I_4 e I_5.

Figura 6.8 Resultado del algoritmo para las imágenes I_1, I_2, I_3, I_4, I_5 e I_6.

Figura 6.9 Resultado del algoritmo para las imágenes I_1, I_2, I_3, I_4, I_5, I_6 e I_7.

6.7 Conclusiones

El cálculo de los parámetros que provean el mejor ajuste de un conjunto de datos a un modelo de las características del tratado en este capítulo es un problema matemático cerrado, es decir, con una solución definida. El reto abordado no ha sido, por tanto, cómo solucionar la minimización del error, sino qué simplificaciones del problema escoger para lograr disminuir considerablemente el tiempo de resolución. Para ello, ha sido necesario conocer de forma previa las condiciones de vuelo que se darán durante las misiones de inspección, permitiendo la reducción de los seis grados de libertad propios de un cuerpo en un espacio tridimensional a tres grados de libertad (Figura 6.10).

Por otra parte, la resolución del problema de mínimos cuadrados mediante el uso de la descomposición en valores simples es una forma de calcular la transformación que requiere poco tiempo de computación, además de dotar al algoritmo de robustez y eficiencia. Pese a que, matemáticamente, puedan existir formas menos arduas de calcular los autovalores y los autovectores de una matriz, estos algoritmos requieren mayor tiempo de implementación y de ejecución.

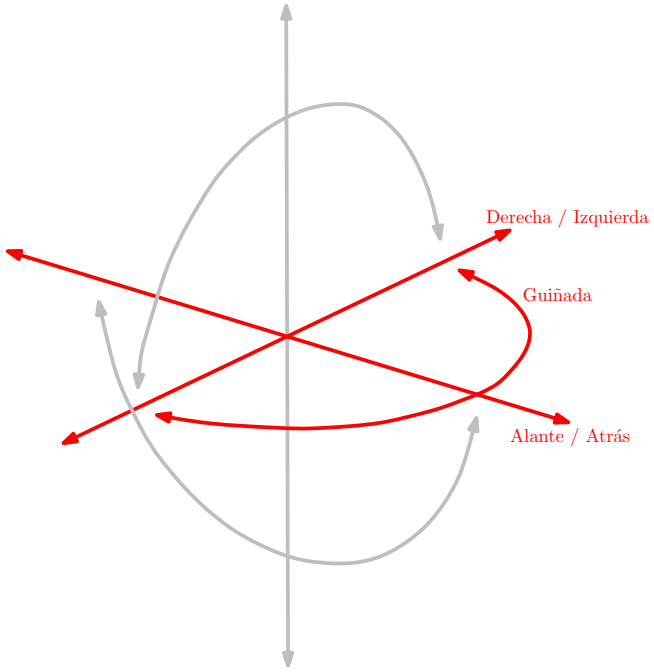

Figura 6.10 Grados de libertad escogidos.

7 Presentación y Análisis de Resultados

7.1 Introducción

En este capítulo se describirá todo el proceso relativo a la presentación de los resultados, tanto del mosaico generado, como de toda la información complementaria descrita en el Apartado 1.1. Por otra parte, se realizará un breve análisis del proceso de obtención del mosaico, que se centrará principalmente en los tiempos de ejecución requeridos, y de la calidad de los resultados mediante la comparativa de la imagen obtenida con la generada por el software *Pix4Dmapper*.

Con el objetivo de validar el método diseñado, se cuenta con conjuntos de imágenes adquiridos por un robot aéreo DJI Phantom 4 Pro [1] en distintas tareas de inspección. Se trata de un robot de 1,388 kg de peso y 350 mm de longitud (de hélice a hélice), equipado con sistemas de posicionamiento por satélite GPS y GLONASS. Posee una cámara CMOS de 20 megapíxeles capaz de ofrecer imágenes de tamaño 3648x5472 píxeles en proporción 16:9.

El conjunto de imágenes utilizado en este capítulo corresponde a una de las trayectorias descritas por el robot. El vuelo se realizó a una altura de 20 metros y con una velocidad de desplazamiento de 4 m/s. La cámara realizó una captura cada 2 metros, de forma que dos imágenes consecutivas poseen un porcentaje de puntos en común de entre el 60 y el 70 %. La duración aproximada de la inspección fue de 5 minutos.

Figura 7.1 DJI Phantom 4 Pro. Fuente: [1].

Figura 7.2 Vista general de la zona inspeccionada.

7.2 Mosaico

El mosaico generado mediante el algoritmo descrito en este documento a partir del conjunto de imágenes capturadas por el robot aéreo se muestra en la Figura 7.8-izquierda. Se trata de un mosaico compuesto por la unión de treinta imágenes.

Para su composición se ha utilizado un *edge threshold* de 50 píxeles, se han extraído 100 *features* por imagen, se ha utilizado una ventana de búsqueda para la asociación de 150 píxeles y se han seleccionado los 10 mejores *matches* de cada correlación. Además, se han escalado las imágenes a un tamaño de 912x1368 píxeles (un 25 % de su tamaño original) para reducir el tiempo requerido por la extracción de puntos de interés.

7.3 Georeferencias

Uno de los objetivos descritos en el Apartado 1.1 es el de aportar georeferencias a las imágenes que componen el mosaico. Esta información será ofrecida por el método desarrollado como una imagen diseñada para superponer sobre el mosaico. En esta imagen, se situarán marcas con las correspondientes coordenadas GPS. El resultado de suponer las georeferencias al mosaico puede verse en la Figura 7.8-centro.

Figura 7.3 Imagen de referencias GPS superpuesta sobre un extracto del mosaico.

7.3.1 Cálculo de las coordenadas GPS

Las coordenadas para georeferenciar las imágenes son obtenidas a partir de la proyección de las coordenadas GPS del UAV (Figura 7.3). En caso de utilizar un altímetro, el primer paso es corregir su medida de forma que se tengan en cuenta las variaciones en *pitch* y *roll*:

$$\hat{H} = H\cos\varphi\cos\theta \tag{7.1}$$

En segundo lugar, se calculan las variaciones en x e y producidas por la orientación de la cámara:

$$\begin{aligned} \Delta x &= \hat{H}\tan\theta = H\cos\varphi\sin\theta \\ \Delta y &= \hat{H}\tan\varphi = H\sin\varphi\cos\theta \end{aligned} \tag{7.2}$$

De esta forma, la proyección del punto $(x_0, y_0, \hat{H})$ vendrá dada por la expresión $(x_0 + |\Delta x|, y_0 + |\Delta y|)$.

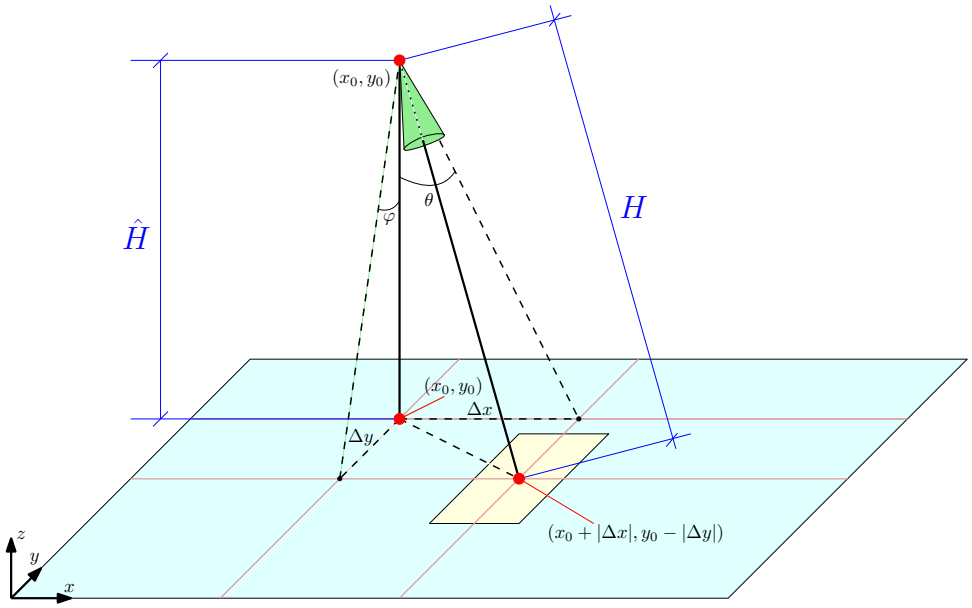

Figura 7.4 Proyección de coordenadas GPS.

7.4 Niveles de Concentración

Otro de los objetivos del algoritmo es el de incorporar al mosaico información relativa a la concentración en aire de un cierto gas. Para facilitar al operario que realice la inspección la detección de una posible fuga, los niveles de concentración obtenidos por el sensor situado en el robot serán transformados en marcas cuyo color dependa del valor medido. Al igual que con las referencias GPS, esta información será ofrecida por el algoritmo desarrollado como una imagen diseñada para superponer sobre el mosaico. El resultado de suponer los indicadores de concentración al mosaico puede verse en la Figura 7.8-derecha.

Figura 7.5 Imagen de indicadores de concentración de gas superpuesta sobre un extracto del mosaico.

7.4.1 Cálculo del color de los indicadores

El color de los indicadores será calculado, en función de los niveles de concentración medidos, según la Tabla 7.1.

Tabla 7.1 Cálculo del color a partir de los niveles de concentración.

Concentración	R	G	B
Desde x_{min} hasta $x_{min} + \frac{1}{4}(x_{max} - x_{min})$	0	$0 \rightarrow 255$	255
Desde $x_{min} + \frac{1}{4}(x_{max} - x_{min})$ hasta $x_{min} + \frac{1}{2}(x_{max} - x_{min})$	0	255	$255 \rightarrow 0$
Desde $x_{min} + \frac{1}{2}(x_{max} - x_{min})$ hasta $x_{min} + \frac{3}{4}(x_{max} - x_{min})$	$0 \rightarrow 255$	255	0
Desde $x_{min} + \frac{3}{4}(x_{max} - x_{min})$ hasta x_{max}	255	$255 \rightarrow 0$	0

A partir de la tabla anterior, pueden definirse las siguientes funciones a trozos:

$$I_R(x) = \begin{cases} 0 & si \quad x_{min} \leq x \leq x_{min} + \frac{1}{2}(x_{max} - x_{min}) \\ 255 \cdot \frac{4(x-x_{min})-2(x_{max}-x_{min})}{x_{max}-x_{min}} & si \quad x_{min} + \frac{1}{2}(x_{max} - x_{min}) < x \leq x_{min} + \frac{3}{4}(x_{max} - x_{min}) \\ 255 & si \quad x_{min} + \frac{3}{4}(x_{max} - x_{min}) \leq x \leq x_{max} \end{cases} \tag{7.3}$$

$$I_G(x) = \begin{cases} 255 \cdot \frac{4(x-x_{min})}{x_{max}-x_{min}} & si \quad x_{min} \leq x \leq x_{min} + \frac{1}{4}(x_{max} - x_{min}) \\ 255 & si \quad x_{min} + \frac{1}{4}(x_{max} - x_{min}) < x \leq x_{min} + \frac{3}{4}(x_{max} - x_{min}) \\ 255 + 255 \cdot \frac{3(x_{max}-x_{min})-4(x-x_{min})}{x_{max}-x_{min}} & si \quad x_{min} + \frac{3}{4}(x_{max} - x_{min}) \leq x \leq x_{max} \end{cases} \tag{7.4}$$

$$I_B(x) = \begin{cases} 255 & si \quad x_{min} \leq x \leq x_{min} + \frac{1}{4}(x_{max} - x_{min}) \\ 255 + 255 \cdot \frac{(x_{max}-x_{min})-4(x-x_{min})}{x_{max}-x_{min}} & si \quad x_{min} + \frac{1}{4}(x_{max} - x_{min}) < x \leq x_{min} + \frac{1}{2}(x_{max} - x_{min}) \\ 0 & si \quad x_{min} + \frac{1}{2}(x_{max} - x_{min}) \leq x \leq x_{max} \end{cases} \tag{7.5}$$

Cuya representación puede verse en las Figuras 7.6 y 7.7.

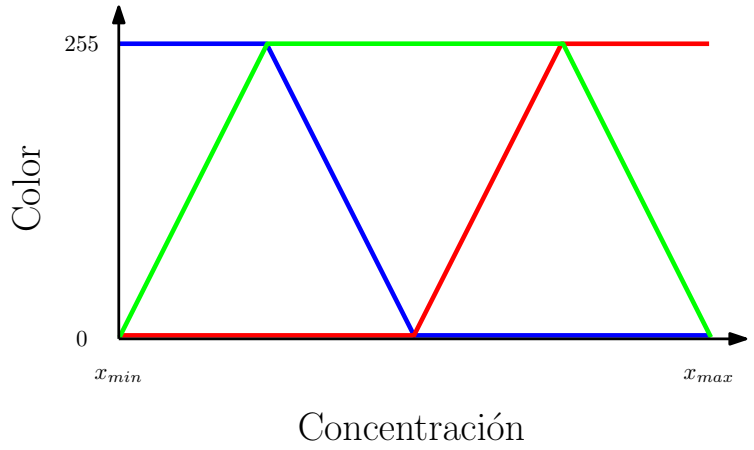

Figura 7.6 Representación del color como función del nivel de concentración (I).

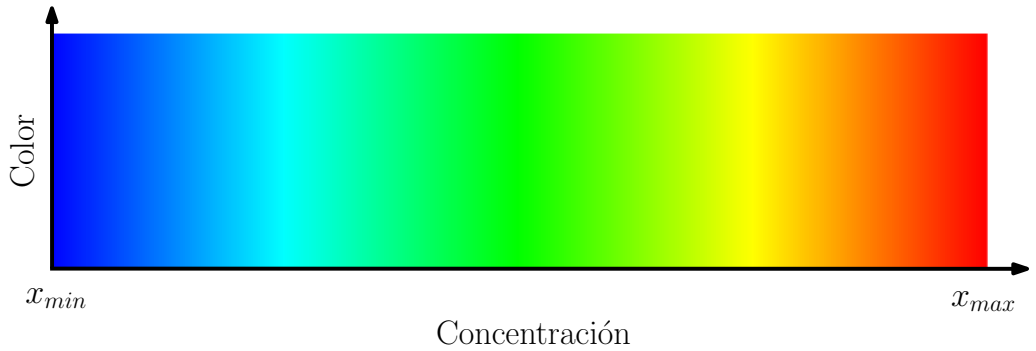

Figura 7.7 Representación del color como función del nivel de concentración (II).

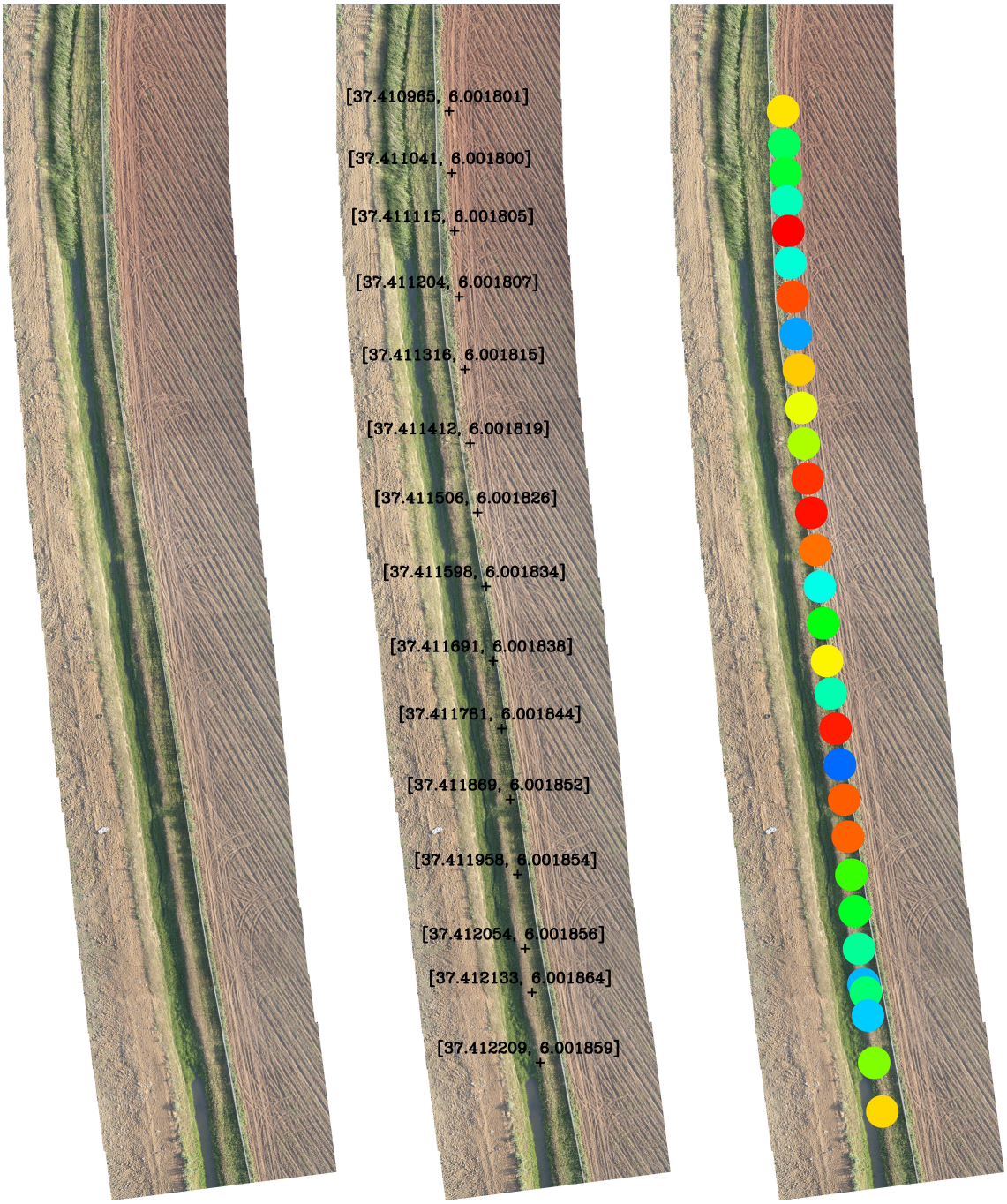

Figura 7.8 Mosaico generado integrado treinta imágenes.

7.5 Trayectoria descrita

A partir de los parámetros utilizados por el algoritmo para transformar las imágenes, puede ser estimada la posición y orientación del robot aéreo durante la inspección. En las Figuras 7.9 y 7.10 se presenta la trayectoria descrita durante el vuelo.

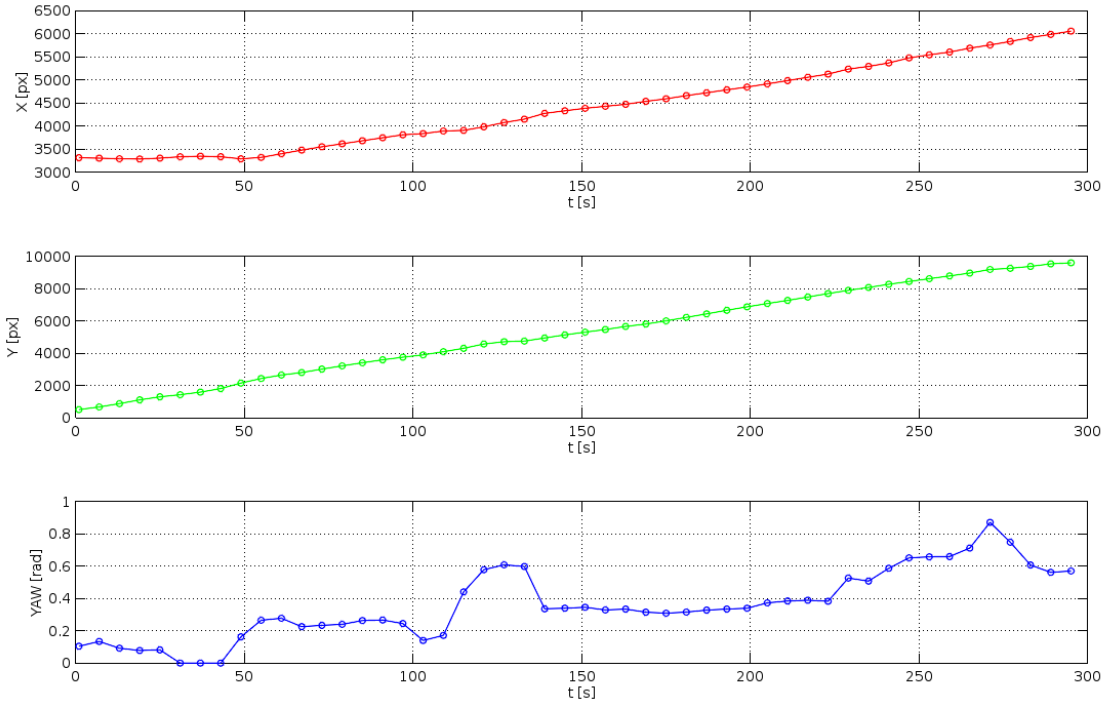

Figura 7.9 Representación de la trayectoria descrita durante la inspección (I).

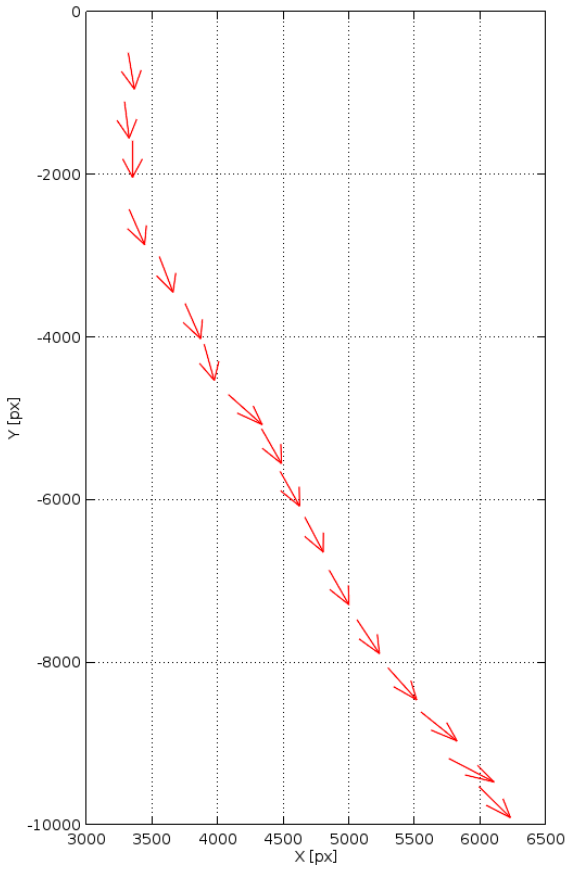

Figura 7.10 Representación de la trayectoria descrita durante la inspección (II).

7.6 Análisis de Resultados

7.6.1 Calidad del mosaico generado

El uso de interpolación en la generación del mosaico es, indirectamente, una forma de poner de manifiesto la calidad del mismo. En la Figura 7.11 puede apreciarse como la interpolación entre dos imágenes provoca resultados similares a la superposición de una sobre otra. Este hecho puede ser interpretado como un indicador de la buena calidad del resultado ofrecido por el algoritmo.

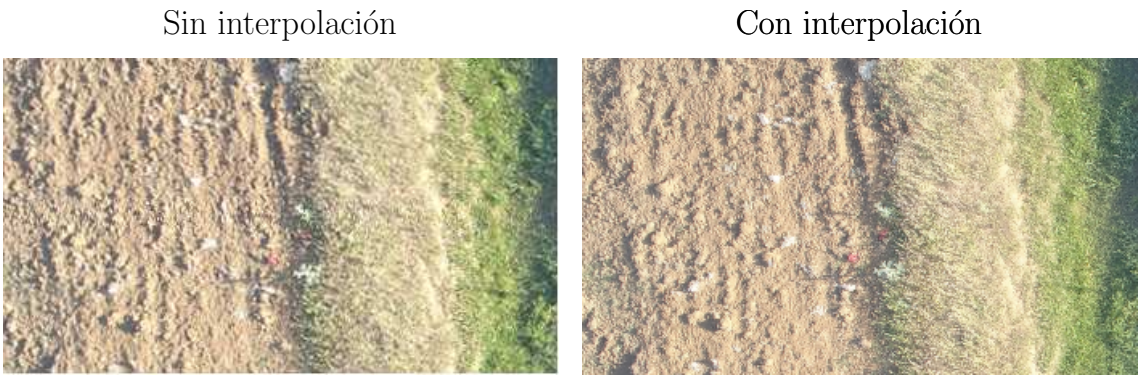

Figura 7.11 Detalles de mosaicos con y sin interpolación aplicada.

7.6.2 Distribución temporal

El método diseñado requiere aproximadamente 0.25 segundos por unión. De ese tiempo, 0.2 segundos se dedican a la extracción de puntos singulares (0.1 segundos por imagen), 0.001 segundos a la asociación de los puntos extraídos y 0.05 segundos al cálculo y aplicación de la transformación. Todos estos datos son aproximados y se corresponden con pruebas realizadas utilizando un procesador AMD Ryzen 5 2600 de 6 núcleos y 1.47 GHz y dos memorias RAM Kingston Predator KHX2933C15D4/8GX de 4GB.

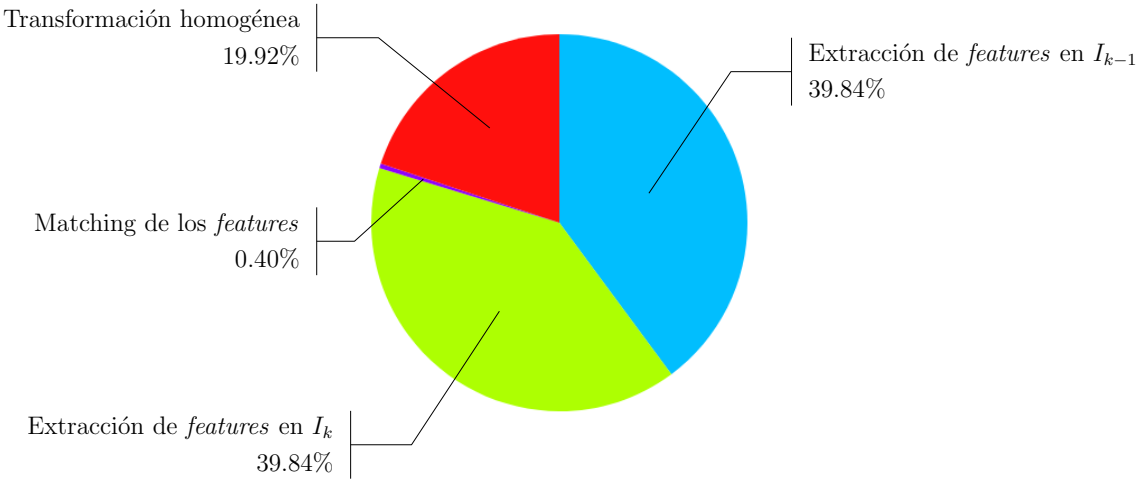

Figura 7.12 Distribución temporal aproximada de las etapas del algoritmo.

Tabla 7.2 Calidad y distribución temporal aproximada en función del número de *features* extraídos.

Número de *features*	Extracción (s)	Correlación (s)	Transformación (s)	Total (s)	Calidad
50	0.19	0.0003	0.049	0.2393	Baja
100	0.20	0.0010	0.050	0.2510	Alta
200	0.20	0.0050	0.050	0.2550	Alta
1000	0.22	0.0080	0.051	0.2790	Alta

Tabla 7.3 Calidad y distribución temporal aproximada en función del número de *matches* seleccionados.

Número de *matches*	Extracción (s)	Correlación (s)	Transformación (s)	Total (s)	Calidad
5	0.20	0.0010	0.050	0.2510	Alta
10	0.20	0.0010	0.050	0.2510	Alta
20	0.20	0.0010	0.050	0.2510	Media
50	0.20	0.0010	0.051	0.2520	Baja

7.7 Comparación de Resultados

Con el fin de comprobar la calidad del mosaico generado, se comparará con el resultado ofrecido por el software profesional de fotogrametría y mapeado *Pix4Dmapper* desarrollado por *Pix4D* [2].

Figura 7.13 Logotipo de *Pix4D*. Fuente: [2].

En la Figura 7.14 puede comprobarse como ambos resultados presentan una calidad semejante. La diferencia fundamental entre ambos métodos reside en el tiempo de computación requerido. Mientras que *Pix4Dmapper* tardó aproximadamente 2 minutos en generar el mosaico, el método propuesto por este documento tardó aproximadamente 37 segundos, es decir, un 30.83 % del tiempo empleado por el software de *Pix4D*.

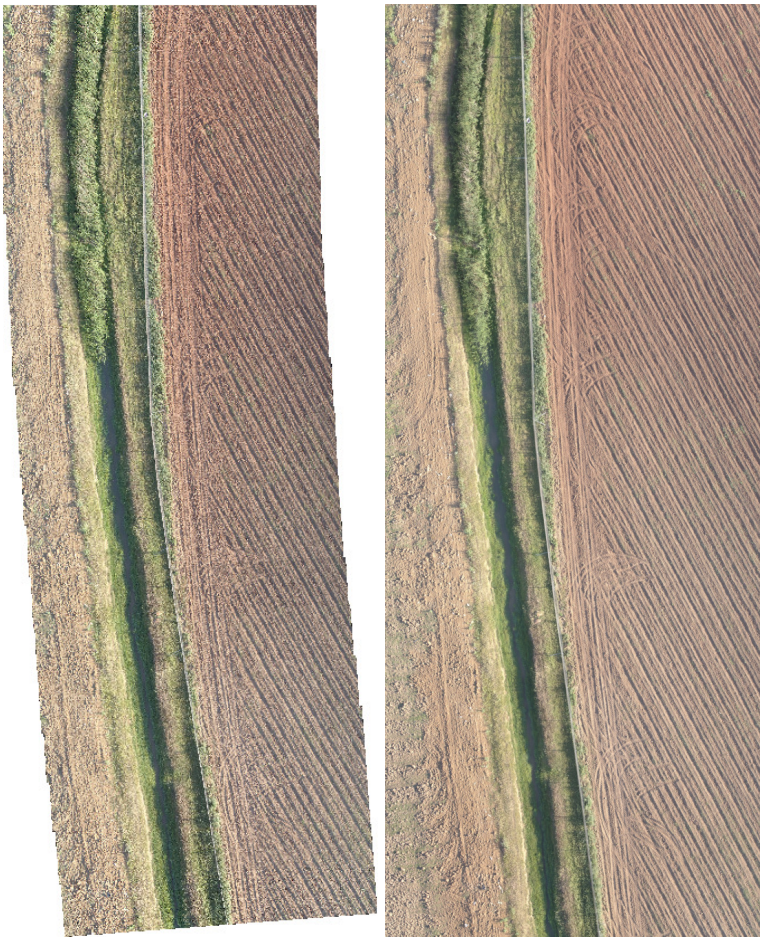

Figura 7.14 Comparativa con mosaico generado por *Pix4Dmapper*.

7.8 Conclusiones

En la literatura asociada a las técnicas de generación de mosaicos existen métodos con diversas características, sin embargo, la gran mayoría posee un carácter general. El algoritmo diseñado utiliza las hipótesis presentadas en el Aparado 1.2.1 para realizar una particularización del problema y, así, ofrecer resultados en un tiempo relativamente bajo en comparación con muchas otras técnicas de *mosaicking*. Además, esta particularización dota al algoritmo de gran eficiencia y robustez.

Al ser el objetivo final del método generar mosaicos para la inspección y mantenimiento de gasoductos, incorporar georeferencias a las imágenes facilita el trabajo de localización de un posible fallo en la infraestructura, sin que ello requiera un excesivo coste computacional. Del mismo modo, los indicadores de concentración de gas son una forma rápida e intuitiva de detectar fugas cuya incorporación al mosaico tampoco requiere un coste temporal muy elevado.

Como puede comprobarse, el fin último del método diseñado es facilitar al operario la tarea de inspección y mantenimiento y generar resultados en el menor tiempo posible. Ambos objetivos repercuten directamente en una disminución del coste económico de la operación.

8 Conclusiones y Desarrollo Futuro

8.1 Introducción

En este capítulo se presentan las conclusiones extraídas durante el desarrollo de este proyecto y se describen posibles aspectos de investigación futura. Se detallarán las características del algoritmo inferidas a partir del análisis de los resultados realizado, sus posibles extensiones a otros tipos de misiones de inspección y los medios de implementación existentes. Por otra parte, se propondrán la modificación del algoritmo para la inspección de estructuras con regiones a distintas alturas y la generación automática de información relativa a calidad del mosaico como vías de desarrollo futuro.

8.2 Comportamiento del Método

Como se comentó en el capítulo anterior, una de las principales ventajas del método propuesto es el uso de simplificaciones derivadas del conocimiento de las condiciones de vuelo para reducir el tiempo de computación lo máximo posible sin sacrificar calidad en los resultados, eficiencia o robustez. Esta forma de proceder se debe a que el objetivo final de este proyecto es aplicar el algoritmo a la inspección de infraestructuras lineales, por lo que ha predominado durante su diseño la especialización frente al carácter general que presentan la mayoría de técnicas.

Sin embargo, la expansión del método a otro tipo de condiciones es posible si se asumen ciertas consecuencias. El algoritmo es capaz de operar sin medidas de GPS, lo que disminuiría su robustez y lo haría más propenso a fallos. También es posible incorporar más grados de libertad al planteamiento del problema, incrementando el número de parámetros de las matrices de transformación que relacionan las imágenes, lo que supondría un aumento bastante considerable del tiempo requerido.

8.3 Implementación del Algoritmo

El método diseñado realiza una lectura de las imágenes reunidas en un soporte de almacenamiento (HDD, SSD, memoria USB, tarjeta de memoria, ...). La generación del mosaico se realiza de forma *offline* una vez que el robot aéreo haya finalizado la inspección y se hayan trasladado las imágenes al dispositivo que realizará el procesamiento.

Aunque esta sea la forma de proceder que ha sido utilizada, el algoritmo también puede trabajar de forma paralela a la adquisición de las imágenes, ya que posee un coste temporal cercano al tiempo real . Para ello, existen dos alternativas: implementar el algoritmo en un dispositivo a bordo del robot que permita su ejecución o realizar un tratamiento *online* de las imágenes, es decir, enviar las adquisiciones desde el robot hasta el dispositivo que realizará el procesamiento haciendo uso de tecnología que permita la interconexión inalámbrica, como por ejemplo una red WiFi. No plantea un problema el hecho de que el tiempo entre capturas sea menor que el tiempo requerido para unir las imágenes, ya que esto únicamente supondría un retraso o *delay* de la generación del mosaico respecto al vuelo del robot.

8.4 Desarrollo Futuro

Dos posibles vías de desarrollo futuro para este proyecto son la relajación de la hipótesis de *tierra plana* o el diseño de métodos automáticos para la validación de resultados.

8.4.1 Relajación de la hipótesis de *tierra plana*

La diferencia de altura de objetos en la infraestructura a inspeccionar puede provocar fallos en la generación del mosaico (Figura 8.1), por este motivo, una de las hipótesis empleadas ha sido considerar que todas las regiones de las imágenes se encuentran al mismo nivel. Para ello, el vuelo del robot se ha realizado a una distancia lo suficientemente elevada como para poder despreciar las diferencias de altura de los objetos (Figura 8.2).

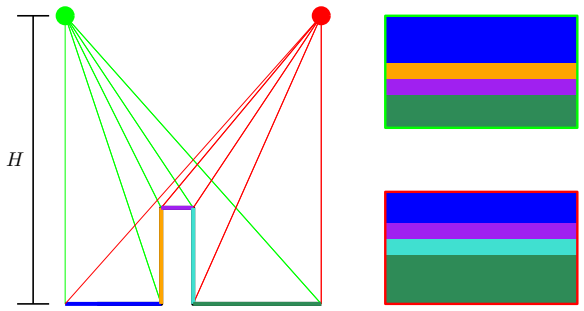

Figura 8.1 Problema asociado a diferencias de altura.

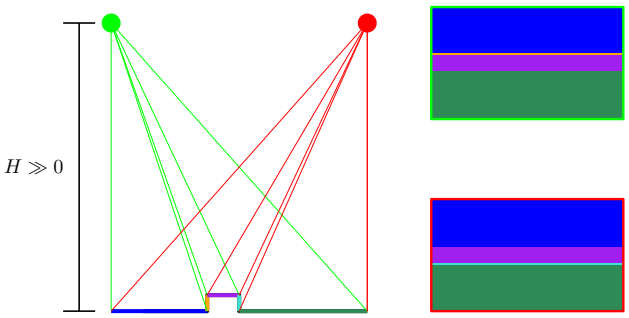

Figura 8.2 Solución al problema asociado a diferencias de altura mediante hipótesis de *tierra plana*.

Una posible extensión del método sería implementar el cálculo de una estimación de la profundidad de los objetos capturados mediante técnicas de visión estéreo (Figura 8.3). De esta forma, podrían inspeccionarse infraestructuras con regiones a alturas claramente diferenciadas o realizar vuelos a distancias no tan elevadas como las requeridas actualmente. Para generar el mosaico, sería necesario agrupar los *features* por altura, realizándose el ajuste de las imágenes a distintos niveles de profundidad.

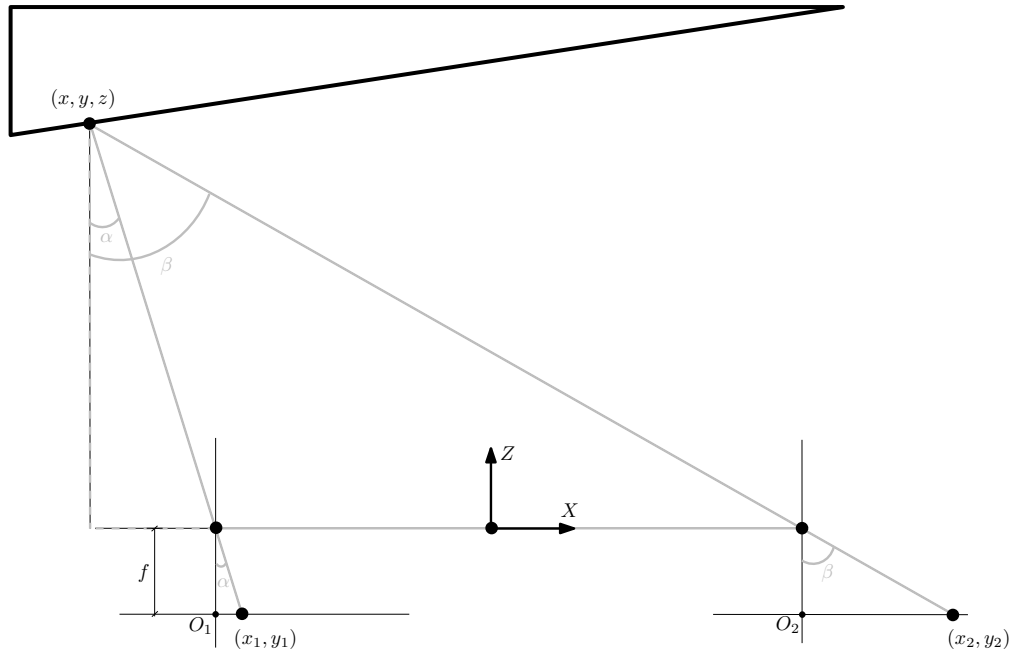

Figura 8.3 Cálculo de profundidad con técnicas de visión estéreo para cámaras con ejes ópticos paralelos.

8.4.2 Técnicas para validación de los resultados

Otra posible mejora puede ser la implementación de un método que permita validar el algoritmo a partir de los resultados obtenidos. Este método debería ofrecer, de forma automatizada, información relativa a calidad del mosaico y otros datos de interés que permitan una comparación tanto cualitativa como cuantitativa. El medio para entregar al usuario todos estos datos podría ser un informe generado junto al mosaico.

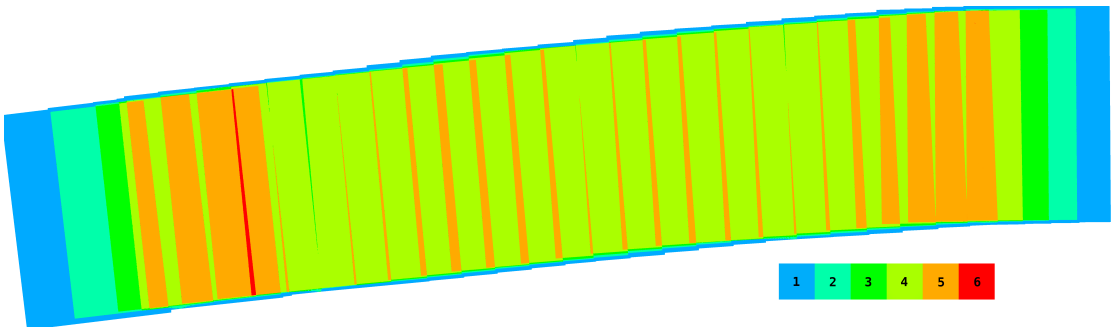

Figura 8.4 Ejemplo de información complementaria al mosaico. Número de imágenes superpuestas.

9 Anexo A: Sistema de Referencia Cartesiano a partir de GPS

El método diseñado se caracteriza, a diferencia de otros muchos presentes en la literatura asociada a la generación de mosaicos, por una particularización que le confiere eficiencia, robustez y disminución del tiempo de ejecución. Este comportamiento es adquirido gracias al uso del GPS con el fin de obtener una estimación del desplazamiento entre una captura y otra. Ejemplos del uso de esta estimación son el cálculo de las zonas comunes a dos imágenes consecutivas o el uso de ventanas de búsqueda durante la fase de correlación.

En este anexo se presentan los sistemas de geolocalización empleados por el algoritmo y se detalla la construcción, a partir de las medidas del GPS, de un sistema de referencia cartesiano que permita localizar al robot aéreo. Durante el desarrollo de este documento se ha hecho alusión en varias ocasiones a este sistema para referir la posición de la cámara al punto de origen de la trayectoria.

9.1 Sistemas de Geolocalización

Un sistema de geolocalización permite obtener la ubicación geográfica. Se trata de un caso particular de sistema de posicionamiento usado para objetos situados en la Tierra. Entre los sistemas de referencia de geolocalización más empleados se encuentran WGS, ECEF y LTP [55].

9.1.1 WGS

WGS (*World Geodetic System*) es un sistema de coordenadas elipsoidales (latitud, longitud y altitud) que toma como origen el centro de masa terrestre. Actualmente se utiliza el estándar WGS 84, que posee un error máximo de 2 cm y emplea un modelo terrestre descrito en la Tabla 9.1.

Tabla 9.1 Modelo terrestre utilizado por WGS 48.

Semieje mayor	6378137 m
Semieje menor	6356752.314 m
Velocidad angular	$7.292 \cdot 10^{-5}$ rad/s

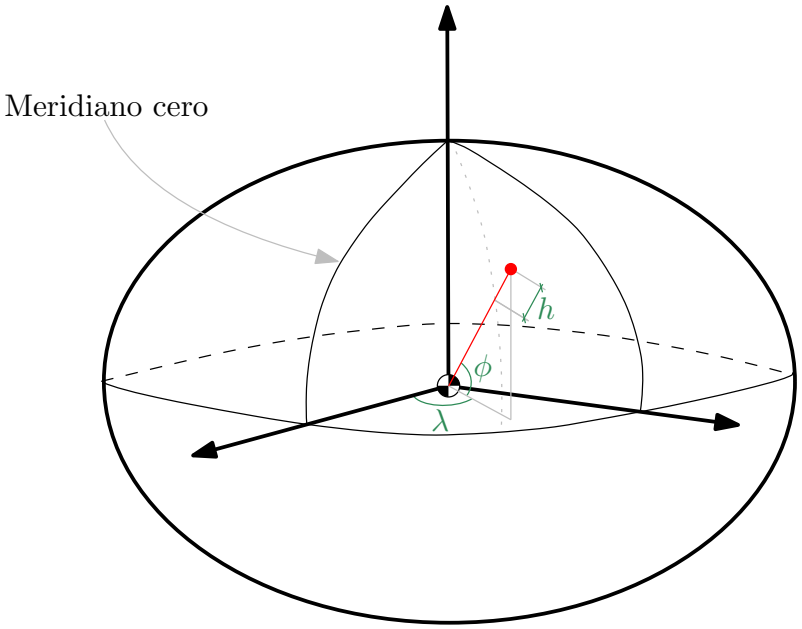

Figura 9.1 Sistema WGS.

9.1.2 ECEF

ECEF (*Earth-Centered, Earth-Fixed*) es un sistema cartesiano que toma como origen el centro de masa de la Tierra. La distancia de un punto al origen se conoce como distancia geodésica. La situación de los ejes del sistema se muestra en la Figura 9.2.

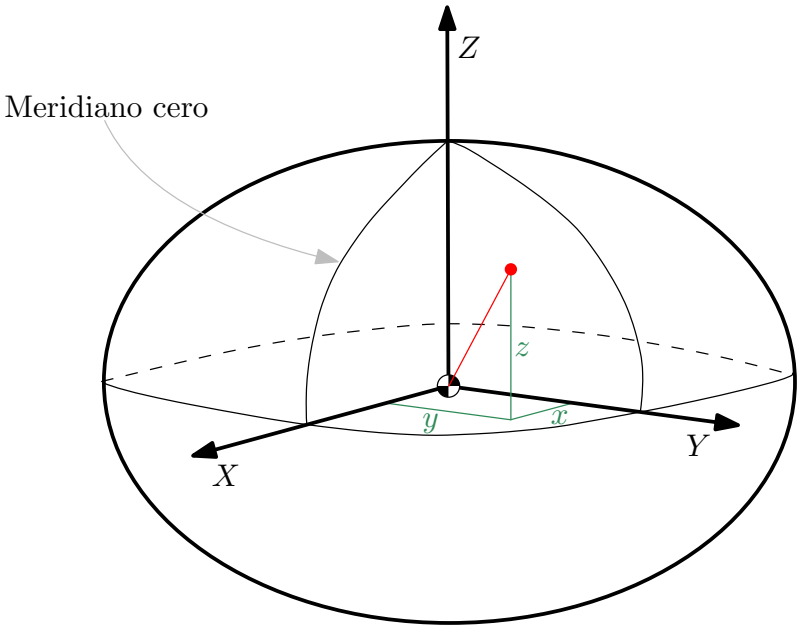

Figura 9.2 Sistema ECEF.

9.1.3 LTP

LTP (*Local Tangent Plane*), también llamado LVLH (*Local Vertical, Local Horizontal*), es un sistema cartesiano cuyos ejes se soportan sobre un plano tangente a la superficie terrestre. Dos de los ejes pertenecen a dicho plano, mientras que el tercero tiene la dirección perpendicular que apunta al centro de la Tierra. Existen dos variantes de este sistema (Figura 9.3): ENU (*East North Up*) y NED (*North East Down*), cuya diferencia reside en la posición de los ejes en el plano.

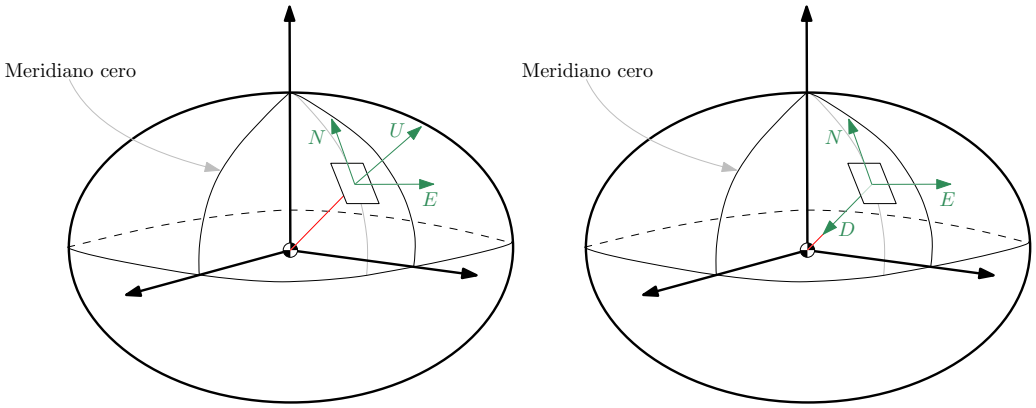

Figura 9.3 Sistemas LTP.

9.2 Sistemas de Medida de Posición para Geolocalización

Existen diversas formas de obtener la geolocalización de un objeto. Los sistemas más empleados son los de geolocalización por satélite [8]:

- GPS (*Global Positioning System*), originalmente conocido como Navstar GPS, es una red de satélites desarrollada por el Departamento de Defensa de los Estados Unidos de América.
- GLONASS (*GLObal NAvigation Satellite System*) es una red de satélites desarrollada por la Unión Soviética.
- Galileo es una red de satélites desarrollada por la Unión Europea y la Agencia Espacial Europea.

Otros sistemas no basados en redes de satélites son GSM, que emplea las torres de telefonía, y WiFi, que utiliza las direcciones IP de los routers cercanos.

9.3 Establecimiento de Sistema de Referencia Cartesiano

Este apartado tiene como objetivo detallar la construcción de un sistema de referencia cartesiano a partir de las medidas proporcionadas por el GPS [13] [18] [42].

9.3.1 Conversión de coordenadas WGS a ECEF

Los dispositivos GPS ofrecen medidas WGS. El primer paso para constituir el sistema buscado es convertir estas medidas en coordenadas ECEF. Sean ϕ la latitud, λ la longitud y h la altitud de un punto, sus coordenadas ECEF se pueden calcular como:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} (R+h)\cos\phi\cos\lambda \\ (R+h)\cos\phi\sin\lambda \\ (\frac{b^2}{a^2}R+h)\sin\phi \end{bmatrix} \tag{9.1}$$

Donde a es el semieje mayor del modelo elipsoidal de la Tierra, b el semieje menor y R el radio de curvatura principal calculada mediante la expresión:

$$R = \frac{a}{\sqrt{1 - e^2 \sin^2 \phi}} \tag{9.2}$$

Siendo e la primera excentricidad del elipsoide:

$$e = \sqrt{1 - \frac{b^2}{a^2}} \tag{9.3}$$

9.3.2 Conversión de coordenadas ECEF a ENU

A partir de las coordenadas ECEF, con origen en el centro de masa de la Tierra, se puede construir un sistema ENU con origen en un plano tangente a la superficie terrestre [56]. Sean ϕ la latitud, λ la longitud y h la altitud de un punto:

$$\begin{bmatrix} E \\ N \\ U \end{bmatrix} = \begin{bmatrix} -\sin\lambda & \cos\lambda & 0 \\ -\cos\lambda\sin\phi & -\sin\lambda\sin\phi & \cos\phi \\ \cos\lambda\cos\phi & \sin\lambda\cos\phi & \sin\phi \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{9.4}$$

Por tanto, un punto $P(\phi,\lambda,h)$ perteneciente a la trayectoria del robot puede ser descrito en coordenadas cartesianas referidas al punto inicial del vuelo como $P(x,y,z)$, siendo $x = E$, $y = N$ y $z = U$.

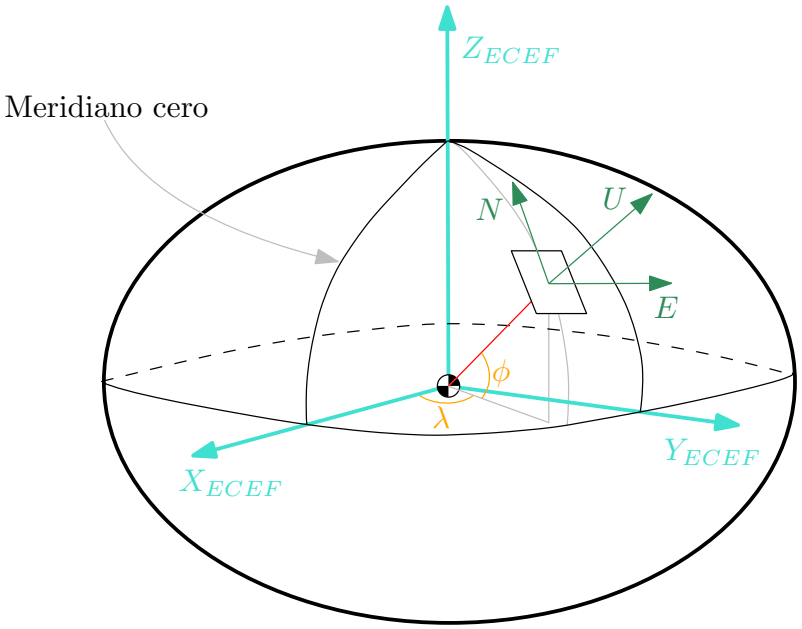

Figura 9.4 Construcción de sistema ENU a partir de WGS.

Índice de Figuras

Índice de Tablas

Índice de Códigos

Bibliografía

[1] *DJI*, "https://www.dji.com/", Junio 2019.

[2] *Pix4D*, "https://www.pix4d.com", Junio 2019.

[3] *GRVC*, "https://grvc.us.es/national-projects/", Mayo 2019.

[4] *OpenCV*, "https://opencv.org/", Mayo 2019.

[5] *ROS*, "https://www.ros.org", Mayo 2019.

[6] Rintu Abraham and Philomina Simon, *Review on mosaicing techniques in image processing*, 04 2013, pp. 63–68.

[7] Mohammadreza Aghaei, S Leva, and F Grimaccia, *PV power plant inspection by image mosaicing techniques for IR real-time images*, 06 2017, pp. 1–6.

[8] Reha Alkan, Himmet Karaman, and M Sahin, *GPS, GALILEO and GLONASS satellite navigation systems and GPS modernization*, 07 2005, pp. 390–394.

[9] Pietro Azzari, L Di Stefano, and Alessandro Bevilacqua, *An effective real-time mosaicing algorithm apt to detect motion through background subtraction using a PTZ camera*, vol. 2005, 10 2005, pp. 511–516.

[10] H Bay, A Ess, T Tuytelaars, and Luc Van Gool, *SURF: Speeded up robust features*, Computer Vision and Image Understanding **110** (2008), 346–359.

[11] Udhav Bhosle, Subhasis Chaudhuri, and Sumantra Dutta Roy, *A fast method for image mosaicing using geometric hashing*, IETE Journal of Research **48** (2015).

[12] Erkan Bostanci, Nadia Kanwal, Betul Bostanci, and Mehmet Guzel, *A fuzzy brute force matching method for binary image features*, (2017).

[13] Guowei Cai, Ben Chen, and Tong Heng Lee, *Unmanned rotorcraft systems*, 01 2011.

[14] Jun Chen, Quan Xu, Linbo Luo, Yongtao Wang, and Shuchun Wang, *A robust method for automatic panoramic UAV image mosaic*, Sensors **19** (2019), 1898.

[15] Pankaj Daga, François Chadebecq, Dzhoshkun Shakir, Luis Garcia-Peraza-Herrera, Marcel Tella, George Dwyer, Anna David, Jan Deprest, Danail Stoyanov, Tom Vercauteren, and Sebastien Ourselin, *Real-time mosaicing of fetoscopic videos using SIFT*, **9786** (2016).

[16] L.S. Davis, *A survey of edge detection techniques*, Computer Graphics and Image Processing **4** (1975), 248–260.

[17] Nilanjan Dey, Pradipti Nandi, Nilanjana Barman, Debolina Das, and Subhabrata Chakraborty, *A comparative study between Moravec and Harris Corner Detection of noisy images using adaptive wavelet thresholding technique*, Int. J. Engineering Research and Applications **2** (2012).

[18] Samuel Drake, *Converting GPS coordinates [phi, lambda, h] to navigation coordinates (ENU)*, DSTO **DSTO-TN** (2002).

[19] C G. Harris and M J. Stephens, *A combined corner and edge detector*, Proceedings 4th Alvey Vision Conference **1988** (1988), 147–151.

[20] Junhong Gao, Seon Joo Kim, and Michael S. Brown, *Constructing image panoramas using dual-homography warping*, 07 2011, pp. 49–56.

[21] Debabrata Ghosh and Naima Kaabouch, *A survey on image mosaicing techniques*, Journal of Visual Communication and Image Representation **34** (2015).

[22] Jinlong Hu, Xianrong Peng, and Chengyu Fu, *A comparison of feature description algorithms*, Optik-International Journal for Light and Electron Optics **126** (2014).

[23] R.B. Inampudi, *Image mosaicing*, 08 1998, pp. 2363–2365 vol.5.

[24] Şahin Işık, *A comparative evaluation of well-known feature detectors and descriptors*, International Journal of Applied Mathematics, Electronics and Computers **3** (2014).

[25] Amila Jakubovic and Jasmin Velagic, *Image feature matching and object detection using brute-force matchers*, 09 2018, pp. 83–86.

[26] Ebrahim Karami, Siva Prasad, and Mohamed Shehata, *Image matching using SIFT, SURF, BRIEF and ORB: Performance comparison for distorted images*, 11 2015.

[27] Taygun Kekec, Alper Yildirim, and Mustafa Unel, *A new approach to real-time mosaicing of aerial images*, Robotics and Autonomous Systems **62** (2014).

[28] Kourosh Khoshelham, *Closed-form solutions for estimating a rigid motion from plane correspondences extracted from point clouds*, ISPRS Journal of Photogrammetry and Remote Sensing **114** (2016), 78–91.

[29] David Kriegman, *Homography estimation*, Computer Vision I **CSE 252A** (2007).

[30] Jing Li, Tao Yang, Jingyi Yu, Zhaoyang Lu, Ping Lu, Xia Jia, and Wenjie Chen, *Fast aerial video stitching*, International Journal of Advanced Robotic Systems **11** (2014).

[31] Ming Li, D Li, and D Fan, *A study on automatic UAV image mosaic method for paroxysmal disaster*, ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences **39** (2012), 123–128.

[32] W Li, S Zhao, W Zhang, X Liu, and G Yu, *A mosaic method for UAV images based on filtering*, Wuhan Daxue Xuebao (Xinxi Kexue Ban)/Geomatics and Information Science of Wuhan University **43** (2018), 943–950.

[33] Yan Li, Carly J. Randall, Robert van Woesik, and Eraldo Ribeiro, *Underwater video mosaicing using topology and superpixel-based pairwise stitching*, Expert Systems with Applications **119** (2019), 171–183.

[34] Yuping Lin and Gérard Medioni, *Map-enhanced UAV image sequence registration and synchronization of multiple image sequences*, 06 2007.

[35] Q Liu, W Liu, Lei Zou, J Wang, and Y Liu, *A new approach to fast mosaic UAV images*, ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences **XXXVIII-1/C22** (2012), 271–276.

[36] David Lowe, *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision **60** (2004), 91–110.

[37] Anlong Ming and Huadong Ma, *A blob detector in color images*, 07 2007, pp. 364–370.

[38] Lansu Nie, Yan Tang, and Qin Xu, *Survey of image mosaics technologies*, vol. 218, pp. 699–707, 01 2013.

[39] M P. Mukhina, Yu V. Trach, and A P. Prymak, *Comparison of BRIEF and ORB binary descriptors*, Electronics and Control Systems **4** (2018).

[40] S.Y. Pattar, *Study of corner detection algorithms and evaluation methods*, International Journal of Innovative Research in Science, Engineering and Technology **4** (2015), 2780–2787.

[41] Shmuel Peleg, Benny Rousso, Alex Rav-Acha, and Assaf Zomet, *Mosaicing on adaptive manifolds*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **22** (2000), 1144–1154.

[42] M Pierozzi, *On the transformation from WGS84 to a local geodetic system*, **48** (1989), 45–55.

[43] Kshitija Pol, Diksha Chaudhary, and Parul Bansal, *Biomedical image mosaicing: A review*, 11 2016, pp. 155–157.

[44] Ramnarayan R, Nikita Saklani, and Vasundhara Verma, *A review on edge detection technique "canny edge detection"*, International Journal of Computer Applications **178** (2019), 28–30.

[45] Elena Ranguelova, *A salient region detector for structured images*, 11 2016, pp. 1–8.

[46] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski, *ORB: an efficient alternative to SIFT or SURF*, 11 2011, pp. 2564–2571.

[47] J.B Ryu, C.G Lee, and H.H Park, *Formula for Harris Corner Detector*, Electronics Letters **47** (2011), 180–181.

[48] Tomokazu Sato, Sei Ikeda, Masayuki Kanbara, Akihiko Iketani, Noboru Nakajima, and Naokazu Yokoya, *High-resolution video mosaicing for documents and photos by estimating camera motion*, Proc SPIE **5299** (2004).

[49] Javier Sánchez, Nelson Monzón, and Agustín Salgado, *An analysis and implementation of the Harris Corner Detector*, Image Processing On Line **8** (2018), 305–328.

[50] Shaharyar Ahmed Khan Tareen and Zahra Saleem, *A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK*, 03 2018.

[51] Marcel Tella, Pankaj Daga, Francois Chadebecq, Stephen Thompson, Dzhoshkun Shakir, George Dwyer, Ruwan Wimalasundera, Jan Deprest, Danail Stoyanov, Tom Vercauteren, and Sebastien Ourselin, *A combined EM and visual tracking probabilistic model for robust mosaicking: Application to fetoscopy*, 07 2016.

[52] Hoan Trinh, Christian Daul, Walter Blondel, and Dominique Lamarque, *Mosaicing of images with few textures and strong illumination changes: Application to gastroscopic scenes*, 10 2018, pp. 1263–1267.

[53] Dushyant Vaghela and Kapildev Naina, *A review of image mosaicing techniques*, International Journal of Advance Research in Computer Science and Management Studies **2** (2014).

[54] Harriet W. Mulder, Josien P.W. Pluim, Ben Ren, Alexander Haak, Max A. Viergever, Johan G Bosch, and Marijn van Stralen, *Atlas-based mosaicing of 3D transesophageal echocardiography images of the left atrium*, 10 2015, pp. 1–4.

[55] Yansen Wang, Giap Huynh, and Chatt Williamson, *Integration of Google Maps/Earth with microscale meteorology models and data visualization*, Computers and Geosciences **61** (2013), 23–31.

[56] John Wieczorek, Qinghua Guo, and Robert Hijmans, *The point-radius method for georeferencing locality descriptions and calculating associated uncertainty*, International Journal of Geographical Information Science **18** (2004), 745–767.

[57] Jun Wu, Zhongkui Dong, Zhigang Liu, and Guoqing Zhou, *Geo-registration and mosaic of UAV video for quick-response to forest fire disaster*, Proceedings of SPIE - The International Society for Optical Engineering **6788** (2007).

[58] Li Xiuhua, Song Liming, Wang Shuo, Hou Alin, and Wang Xin, *Survey of CCD image mosaic techniques*, 2010 International Conference on Future Information Technology and Management Engineering, FITME 2010 **3** (2010).

[59] Alessio Xompero, Oswald Lanz, and Andrea Cavallaro, *MORB: A multi-scale binary descriptor*, 10 2018, pp. 2167–2171.

[60] Ezzeddine Zagrouba, Walid Barhoumi, and S Amri, *An efficient image-mosaicing method based on multifeature matching*, Machine Vision and Applications **20** (2009), 139–162.

Índice alfabético

Glosario

BRIEF Binary Robust Independent Elementary Features. 14, 29, 38

CDTI Centro para el Desarrollo Tecnológico Industrial. 4
CIEN Consorcios de Investigación Empresarial Nacional. 4
CMOS Complementary Metal–Oxide-Semiconductor. 69
CV Computer Vision. 4

ECEF Earth-Centered, Earth-Fixed. 83–86
ENU East North Up. 85, 86

FAST Features from Accelerated Segment Test. 14, 29, 30

GLONASS GLObal NAvigation Satellite System. 69, 85
GPS Global Positioning System. 25, 33, 39, 48, 50, 54, 69–72, 83, 85
GRVC Grupo de Robótica, Visión y Control. 4
GSM Global System for Mobile communications. 85

HCD Harris Corner Detector. 12
HDD Hard Disk Drive. 79

INSPECTOR INdustrial inSPECtion and mainTenance of complex OR unattended facilities. 4

LTP Local Tangent Plane. 83, 85
LVLH Local Vertical, Local Horizontal. 85

NED North East Down. 85

oFAST Oriented Features from Accelerated Segment Test. 14, 30
ORB Oriented FAST and Rotated BRIEF. 13, 14, 24, 25, 29–31, 34–36, 38, 45

rBRIEF Rotated Binary Robust Independent Elementary Features. 14, 38, 45
ROI Region Of Interest. 34
ROS Robot Operative System. 4

SIFT Scale Invariant Feature Transform. 13, 14, 24

SSD Solid-State Drive. 79

SURF Speed Up Robust Features. 13, 14, 24

SVD Singular Value Decomposition. 62

UAS Unmanned Aerial System. 1

UAV Unmanned Aerial Vehicle. 24, 33, 35, 39, 45, 48, 49, 71

USB Universal Serial Bus. 79

WGS World Geodetic System. 83, 85

WiFi Wireless Fidelity. 79, 85