Proyecto Fin de Carrera
- Ingeniería Informática -
Febrero de 2012

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Aplicación de técnicas de alineamiento de imágenes para la georreferenciación automática de imágenes aéreas y satelitales

Autor: Pablo Fuertes Correa
Director: Pedro R. Muro Medrano

Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza

Grupo de Sistemas de
Información Avanzados (IAAA)
Centro Politécnico Superior
María de Luna, 1
50018 Zaragoza (España)

Índice

1. Introducción

1.1. Resumen

En el presente proyecto fin de carrera se investigan las técnicas de alineación de imágenes con la finalidad de establecer la correspondencia de una imagen capturada con una base de imágenes que sirvan como referencia. Esta base de imágenes será la aportada por el Plan Nacional de Ortofoto Aérea (PNOA) que ofrece una fotografía continua de toda España perfectamente georreferenciada.

Con tal fin se hace una implementación de una técnica basada en características llegando a la conclusión de que esta no ofrece unos resultados aceptables en el ámbito de aplicación.

Teniendo en cuenta estos resultados se han buscado alternativas que pudieran resolver esta problemática. Para ello se ha ampliado el abanico de técnicas posibles puesto que se ha considerado que el ámbito de aplicación no requiere los resultados en tiempo real o cerca del mismo. Esto ha hecho que también sean contempladas técnicas basadas en la intensidad.

Así mismo se han realizado pruebas con diferentes lenguajes para determinar cual resulta más adecuado en cuanto a velocidad de prototipado y a eficiencia. De estas pruebas se ha decidido realizar la implementación en C++ junto a OpenCV. No obstante el prototipo final ha sido realizado en Matlab, por motivos que serán detallados más adelante.

Una vez escogido el algoritmo [1] que más se ajusta a las restricciones del problema y que mejores resultados ofrece, se ha procedido a su implementación. Se ha observado que era demasiado complejo como primera aproximación y por tanto se ha buscado un algoritmo anterior [2], de los mismos autores. Este artículo es la base en el que se apoya el que se pretende implementar, este hecho hace que compartan la tecnología subyacente pero con menor número de optimizaciones y por tanto resulta más fácil su implementación e interpretación.

Se ha implementado dicho algoritmo, obteniendo resultados no satisfactorios. Esto ha hecho necesario realizar una revisión en profundidad del funcionamiento del método y de los aspectos técnicos de su implementación. Siendo también necesario rehacer el análisis matemático planteado por los autores para buscar alguna errata en su publicación u omisión de algún paso que provocase el funcionamiento incorrecto. Con el fin de validar que la implementación realizada fuera correcta se ha realizado una implementación alternativa en Matlab con idéntico resultado. Desde este momento se ha continuado trabajando con el prototipo de Matlab y se han explorado múltiples opciones para intentar solucionar el problema, delegando en funciones de matlab, transformando el problema en el inverso, realizando el cálculo numérico de las derivadas y otras muchas variantes hasta llegar a una solución que aportase un resultado aceptable.

1.2. Contexto profesional

El presente proyecto fin de carrera ha sido desarrollado en el Grupo de Sistemas de Información Avanzados (IAAA) del Departamento de Informática e Ingeniería de Sistemas (DIIS) perteneciente a la Universidad de Zaragoza.

La actividad de investigación del grupo de Sistemas de Información Avanzados (IAAA) está enfocada en las tecnologías de software para la creación

de sistemas de información con datos georreferenciados (también denominados generalmente datos geográficos y ahora más comúnmente datos espaciales) y, especialmente, en la configuración de un nuevo paradigma que se ha convenido en llamar Infraestructuras de Datos Espaciales (IDEs). Se trata de cubrir desde aspectos básicos de ingeniería de servicios hasta aspectos metodológicos de la puesta en funcionamiento de sistemas industriales que explotan la información geográfica. Concretamente, la labor de investigación del grupo aborda problemáticas vinculadas a las ontologías y meta-datos; interoperabilidad, composición y encadenamiento de servicios; visualización inteligente de información geográfica, meta-datos, procesamiento semántico y recuperación inteligente de información (indexación, interrogación y recuperación); métodos y procesos para la creación de información y el modelado de contenidos heterogéneos; y, finalmente, un aspecto que se considera fundamental para la realimentación técnica y que es la definición, creación y puesta en funcionamiento real de nuevos servicios y aplicaciones.

1.3. Contexto tecnológico

El desarrollo de este proyecto se realiza en el ámbito del conjunto de problemas englobados dentro del término anglosajón *image registration* que en adelante denominaremos alineamiento geométrico, puesto que en el fondo se trata de encontrar una relación entre las distintas imágenes.

De forma más precisa, tal como se describe en el estudio de Brown [3]:

> El alineamiento geométrico es el proceso de transformación de diferentes conjuntos de datos a un sistema de coordenadas común. Los datos pueden ser múltiples fotografías, datos de diferentes sensores, de diferentes épocas o de diferentes puntos de vista. Se utiliza en visión artificial, imagen médica, reconocimiento automático de objetivo y en la recopilación y análisis de imágenes y datos de los satélites. El registro es necesario para poder comparar o integrar los datos obtenidos de estas diferentes mediciones.

Clasificación general de los algoritmos:

- Basados en la intensidad versus basados en características

  Los métodos basados en la intensidad comparan los patrones de intensidad en las imágenes a través de métricas de correlación, mientras que los métodos basados en las características encuentran la correspondencia entre las características de las imágenes, tales como puntos, líneas y contornos.

  Los métodos basados en la intensidad registran imágenes completas o sub-imágenes. Si las sub-imágenes están alineadas, los centros de las correspondientes sub-imágenes se consideran como puntos de características correspondientes. Mientras que los métodos basados en características establecen la correspondencia entre varios puntos en las imágenes. Luego, conociendo la correspondencia entre varios puntos en las imágenes, se determina una transformación para mapear la imagen objetivo a las imágenes de referencia, estableciendo punto por punto, la correspondencia entre las imágenes de referencia y la objetivo.

- Modelos de transformación

  La primera categoría de modelos de transformación incluye las transformaciones lineales, que son la traslación, rotación, escalamiento y otras transformaciones afines. Estas son de naturaleza global, por lo tanto, no pueden modelar diferencias geométricas locales entre las imágenes.

  La segunda categoría de transformaciones permiten que estas sean *elásticas* o *no rígidas*. Estas transformaciones son capaces de deformar localmente la imagen objetivo para alinearla con la imagen de referencia. Las transformaciones no rígidas incluyen las funciones de base radial, modelos físicos continuos y los modelos de grandes deformaciones.

- Métodos de dominio espacial versus frecuencia

  Los métodos espaciales operan en el dominio de la imagen, haciendo coincidir los patrones de intensidad o las características de las imágenes. Algunos de los algoritmos que emparejan características son derivaciones de las técnicas tradicionales para realizar el alineamiento manual de la imagen, en el que un operador decide los correspondientes puntos de control en las imágenes. Cuando el número de puntos de control supera el mínimo requerido para definir el modelo de transformación apropiado, los algoritmos iterativos como RANSAC (RANdom SAmple Consensus) se pueden utilizar para estimar de manera robusta los parámetros de un tipo particular de transformación (por ejemplo, afín) para el alineamiento de las imágenes.

  Los métodos en el dominio de la frecuencia encuentran los parámetros de transformación para el registro de las imágenes mientras trabajan en el dominio de transformación. Estos métodos trabajan por simple transformación, tales como la traslación, la rotación y el escalado. La aplicación del método de correlación de fase a un par de imágenes produce una tercera imagen que contiene un solo pico. La ubicación de este pico corresponde a la traslación relativa entre las imágenes. A diferencia de muchos algoritmos en el dominio espacial, el método de correlación de fase es insensible al ruido, las oclusiones y otros defectos típicos de las imágenes médicas o por satélite. Además, la correlación de fase utiliza la transformada rápida de Fourier (FFT) para calcular la correlación cruzada entre las dos imágenes, lo que por lo general resulta en grandes mejoras en el rendimiento. Este método puede ser extendido para determinar la rotación y las diferencias de escala entre dos imágenes convirtiendo, en primer lugar, las imágenes a coordenadas log-polar. Debido a las propiedades de la Transformada de Fourier, los parámetros de rotación y de escalamiento se pueden determinar de forma invariable a la traslación.

- Métodos monomodal versus multimodal

  Los métodos monomodales tienden a registrar las imágenes en la misma modalidad adquirida por el único tipo de escáner/sensor, mientras que los métodos de multimodales tienden a registrar las imágenes obtenidas por diferentes tipos de escáner/sensor.

  Los métodos de registro multimodal se utilizan a menudo en imagen médica ya que las imágenes de un paciente se obtienen con frecuencia a partir de escáneres diferentes. Los ejemplos incluyen el registro de imágenes del

cerebro por tomografía axial computarizada/resonancia magnética o de todo el cuerpo por PET/TAC para la localización de tumores, el registro de imágenes TAC con y sin realce de contraste para la segmentación de imágenes de partes específicas de la anatomía y el registro de ultrasonido y de imágenes TAC para la localización de la próstata en radioterapia.

- Métodos automáticos versus interactivos

  Se han desarrollado métodos manuales, interactivos, semi-automáticos y automáticos. Los métodos manuales proporcionan herramientas para alinear las imágenes de forma manual. Los métodos interactivos reducen el sesgo del usuario mediante la realización de ciertas operaciones claves de forma automática mientras que todavía confían en que el usuario guíe el registro. Los métodos semiautomáticos realizan más pasos de registro de forma automática pero dependen del usuario para verificar la corrección de un registro. Los métodos automáticos no permiten la interacción del usuario y realizan todos los pasos del registro de forma automática.

- Medidas de similitud para el registro de la imagen

  Una medida de similitud de imágenes cuantifica el grado de similitud entre los patrones de intensidad de dos imágenes. La elección de una medida de similitud de imágenes depende de la modalidad de las imágenes a ser alineadas. Los ejemplos más comunes de las medidas de similitud de imágenes incluyen la correlación cruzada, la información mutua, la suma de los cuadrados de las diferencias de intensidad y la razón de uniformidad de la imagen. La información mutua y la información mutua normalizada son las medidas de similitud de imagen más populares para el registro de imágenes multimodales. La correlación cruzada, la suma de los cuadrados de las diferencias de intensidad y la razón de uniformidad de la imagen se utilizan para el registro de imágenes en la misma modalidad.

2. Trabajo realizado

2.1. Primera aproximación (basada en características)

En esta aproximación al problema se implementó un método basado en *features SURF*[4] para encontrar los descriptores de ambas imágenes y emparejarlos según su similitud. Luego mediante un procedimiento estadístico de *RANSAC* se descartan los emparejamientos espurios al mismo tiempo que se calcula la homografía con el conjunto resultante de *inliers*. Una vez que tenemos la transformación proyectiva entre ambas imágenes podemos alinearlas o georreferenciarlas.

Para realizar este prototipo se ha utilizado como lenguaje C++ y la librería de visión por computador OpenCV. Se ha hecho uso de su funcionalidad para poder extraer los descriptores, emparejarlos y encontrar la transformación realizando el proceso de RANSAC de manera implícita en la función que encuentra la homografía. Como referencia para determinar el algoritmo y que funciones utilizar se ha trabajado con los libros *Learning OpenCV: computer vision with the OpenCV library* [5], *OpenCV 2 Computer Vision Application Programming*

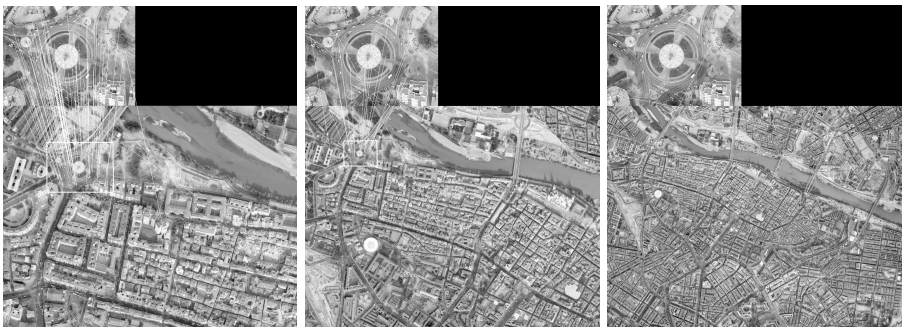


Figura 1: Diferencia de escala

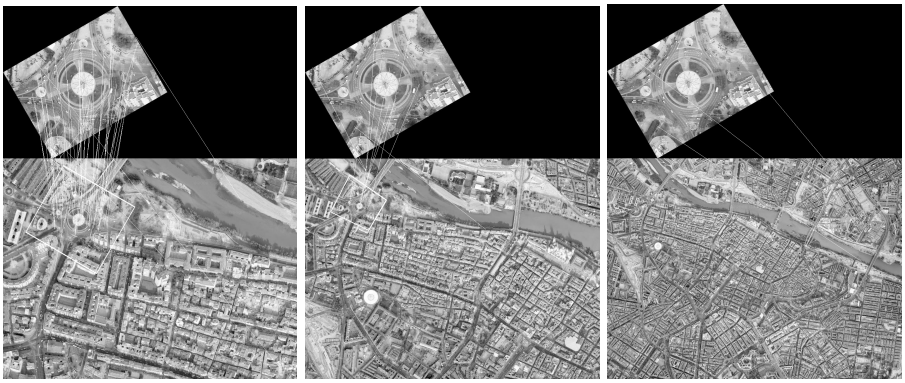


Figura 2: Diferencia de escala y rotación

Cookbook [6], *Computer Vision: Algorithms and Applications* [7], y el artículo *Image alignment and stitching: A tutorial* [8]

Una vez se tiene el prototipo implementado se realizan una serie de pruebas con distintas imágenes para evaluar su comportamiento y en el caso de resultar satisfactorio refinar utilizando métodos más sofisticados y eficientes.

En la figura 1 se aprecia como varían los resultados conforme se hace un cambio de escala. Si ambas imágenes tienen un factor de escala similar se consiguen muchos emparejamientos correctos y el *RANSAC* es capaz de desestimar los incorrectos de manera adecuada. Conforme aumenta la diferencia de escala se van obteniendo menos emparejamientos hasta que llega un momento en el que ya no hay emparejamientos con los que trabajar.

En la figura 2 se aprecia el mismo proceso de variación en escala pero con el añadido de una rotación de $-30°$, el efecto de esta rotación es que la degradación de los emparejamientos se produce antes.

En la figura 3 se aprecia el proceso de alineamiento según aumenta la diferencia en perspectiva de ambas imágenes a una escala similar. En general se obtienen buenos resultados incluso con variaciones bastante grandes. En este caso se aprecia mejor que se trata de una transformación proyectiva entre ambas imágenes, provocando deformaciones no lineales que son más complicadas de emparejar. En las imágenes donde ya no se puede encontrar una transformación satisfactoria se aprecia que el *RANSAC* es un método de montecarlo y por lo

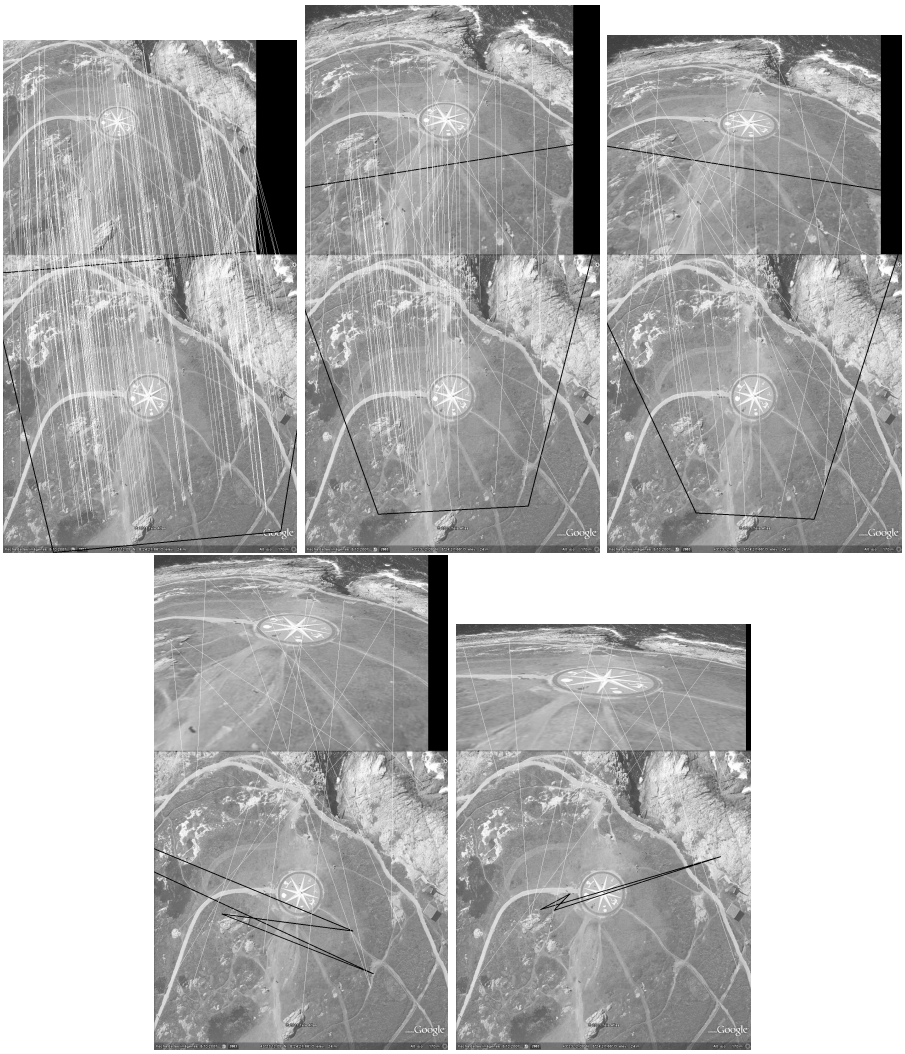


Figura 3: Diferencia de perspectiva

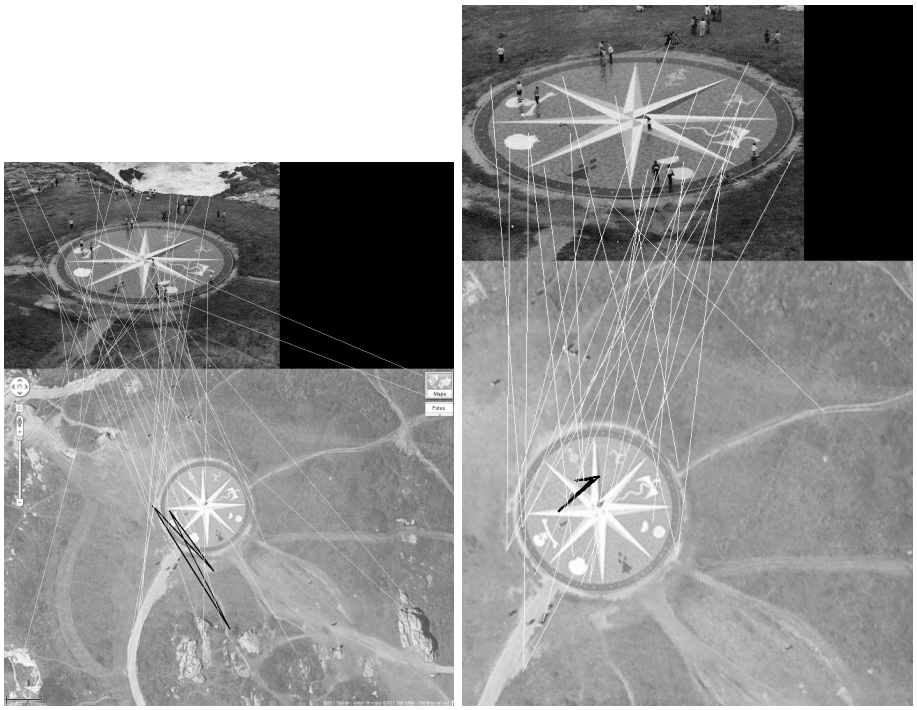


Figura 4: Casos reales

tanto nos devuelve siempre un resultado, sea este correcto o no, entonces nos corresponde a nosotros discriminar verosimilitud del mismo.

En las pruebas anteriores se han usado imágenes obtenidas de la misma fuente. Por lo tanto, aunque sean diferentes tienen otras muchas cosas en común como por ejemplo la resolución con la que han sido captadas, niveles de ruido similares pero sobre todo iluminación y textura que son los parámetros que más afectan a los descriptores empleados y en general a cualquier método. Por este motivo se han realizado una serie de pruebas con imágenes reales, mostradas en la figura 4. En esta figura se puede observar como en los casos extremos donde hay mucha variación de perspectiva además de imágenes de fuentes muy distintas no se obtiene un resultado positivo.

El resultado negativo en el caso anterior era esperable puesto que se ha planteado un caso muy complicado al introducir una gran variación en el punto de vista con el que están tomadas las imágenes y que ambas proceden de fuentes e instantes distintos. Por ello, en la figura 5 se ha considerado un caso trivial pero con imágenes obtenidas de distintas fuentes. Lo sorprendente de este caso es que el método no ha sido capaz de darnos un resultado positivo como se puede apreciar en la figura 6

Este resultado inesperado ha hecho que analicemos este caso particular con más detalle en la figura 7, para poder encontrar el origen del problema y poder presentar distintas líneas de actuación que permitan solventarlo.

Lo que se aprecia en esta figura 7. El mismo recorte de la misma imagen obtenida de fuentes distintas a la misma escala, es que la resolución es demasiado

(a) PNOA (b) Google Maps

Figura 5: PNOA y GMAPS

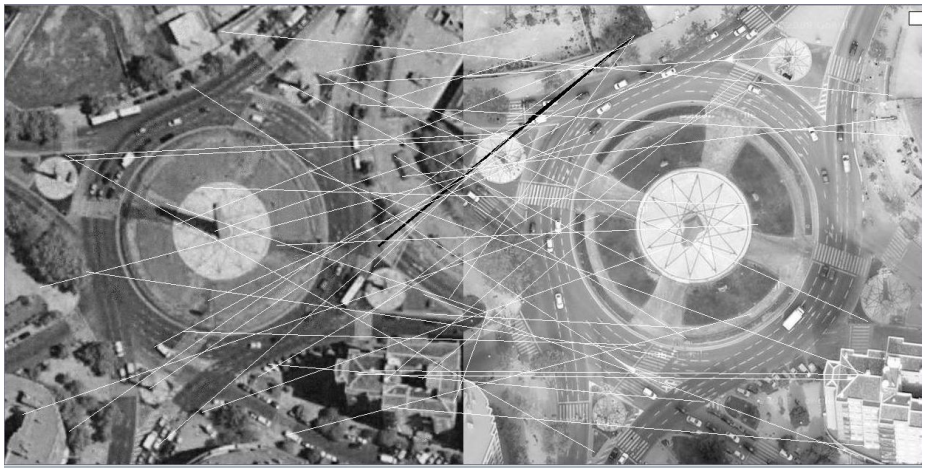


Figura 6: PNOA contra GMAPS

Figura 7: Análisis detallado

distinta y en una tenemos muchos detalles mientras que en la otra apenas se aprecian y además se observan artefactos en la imagen. Esta gran diferencia ha hecho que el resultado no fuese adecuado con estas dos imágenes que a simple vista nos parecían muy similares como para que el método fallase.

Esta problemática nos ha llevado a consultar con un experto en la materia para poder discriminar si se trataba de un problema de nuestro método o bien se trata de un problema intrínseco al caso planteado, ya que en el ámbito que en el que se quiere aplicar este método, estas son el tipo de imágenes a las que nos vamos a enfrentar.

Nuestro método es demasiado *ingenuo* ya que se trata de un prototipo cuyo único objetivo en esta fase es evaluar la eficacia de manera preliminar. Esto hace que el experto nos recomiende una serie de mejoras y mientras estas son implementadas él realizará una serie de pruebas a las imágenes con sus propios métodos, más avanzados, para comprobar hasta donde puede llegar el estado de la técnica.

En la figura 8 se pueden observar el resultado obtenido por el método del experto, que según sus comentarios ha sido difícil de conseguir porque había pocos emparejamientos buenos y ha sido necesario ajustar los parámetros de ajuste del descriptor *SIFT*. Esto nos lleva a la recomendación de realizar un emparejamiento guiado y después realizar una estimación robusta en una segunda iteración, para terminar volviendo a realizar una estimación de la transformación precisa con el método anterior.

El inconveniente de esta aproximación es que requiere la intervención del usuario en una etapa preliminar y el objetivo buscado es que todo sea lo más automático y general posible, no siendo dependientes del ámbito concreto o siendo necesaria la interacción con el usuario. Aunque si sería aceptable la intervención del usuario en la parte final del proceso donde se le podrían presentar varios resultados y que fuese él quien eligiera cual de ellos es más verosímil.

No obstante, el resultado obtenido aunque satisfactorio se ha conseguido con muy pocos *inliers* lo que hace sospechar que en un caso general no se va a tener tanta suerte, con el inconveniente de que este método sólo nos ofrece un

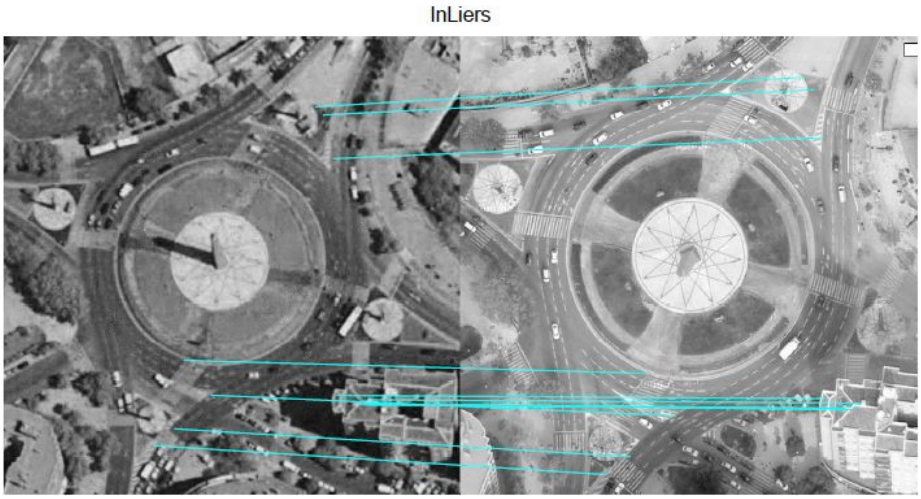


Figura 8: Análisis externo

resultado pero no nos garantiza que este sea positivo. Esto y la dependencia de tener que ajustar los parámetros en caso concreto y la necesidad de interacción con el usuario hace que nos planteemos la necesidad de utilizar otros métodos y para ello veremos en detalle cuales se han seleccionado.

2.2. Método seleccionado (basado en la intensidad)

El objetivo ha sido encontrar alternativas al método empleado que está basado en descriptores, aunque sea el tipo de métodos más empleado en la actualidad se ha comprobado como en el ámbito de aplicación donde va a ser empleado no da unos resultados del todo satisfactorios. Además los descriptores *SURF*, y similares, suelen estar orientados a trabajar en tiempo real y en nuestro caso no contamos con esta restricción por lo que existen mayor número de alternativas.

Se busca un método de ámbito de aplicación general pero no tanto como para que no se puedan establecer algunas restricciones que hagan posible la tarea. Por lo tanto vamos a suponer que al tener imágenes tomadas desde una altura considerable, estas imágenes se pueden considerar planas y por lo tanto la relación entre ambas se puede encontrar mediante una única transformación perspectiva.

Otro detalle que hay que tener en cuenta es que en nuestro caso queremos obtener una alineación lo más ajustada posible pero no necesitamos una correspondencia densa de todos los pixels y por lo tanto podemos descartar estas técnicas que realizan deformaciones de la imagen más complejas.

No vamos a suponer ninguna escala o rango de escalas y por este motivo nos sería útil un método que fuese capaz de operar con variaciones importantes de escala. Esto no va a ser un requisito muy estricto puesto que la mayoría de los métodos existentes no se comportan bien con los cambios de escala puesto que cuando hay mucha variación la pérdida de información es demasiado grande como para poder obtener cualquier tipo de resultado.

Basándonos de manera principal en los artículos de Brown [3] y Zitová [9],

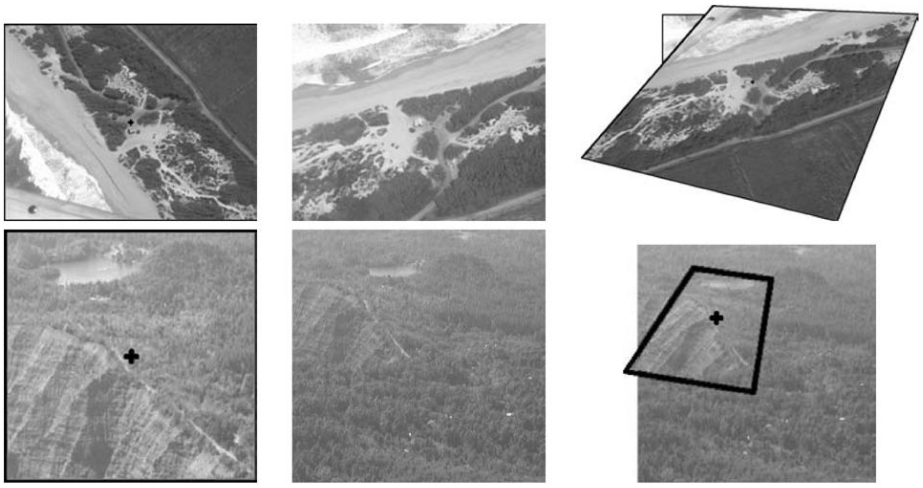


Figura 9: Imágenes aéreas
(a) Imágenes observadas (b) Imágenes de referencia (c) Alineamiento resultante

Figura 10: Caso con poca textura

junto con la información obtenida de otros artículos no referenciados, se ha decidido que el método que más se acercan a nuestras necesidades es el que proponen Zokay y Wolberg [1].

En la figura 9 se pueden observar un ejemplo, complejo de resolver con otros métodos, de uso en el caso de imágenes aéreas que se ajusta bastante a nuestros objetivos. Además tiene otros ejemplos como el mostrado en la figura 10. Este ejemplo está fuera del ámbito de aplicación objetivo, no obstante se trata de un caso de extrema dificultad para la mayoría de los métodos alternativos debido a la carencia de texturas y la uniformidad de iluminación de la imagen.

Este artículo [1], titulado *Image Registration Using Log-Polar Mappings for Recovery of Large-Scale Similarity and Projective Transforms*, cumple con nuestros requisitos y está muy bien documentado matemáticamente por lo que se ve factible su implementación sin suponer pasos intermedios no explicados o usar librerías externas de su propiedad o de terceros.

Se trata de un método híbrido que combina transformaciones al dominio log-polar junto a una optimización por mínimos cuadrados no lineales. La utilización de técnicas log-polar en el dominio espacial se utiliza como preprocesado para poder recuperar grandes cambios de escala y rotaciones arbitrarias y las otras componentes se ajustan de manera más tradicional.

El principal problema de todos los métodos de optimización es que pueden fallar si se quedan en algún mínimo local y esto es tanto más probable cuanto mayor sea la diferencia entre ambas imágenes y por lo tanto mayor camino a recorrer hasta encontrar la transformación que las relaciona. Esto se consigue solucionar con la alineación usando la transformación al dominio log-polar, que nos da una aproximación a la solución incluso con grandes cambios de escala y rotaciones arbitrarias. Por lo que se obtiene un buen punto de partida desde el que aproximar utilizando la optimización no lineal. De manera adicional se aplica el método haciendo uso de una pirámide en el espacio de escalas como se observa en la figura 11, de esta manera se trabaja a menor resolución en las primeras etapas y se va aumentando la resolución cuando se van obteniendo resultados positivos. Esta estrategia hace que muchos emparejamientos negativos sean detectados como tales con una baja resolución de las imágenes, esto hace que al tratar con un menor número de pixels este proceso se realiza de una manera mucho más rápida y eficiente ya que sólo se analizará la imagen completa en los casos más prometedores.

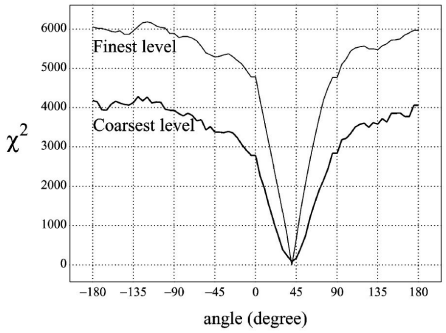


Figura 11: Variación de la función del error en función del ángulo , se trata de uno de los múltiples parámetros de transformación. Esto se muestra para dos de los distintos niveles en el espacio de escalas. De esta manera cuando se trabaja a menor resolución cuesta menos procesar las imágenes y por lo tanto "recorrer" esta curva de error hasta llegar al mínimo. Por otro lado el mínimo encontrado será menos preciso y por este motivo una vez encontrado este se procederá con el siguiente nivel del espacio de escalas, con la ventaja adicional de disponer de un valor inicial más próximo a la solución que también sirve para evitar el problema de los mínimos locales.

Otros dos métodos que funcionan bien con imágenes exteriores y grandes (>4x) cambios de escala, derivados de zoom óptico, son los métodos *Distinctive image features from scale-invariant keypoints* [10] y *Matching images with different resolutions* [11]. Ambos se basan en la detección de esquinas, puntos característicos de la imagen, y convertirlos en descriptores invariantes. Para luego emparejarlos basándose en la distancia de *Mahalanobis* o en un *k-d tree* y eliminar los *outliers* mediante un algoritmo de *RANSAC*. Por lo tanto estos métodos pueden fallar al alinear regiones suaves o poco detalladas, al estar diseñados para trabajar con los detalles de las texturas. Mientras que el método propuesto no tiene esta dependencia y por lo tanto puede trabajar con un conjunto de datos que puede ser el nuestro al tratar con imágenes de zonas muy

homogéneas.

El problema principal que tiene el método propuesto es que se trata de un algoritmo costoso computacionalmente comparado con los dos anteriores y que al trabajar de manera global con la imagen no es viable para aplicaciones en tiempo real. Aunque sea un proceso costoso este método usa una aproximación jerárquica por lo que comienza el análisis con la menor resolución posible y la va aumentando según es necesario en cada momento, esto hace que los cálculos más costosos se realicen a menor resolución y que sea necesario un menor número de ellos.

Otro punto que hay que tener en cuenta al tratar sobre el coste computacional es la complejidad y paralelización de cada fase del método. Mientras que los anteriores métodos utilizan operaciones complejas que no pueden ser paralelizadas por hardware el método propuesto se compone de operaciones simples que pueden implementarse en una *GPU* por lo que se puede computar de manera muy eficiente. Aunque esto no disminuye la complejidad y para tiempo real siguen siendo mucho más eficiente la otra aproximación.

Se ha analizado el método con detalle para su implementación pero se ha concluido con que es demasiado complejo para una primera iteración, lo que supondría demasiado tiempo de desarrollo y si apareciese algún error o parte no especificada adecuadamente por los autores, por error u omisión, provocaría que encontrar dicho fallo fuese complicado al no estar seguros sin realizar un análisis riguroso y en profundidad de las demostraciones matemáticas y de la correctitud de nuestra propia implementación. Otro inconveniente de este método es que al trabajar en el dominio log-polar hace que las etapas intermedias sean menos intuitivas y esto evita que se puedan realizar trazas que sean realmente útiles al tener que fiarnos de los resultados que nos devuelve el método sin poder realizar una traza manual del resultado esperado.

Por estos motivos se ha decidido empezar implementando dos métodos [2] [12] anteriores pero de los mismos autores. Siendo en particular estos dos artículos, la base sobre la que se ha construido el artículo que se pretende implementar una vez se tengan estos. Y por lo tanto se apoyan en la misma tecnología y similares restricciones, lo que nos permitirá aplicar el conocimiento adquirido para desarrollar el método objetivo de una manera mucho más rápida, al mismo tiempo que permitirá reutilizar la mayor parte del código generado.

En el artículo *Robust Image Registration Using Log-Polar* Transform [12] se realiza el alineamiento de imágenes en el dominio log-polar pero en un dominio más restringido ya que sólo es capaz de detectar transformaciones afines. No obstante de esta manera podemos testar la implementación de un subcaso del alineamiento en el dominio log-polar y verificar que funciona de manera adecuada, lo que nos daría bastante seguridad de en caso de fallo en el método objetivo no sea esta la parte que pueda ser la causante del problema. Y de manera adicional en el caso de facilitarle al método objetivo dos imágenes que se pudiesen alinear sólo mediante una transformación afín, el resultado y pasos intermedios deberían ser similares por lo que serviría como traza modelo para localizar o descartar donde se puede haber producido el error.

En el artículo *Image Registration For Perspective Deformation Recovery* [12] se propone la estimación de la transformación perspectiva de manera indirecta a través de múltiples estimaciones de transformaciones afines. Por lo tanto este artículo propone realizar el proceso en dos etapas, en la primera se dividen

las imágenes en tiles del mismo tamaño para poder calcular la función χ^2 y se realiza el alineamiento afín de todas ellas. En la segunda etapa, dado que hemos conseguido un conjunto de parámetros afines, se puede resolver un sistema de ecuaciones sobredeterminado para obtener los ocho parámetros de la transformación perspectiva buscada, siendo posible añadir más restricciones a la solución si tenemos en cuenta que conocemos los centros de dichas tiles.

2.3. Implementación

Este algoritmo, para hallar la transformación perspectiva, se basa en particionar las imágenes en un conjunto regular de tiles y alinear estas estimando únicamente la transformación afín que las relaciona. De esta manera una vez se obtiene este conjunto de datos relacionados, se puede estimar la transformación perspectiva que las relaciona. El número de particiones es arbitrario y dependerá del tiempo que se esté dispuesto a invertir en su resolución ya que al aumentar el número de tiles se aumenta el número de emparejamientos necesarios y por lo tanto el coste computacional pero a cambio se obtiene una mayor precisión en la estimación de los parámetros de la transformación.

La primera fase de la implementación ha consistido en construir el esquema exterior del algoritmo, de esta manera se pospone la implementación del método de optimización hasta haber podido realizar ciertas pruebas de rendimiento y usabilidad. En esta parte se ha llevado a cabo la preparación de los datos y se ha decidido como particionar el conjunto de tiles. En concreto se ha decidido no obtener el conjunto como tal, en su lugar se ha preferido ir trabajando sobre la región correspondiente a cada tile en las propias imágenes puesto que esto se puede realizar de manera eficiente sin duplicar información de manera innecesaria.

Esta parte se ha realizado de tres formas diferentes (C++ con OpenCV, Pyhton con OpenCV y Python con PIL) con el objetivo de escoger la que mejor se adapte a nuestro propósito, véase el anexo 4.16 para más detalles de los lenguajes contemplados. Se ha partido de la base de utilizar C++ por ser el lenguaje de referencia en la mayoría de los trabajos y la documentación asociada que esto nos proporciona. Junto con OpenCV que nos proporciona un conjunto de funciones asociadas con el manejo de imágenes, ya que esta librería se encarga de la carga de las mismas sin preocuparnos del formato en el que se encuentren y la forma de manejarlas es muy eficiente ya que puede trabajar sin duplicarlas al trabajar mediante referencias, junto con el conteo y gestión de las mismas. Esta característica se da incluso en el acceso a subregiones, lo que en nuestro caso se traduce en un acceso eficiente y en no necesitar estructuras de datos más complejas para manejarlas.

Se ha mostrado interés en Python porque se trata de un lenguaje que ofrece una elevada productividad y expresividad, aunque esto haga que de manera inevitable sea más lento que otros lenguajes compilados. No obstante, si en un futuro fuera necesario aumentar la velocidad de ejecución de este lenguaje, se podría optar por RPython (Restricted Python) que es un subconjunto de Python que se puede portar a código C de manera automática y genera código eficiente. O bien por Cython que al convertirlo en un lenguaje tipado permite compilar Python a C y al mismo tiempo permite llamar funciones y métodos

de C/C++, lo que ofrece la posibilidad de optimizar los cuellos de botella del método sin necesidad de modificar toda la implementación.

Python cuenta con PIL(Python Imaging Library) que se trata de una extensión del lenguaje que ofrece primitivas básicas para trabajar con imágenes. Aunque es bastante limitada en principio es suficiente para el propósito del algoritmo que se va a implementar puesto que las únicas funciones que vamos a delegar en ella son la carga de imágenes y la interpolación al reescalar. Al realizar la implementación del esquema exterior de esta manera se ha observado que se trabaja de manera muy cómoda porque el acceso a las subregiones se realiza de manera muy intuitiva y simple. El inconveniente encontrado al hacerlo de esta manera es que el tiempo de ejecución mayor que el de C++ en exceso y esto teniendo en cuenta que no se ha implementado la parte computacionalmente costosa del algoritmo.

Esto ha hecho desestimar esta aproximación, pero no el lenguaje Python que ha permitido prototipar la implementación de una manera bastante rápida. Para mantener el lenguaje y al mismo tiempo ganar eficiencia se ha hecho uso de la interfaz para Python que nos ofrece OpenCV. Esta interfaz la ofrece de manera nativa y está documentada junto a la documentación de C++. Esto es posible por la forma de trabajar de OpenCV, ya que hace uso de referencias y maneja el conteo de las mismas de manera automática, por lo que es compatible con la gestión de memoria dinámica de Python. Esta implementación ha resultado ser similar en tiempo de ejecución a la de C++, lo que hace tenerla en cuenta como muy buena opción para implementar todo tipo de métodos de manera rápida. El inconveniente es que algunos detalles de la interfaz y la forma de trabajar de Python han hecho que para acceder a algunas propiedades de las imágenes, en concreto las subregiones, haya sido necesario hacerlo de manera indirecta y de forma no muy intuitiva. Esto ha supuesto una pérdida de tiempo considerable, lo que ha hecho replantear esta opción, por lo menos de momento, porque este inconveniente hace que resulte más rápido el uso de un lenguaje al que se está más acostumbrado como C++.

Una vez que se ha decidido el lenguaje se ha procedido al estudio minucioso del artículo para su implementación. Para ello ha sido necesario recurrir a bibliografía sobre métodos matemáticos aplicados en el ámbito de la visión por computador [13] [14] [15] [16] [17]. Esto nos ha dotado de las herramientas necesarias para poder comprender el funcionamiento del método y para poder implementarlo de manera adecuada. Por lo tanto se ha comenzado por implementar el método de optimización no lineal LMA (Levenberg-Marquardt Algorithm), que es una combinación del método de Newton-Raphson y el de descenso del gradiente.

La implementación del método LMA ha sido relativamente rápida porque en el artículo se especifican todos los detalles de manera adecuada. Sólo se ha encontrado algún inconveniente al tener que derivar la función que mide la similitud entre las imágenes, puesto que se trata de una función multidimensional. Y al elegir como resolver el sistema de ecuaciones lineales porque sobre esto hay demasiados métodos para elegir dependiendo de las condiciones concretas de las ecuaciones. En este caso no son ecuaciones que requieran un tratamiento muy complejo y se ha optado por resolverlo usando la pseudoinversa, porque no hace falta calcular la inversa y así se evitan posibles problemas.

Como paso previo a construir el resto del algoritmo e integrarlo con él, se

han realizado pruebas para evaluar su correcto funcionamiento. De estas pruebas no se han obtenido resultados satisfactorios, lo que ha conllevado una revisión de la solución implementada. Al no encontrarse ningún problema en la parte algorítmica se han buscado los detalles de implementación y en dicha revisión se ha observado que había un error en el acceso a los datos de las imágenes debido a la forma en que estas son almacenadas y como eran accedidas por los métodos.

Una vez solucionados estos errores de implementación se han vuelto a obtener resultados no satisfactorios, esto ha hecho necesaria una revisión muy detallada de todos los pasos del algoritmo. Lo primero en revisarse ha sido la resolución del sistema de ecuaciones al ser lo que gobierna el funcionamiento del algoritmo. Para ello se han usado distintos métodos de resolución de ecuaciones pero de todos ellos se ha obtenido el mismo resultado. Esto nos ha hecho centrar la atención en los datos con los que se resuelve el sistema de ecuaciones, estos son la Hessiana(matriz de derivadas segundas) y B(vector de derivadas), y en su cálculo se ha encontrado y arreglado algún error a causa de la precisión de los cálculos pero no tan grave como para hacer fallar el algoritmo.

Al no encontrar ningún error en el código, se ha procedido a revisar las demostraciones matemáticas en las que se basa el algoritmo. Tras un análisis detallado se ha llegado a la conclusión de que había una errata. Esto hacía que el sistema de ecuaciones no dependiera del parámetro que disminuye o aumenta la resolución de la solución según estemos más cerca o más lejos del mínimo buscado.

Una vez solucionado este problema de diseño se ha seguido obteniendo una solución similar. Esto ha conllevado a una revisión y verificación de todo el código escrito y funciones utilizadas. Se ha verificado cualquier detalle que pudiera ser una fuente de errores, por ejemplo se ha comprobado que el paso de parámetros se realizase de manera correcta y que se estuviera trabajando con la matriz de transformación correcta y no con su traspuesta y se ha revisado cada una de las funciones y parámetros utilizados en busca de cualquier posible error.

Al no encontrar la causa del funcionamiento incorrecto del algoritmo, se ha vuelto a revisar como se calculan los datos más relevantes para el algoritmo. Como el cálculo de la Hessiana y el vector B depende de la transformación de la imagen, se ha revisado este aspecto. Hasta este momento se había usado una función de OpenCV de transformación perspectiva porque la de transformación afín es un subconjunto de esta y al no variar los parámetros de perspectiva la transformación obtenida es afín igualmente y por lo tanto también se han hecho pruebas para verificar que esto era correcto. Al ser esta la única función que se ha usado de OpenCV (a excepción de cargar las imágenes y mostrarlas) se ha pretendido prescindir de ella y por lo tanto se ha implementado una función de transformación propia. El problema que hemos encontrado aquí es que la interpolación implementada ha sido menos refinada que la que realiza OpenCV de manera interna y esto ha resultado en un empeoramiento del resultado. Al no disponer de una forma alternativa adecuada para calcular dicha transformación, se ha modificado como se calcula la Hessiana y el vector B para que no fuesen dependientes de esta transformación por si esta no estuviese haciendo su función de manera adecuada.

Se han construido casos de prueba triviales para verificar el comportamiento del algoritmo frente a ellos pero también ha fallado, aunque esto nos ha ofre-

Figura 12: Variación del resultado en función del incremento/decremento
La imagen central se corresponde a I2 transformada en I1 con la matriz de transformación resultante.

cido alguna pista ya que se ha observado que no se dan los saltos del tamaño suficiente para alcanzar la solución y por lo tanto el algoritmo se queda estancado en un mínimo local. Por lo tanto se ha variado la precisión con la que se tratan las imágenes y todos los datos intermedios pero tampoco se ha llegado a una mejora significativa. Siguiendo con el análisis de la traza se ha observado que el incremento hacia la solución(la solución del sistema de ecuaciones) iba descendiendo aunque no nos estuviéramos acercando a la solución. Esto aunque contra intuitivo es un resultado que se ajusta a la descripción matemática del algoritmo, ya que de manera simplificada este parámetro afecta "dividiendo" al sistema de ecuaciones y por lo tanto hace que cada vez la solución sea menor.

Como los únicos parámetros con los que se puede ajustar el método son el tamaño de aumento o disminución del paso y los umbrales de terminación del paso y del error, a lo que se ha procedido es a preparar una serie de pruebas con distintos valores para estos parámetros. Lo que se ha observado es que el efecto de los umbrales en el resultado final ha sido mínimo pero no así en el tiempo de ejecución. El que mayor influencia ha tenido, como se muestra en la figura 12, es el incremento o decremento del paso. El método es muy dependiente de este valor pero no muestra una dependencia lineal, esto hace que los resultados sean demasiado variables. El principal inconveniente de esto es que lo hace muy dependiente de los datos concretos con los que se trabaja, lo que implica que aunque se consiguiera ajustar un parámetro adecuado sólo sería válido para esos datos. Además, esto no nos proporciona información útil para ajustar el parámetro para otros datos.

Al seguir sin encontrar una solución satisfactoria se ha decidido intentar validar el algoritmo implementándolo en matlab. De esta forma si los resultados obtenidos son distintos y correctos se encontraría de manera inmediata donde se está cometiendo el error. Mientras que si se obtienen los mismos resultados se podrá descartar un problema de implementación y en todo caso denotaría un problema en el diseño o interpretación del algoritmo. Esto podría ser posible porque a los autores del artículo se les hubiera olvidado incluir alguna etapa necesaria para el correcto funcionamiento del método.

Para volver a implementar este algoritmo en matlab ha hecho falta muy poco tiempo puesto que ya se tenía el conocimiento necesario adquirido al haberlo implementado en otro lenguaje. Aunque los resultados obtenidos por esta nueva implementación han seguido siendo negativos, se ha encontrado que en el artículo había una interpretación ambigua referida al comienzo de coordenadas, aunque ha resultado ser irrelevante para los resultados.

Al no encontrar errores ni en la implementación ni en diseño del método se han estudiado alternativas a la resolución del mismo pero de tal manera que estas no modifiquen la forma básica de funcionamiento para así poder seguir utilizando el mismo esquema general. Se han estudiado métodos de premio-castigo y otros muchos métodos de optimización no lineal, pero todos ellos requerían modificaciones demasiado importantes en el esquema del algoritmo y esto habría supuesto volver a realizar todo el trabajo desde el principio. Por lo tanto como alternativa viable se ha optado por intentar resolverlo mediante las herramientas de optimización que ofrece matlab. El inconveniente de usar estas funciones es que están pensadas para resolver un amplio espectro de problemas, esto hace que ofrezcan mucha más funcionalidad de la que se necesita en este caso concreto y a su vez demasiados parámetros para ajustar y seleccionar el comportamiento

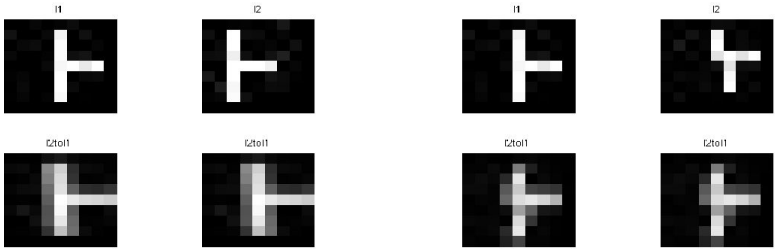


Figura 13: Resultados con fminsearch

de los métodos ofrecidos. Al final se ha optado por utilizar fminsearch que encuentra el mínimo de una función de variables múltiples sin restricciones y sin necesidad de usar las derivadas de la misma.

Los resultados usando fminsearch que se observan en la figura 13, son mejores que los que se habían obtenido con anterioridad. Esto es cierto en el caso de estas imágenes sintéticas con mucho contraste y de poco tamaño, ya que en el caso de imágenes de mayor tamaño los resultados vuelven a ser inservibles.

Esta experiencia no ha resultado del todo infructuosa ya que se ha observado una pequeña mejoría en el resultado y al analizar de manera concienzuda por que ha sucedido esto se ha concluido que la posible causa del mal funcionamiento del método sea debido al cálculo incorrecto de las derivadas.

Tras el análisis de las derivadas se ha llegado a la conclusión de que es posible que al realizar una proyección se pierda información. Por este motivo se ha realizado una implementación alternativa donde se trabaja en el espacio inverso4.9. Para ello ha sido necesario realizar el cálculo de la inversa de la matriz con los parámetros de transformación de manera analítica. Una vez obtenida esta nueva matriz de transformación se han calculado de manera analítica las derivadas (vector B) y las derivadas segundas (hessiana). Una vez resuelto este proceso, el método resultante es similar pero cambiando el espacio donde se buscan los valores transformados. Esta alternativa tampoco ha resultado en resultados satisfactorios, por lo que ha sido necesario seguir buscando formas alternativas de resolver el problema.

Se sigue suponiendo que el problema son las derivadas, puesto que se ha vuelto a testar toda la implementación en busca de posibles errores y para ello se ha realizado una implementación que permite variar parámetros ajenos al método en si pero con una fuerte influencia en los resultados. Cabe destacar el formato de trabajo en coma flotante de las imágenes ya que son almacenadas como valores entre 0 y 255. En busca de errores en este aspecto se han realizado pruebas con todas las posibles combinaciones pero no se ha llegado a ninguna conclusión satisfactoria que la hipótesis inicial con la que se trabajaba. Si bien en algún caso particular se han conseguido resultados más prometedores, estos no han sido consistentes al realizar más pruebas con otro datos.

Las funciones tanto de OpenCV (*warpPerspective()* y *warpAffine()*) como de Matlab (*imtransform()*) no están concebidas para tratar con este tipo de transformaciones proyectivas que son más comunes en el ámbito de los gráficos por computador puesto que en este ámbito si que es necesario realizar estas

proyecciones de un espacio 3D a un espacio 2D para obtener la vista de una cámara. No obstante las funciones con las que se ha trabajado no lo soportan y devuelven un resultado técnicamente correcto pero incorrecto en el ámbito en el que se está trabajando. Esto es a causa de la forma en que trabajan estas funciones ya que para hacer su trabajo de una manera más eficiente trabajan con la inversa de la matriz de transformación y en este caso esta matriz al ser singular no tiene inversa. Visto de otra manera el problema que tienen estas funciones es la interpretación de la proyección, que en el caso de estas funciones es "encoger" la imagen hasta un vector o punto con valor nulo y por lo tanto esto no aporta ninguna información.

Se ha tratado de solucionar este inconveniente en la función *imtransform* mediante el uso de la función *maketform* con la que se pueden generar transformaciones arbitrarias que luego pueden ser interpretadas por *imtransform*. Esta función trabaja con la inversa de la transformación y por lo tanto ha sido necesario implementar una función de transformación inversa general y una función de transformación inversa para cada una de las derivadas de primer orden y de segundo orden, en función de los 8 parámetros de una transformación perspectiva.

En el caso que se está tratando es necesario disponer de esta información puesto que se trata de la información básica con la que trabaja el método LMA y esto ha hecho necesario implementar estas funciones. Por un lado hay que decidir como realizar la transformación de una imagen de un espacio a otro, en este caso se ha decidido por simplicidad realizar la transformación directa. Por otro lado ha sido necesario interpretar la proyección, para ello se han estudiado ocho formas distintas de realizarla.4.10

Mientras tanto se han seguido buscando soluciones alternativas para el cálculo de las derivadas y se ha buscado la manera de prescindir del cálculo de la derivada segunda en el cálculo de la hessiana. Esta manera aproximada de obtener la hessiana ha sido posible puesto que la información que aporta la derivada segunda no es representativa del problema en el que se está trabajando y en consecuencia o bien aporta ruido o bien un valor constante que no afecta a la solución. No obstante este valor suele ser despreciable en comparación con el que aporta la primera derivada, por lo tanto no se esperaba un cambio efectivo en el resultado y esto ha sido lo que se ha podido observar al ponerlo a prueba. Por otro lado, también como esperábamos se han obtenido resultados menos variables en función de los parámetros del algoritmo pero igual de incorrectos.

Otra solución implementada ha consistido en realizar el cálculo de las derivadas de manera numérica como realiza matlab por defecto en la función *fminsearch*, comentada anteriormente. Esta forma de proceder es más costosa al tener que realizar el cálculo de la transformación para el incremento de cada parámetro y ofrece valores aproximados por lo que es mucho más dependiente de los datos concretos. En concreto es necesario realizar dos transformaciones por cada derivada de primer orden y de tres a cuatro transformaciones por cada derivada de segundo orden. Otro inconveniente que tiene esta forma de proceder es que hay que escoger de manera arbitraria un tamaño de incremento para los parámetros. Además hay que tener en consideración que el cálculo de las derivadas no es el mismo en todos los casos y por lo tanto este incremento aunque proporcional tampoco es el mismo. Todo esto ha llevado a tener más parámetros para modificar y ajustar, por ello se han realizado una serie de pruebas en fun-

ción del tamaño del incremento para comprobar cual resultaba más adecuado. El problema encontrado con esta forma de proceder es que ha resultado en exceso dependiente del valor asignado al incremento. Como ejemplo, entre incrementos de tan solo una milésima se han obtenido resultados completamente diferentes y sin seguir ningún patrón de convergencia hacia una solución común. Por otro lado en otros casos el valor del incremento ha sido demasiado pequeño, como consecuencia el método no obtenía valores adecuados haciendo que el camino hacia la solución fuese incorrecto.

Se ha seguido buscando solucionar el problema con este último método y para ello se ha decidido implementar una variante del mismo que tomase el valor del incremento de manera dinámica y para ello se han estudiado varias posibilidades. La primera implementada ha sido hacer que el tamaño del incremento fuese proporcional al valor que tomase el parámetro en el momento de la consulta. Según fuentes consultadas a tal efecto se recomendaba un valor proporcional a 10^{-8} que es el mínimo valor en el que se puede incrementar el 1 en coma flotante, aunque también se han probado otros valores mayores para que la variación experimentada fuese mayor. No obstante el problema encontrado en esta implementación ha sido que el valor de los parámetros de transformación era demasiado próximo a cero, o cero, en la mayoría de los casos y esto ha provocado demasiada poca variación en el valor de las derivadas. Todo esto ha hecho que esta aproximación lejos de conseguir resultados satisfactorios haya resultado ser contraproducente.

En la misma línea se ha contemplado la variación independiente del incremento de cada parámetro en función del valor λ que gobierna el funcionamiento del algoritmo LMA. Esta aproximación ha resultado demasiado compleja al requerir almacenar una traza de cada parámetro y tener en cuenta la variación que representaba cada combinación en una función multivariable. Esto ha hecho que esta aproximación se considerase demasiado esfuerzo y se desestimase antes de seguir la implementación al no tener la demostración matemática que garantizase su correcto funcionamiento ni la certeza o convicción de que esta forma de proceder fuese a reportar una solución mejor que en caso anterior.

Por lo tanto se ha continuado explorando la variación del incremento de manera homogénea para todos los parámetros involucrados. A tal efecto y con la influencia del estudio anterior, se ha considerado conveniente la variación del incremento en función del parámetro λ, que gobierna el método Levenberg-Marquardt. Esta aproximación hace que conforme el método se acerca a un mínimo el intervalo o incremento en el que se calcula la derivada sea menor y por lo tanto más preciso. Mientras que cuando estamos lejos de la solución o se considera que se está en un mínimo local, el valor del incremento aumenta permitiendo una mayor variación en la derivada que hace dirigir la búsqueda a mayor distancia ofreciendo la dirección a seguir en las sucesivas iteraciones.

Esta vez los resultados han sido satisfactorios, como se puede observar en la figura 14 y por lo tanto se ha procedido a realizar más pruebas con distintos conjuntos de datos. Se ha comprobado en un caso anterior que un resultado prometedor en imágenes de pequeño tamaño no significa que se vaya a obtener un resultado significativamente distinto en imágenes de mayor tamaño. Para ello se han utilizado dos imágenes de 512x512 pixels, siendo una de ellas generada de manera sintética a partir de la otra para obtener un caso de prueba que tenga una posible solución. El resultado de esta prueba se puede observar en la figura

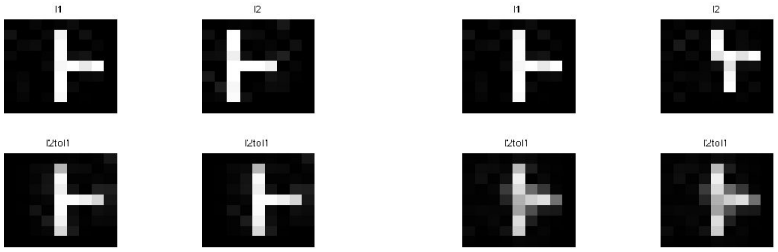
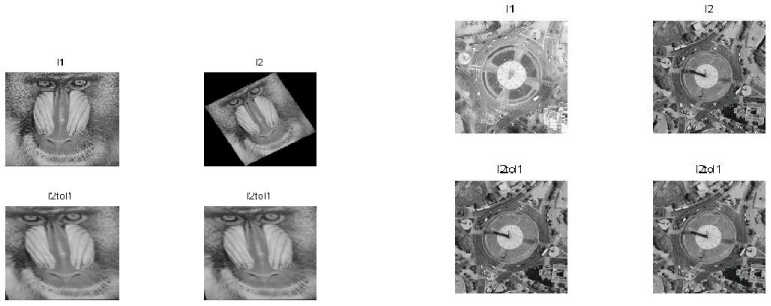


Figura 14: Resultados con derivadas numéricas y variación del incremento en función de λ



(a) Datos sintéticos

(b) Datos reales

Figura 15: Resultados con derivadas numéricas y variación del incremento en función de λ en imágenes de mayor tamaño

15a] que no ofrece la precisión que se hubiese deseado. No obstante si se tienen en consideración los resultados obtenidos con los métodos anteriores se trata de un buen resultado. Para estar más seguros de que el método se va a comportar de manera correcta en la mayoría de las situaciones a las que puede ser sometido se ha considerado oportuno, sin llegar a realizar un estudio exhaustivo, probarlo con datos reales. Para ello se han tomado las imágenes que han originado la búsqueda alternativa a los métodos basados en características y los resultados se pueden observar en la figura 15b, que vuelven a obtener un nivel de precisión similar al anterior y por lo tanto se puede considerar como satisfactorio.

El inconveniente de esta aproximación es el coste computacional, como se ha anticipado anteriormente. Si bien en los métodos en los que las derivadas se calculaban de manera analítica y luego se evaluaba su valor el tiempo de ejecución era alto pero no excesivo, en este caso ha resultado muy elevado teniendo en cuenta que las imágenes eran de sólo 512x512 pixels y ha costado de media unas 22h de cálculo de media. Esta diferencia tan abrumadora hace esperar que se siga una progresión cuadrática o cúbica y en el ámbito de imágenes mayores resultaría en un tiempo de cálculo inaceptable.

3. Conclusiones y futuras líneas de investigación

Como lenguaje de implementación, Matlab ha demostrado ser el más práctico por su rapidez de prototipado. En su mayor parte esto ha sido así porque la funcionalidad necesaria ha sido implementada y no se han necesitado funciones implementadas por terceros. Por otro lado C++ junto a OpenCV se han mostrado mucho más versátiles porque este último ofrece un conjunto de funcionalidad muy amplio en este dominio de aplicación, aunque este sólo haya sido utilizado en el prototipo inicial basado en características por tener una implementación de SURF.

El método basado en características *SURF* ha obtenido unos resultados aceptables en dominios controlados. El problema de este tipo de métodos es que como ha ocurrido en el caso estudiado, un caso que se consideraba trivial no ha sido capaz de resolverlo e incluso el método que tenía disponible el experto lo ha conseguido de manera forzada ajustando los parámetros. No obstante visto los resultados de las otras técnicas habría que seguir explorando en esta línea. Para ello sería necesario construir algoritmos más robustos y buscar descriptores que sean más adecuados para este ámbito de aplicación. También habría que buscar un método de validación del resultado estimado para obtener un intervalo de confianza donde poder estimar la verosimilitud del resultado obtenido de manera automática. Esto es necesario puesto que en los casos probados se depende de la validación visual del resultado a excepción de casos muy evidentes. La gran ventaja de estos métodos es la posibilidad de clasificar los descriptores mediante un *k-d tree* o *Scalable Vocabulary Tree (SVT)*, seleccionar las más representativas con un algoritmo tipo *PCA* y construir una tabla de búsqueda invertida, de esta manera se pueden obtener de manera muy rápida los posibles candidatos y realizar un ajuste posterior.

Por otro lado el método implementado, basado en la intensidad, ha ofrecido unos resultados aceptables pero no con la precisión esperada. Esto unido al alto coste de cálculo necesario para llegar a este resultado y sobre todo teniendo en cuenta la progresión observada, hace que no resulte viable aplicarlo en casos reales donde se espera trabajar con tamaños de imagen mucho mayores a los empleados en nuestros casos de prueba. Para seguir esta línea habría que implementar el algoritmo descrito en *Image Registration Using Log-Polar Mappings for Recovery of Large-Scale Similarity and Projective Transformations* [1], ya que debido a que trabaja en el dominio log-polar y las optimizaciones añadidas sobre el método que se ha implementado promete unos resultados de mayor calidad a los obtenidos. Si esta opción se mostrase viable se podría tener en cuenta una aproximación híbrida, donde un método basado en características ofreciese una transformación aproximada y se refinara haciendo uso de este método. Puesto que este método estimaría mejor la transformación una vez se está cerca de la solución, pero su coste también sería demasiado elevado como para utilizarlo como primera opción.

Otra posible línea de investigación, relacionada tanto con los métodos basados en descriptores pero quizás más relevante en el caso de los métodos basados en intensidad, se trata de mejorar la medida de similitud entre las imágenes. Esta se ha mostrado poco informativa a la hora de ofrecer indicios de como llegar a la solución. En la parte que sólo intervienen transformaciones afines, se puede usar el domino log-polar o la transformación al espacio de frecuencias.

Esta aproximación ofrece buenos resultados pero en el caso de transformaciones perspectivas habría que seguir buscando una medida de similitud mejor que la simple correlación de las imágenes.

Agradecimientos

Este proyecto fin de carrera ha resultado ser mucho más árduo y dilatado en el tiempo de lo estimado. Esto ha provocado que haya habido muchos momentos en los que se ha estado cerca de tirar la toalla al no ver camino que llevase a una solución. Es por ello que quiero mostrar mi agradecimiento a las siguientes personas:

Pedro R. Muro Medrano, por ser el director de este proyecto.

Francisco Javier Zarazaga Soria y Walter Rentería Agualimpia, por su inestimable ayuda en los momentos más difíciles y el tiempo que han dedicado a ayudarme a encontrar posibles errores y sobre todo posibles soluciones.

Josechu Guerrero Campo, por su ayuda como experto para poder evaluar y comparar nuestros resultados con los suyos, indudablemente mejores debido a su experiencia en el campo.

Vinod Kumar Kudithi, que aunque ha estado poco tiempo en el departamento ha colaborado en la fase inicial aportando ideas sobre los distintos algoritmos a probar.

Esperanza Luna Tolosana, Sara Echevarría Benito y Carolina Palacio Julián, por aportar amistad y alegría a todos los días que he compartido con ellas.

También quiero expresar mi agradecimiento de manera general a todos los integrantes del IAAA, por ser un grupo tan acogedor y cohesionado.

Y sobre todo a Mónica Otero Martínez y a mi familia que me han apoyado de manera incondicional para poder sacar adelante este proyecto y la carrera.

4. Anexos

En estos anexos se van a detallar de manera resumida los algoritmos y definiciones matemáticas implicadas en el desarrollo de este proyecto fin de carrera. Para ello se ha dividido en secciones cada una dedicada a un aspecto concreto de cada una de las partes involucradas.

4.1. Modelos no lineales

Se considera el ajuste de un modelo que depende, de manera no lineal, de un conjunto M de parámetros desconocidos, $a_k, k = 0, 1, \ldots, M-1$. La aproximación a su resolución es definir una función de mérito χ^2 y determinar el mejor ajuste de los parámetros mediante su minimización. Al estar tratando con dependencias no lineales la minimización debe realizarse de manera iterativa. Esto es, dado unos valores de inicio para los parámetros se requiere un procedimiento que mejore esta solución. Procedimiento que será repetido hasta que χ^2 llegue al mínimo o disminuya de un umbral asignado.

Este caso de optimización no lineal es diferente al caso general, porque cuando se está cerca del mínimo, se espera que la función χ^2 sea aproximada de manera adecuada de manera cuadrática, lo que se puede representar de la siguiente manera

$$\chi^2(a) \approx \gamma - d \cdot a + \tfrac{1}{2} a \cdot D \cdot a \tag{4.1.1}$$

donde d es un vector de M componentes y D es una matriz de M×M.

Si la aproximación es buena, se sabe como saltar desde la posición actual, dada por el conjunto de parámetros a_cur, hasta el conjunto solución a_min en un salto.

$$a_{min} = a_{cur} + D^{-1} \cdot \left[-\nabla\chi^2(a_{cur})\right] \tag{4.1.2}$$

Por otro lado, la expresión 4.1.1 puede ser una mala aproximación local a la forma de la función que se está tratando de minimizar en a_cur. En este caso, lo que se puede hacer es dar un paso en la dirección del gradiente, como se hace en el método de descenso del gradiente. Siendo la constante lo suficientemente pequeña para no sobrepasar el mínimo, esto se puede expresar de la siguiente manera

$$a_{next} = a_{cur} - constant \times \nabla\chi^2(a_{cur}) \tag{4.1.3}$$

Para poder utilizar las ecuaciones 4.1.2 o 4.1.3, se debe ser capaz de calcular el gradiente de la función χ^2 para cualquier conjunto de parámetros a. Para usar la ecuación 4.1.2 se necesita además la matriz D, que es la matriz de derivadas de segundo orden (Hessiana) de la función de mérito χ^2 para cualquier conjunto a dado.

En un caso general estas derivadas no tienen por que ser conocidas pero en este caso concreto se conoce la forma de χ^2 puesto que se basa en el modelo especificado. Por lo tanto se podrá utilizar la ecuación 4.1.2 en cualquier momento y el motivo de utilizar la ecuación 4.1.3 será en caso de fallo de la 4.1.2 para mejorar el ajuste.

4.2. Cálculo del Gradiente y la Hessiana

El modelo a ajustar es el siguiente

$$y = y(x|a) \tag{4.2.1}$$

y la siguiente función χ^2 es la función de mérito o similitud

$$\chi^2(a) = \sum_{i=0}^{N-1} \left[\frac{y_i - y(x_i|a)}{\sigma_i} \right]^2 \tag{4.2.2}$$

El gradiente de χ^2 respecto a los parámetros a, cuyo valor será cero en el mínimo de χ^2, tiene las siguientes componentes

$$\frac{\partial \chi^2(a)}{\partial a_k} = -2 \sum_{i=0}^{N-1} \left(\frac{[y_i - y(x_i|a)]}{\sigma_i^2} \frac{\partial y(x_i|a)}{\partial a_k} \right) \ , \ k = 0, 1, \ldots, M-1 \tag{4.2.3}$$

Tomando una derivada parcial adicional se tiene la Hessiana

$$\frac{\partial^2 \chi^2(a)}{\partial a_k \partial a_l} = 2 \sum_{i=0}^{N-1} \left(\frac{1}{\sigma_i^2} \left[\frac{\partial y(x_i|a)}{\partial a_k} \frac{\partial y(x_i|a)}{\partial a_l} - [y_i - y(x_i|a)] \frac{\partial^2 y(x_i|a)}{\partial a_k \partial a_l}) \right] \right) \tag{4.2.4}$$

Es habitual eliminar el factor 2 definiendo las siguientes equivalencias

$$\beta_k \equiv -\tfrac{1}{2} \frac{\partial \chi^2}{\partial a_k} \quad \alpha_{kl} \equiv \tfrac{1}{2} \frac{\partial^2 \chi^2}{\partial a_k \partial a_l} \tag{4.2.5}$$

haciendo $\alpha = \frac{1}{2} D$ en la ecuación 4.1.2, en estos términos dicha ecuación se puede rescribir como el siguiente conjunto de ecuaciones lineales

$$\sum_{l=0}^{M-1} (\alpha_{kl} \delta a_l) = \beta_k \tag{4.2.6}$$

Este conjunto se resuelve para los incrementos de δa_l, que son añadidos a la aproximación actual. En el contexto de los mínimos cuadrados, la matriz α, igual un medio de la matriz Hessiana, se denomina de manera habitual como matriz de curvatura.

La ecuación 4.1.3, la fórmula del descenso del gradiente, se traduce en la siguiente expresión

$$\delta a_l = constant \times \beta_l \tag{4.2.7}$$

Los componentes α_{kl} de la matriz Hessiana 4.2.4 dependen de la primera y segunda derivada de la función de mérito con respecto a sus parámetros.

La derivada de segundo orden aparece porque el gradiente 4.2.3 ya tiene una dependencia de $\frac{\partial y}{\partial a_k}$, y por tanto la siguiente derivada debe contener algún

término relacionado con $\frac{\partial^2 y}{\partial a_l \partial a_k}$. El término de la derivada de segundo orden se puede descartar cuando esta es cero o cuando es despreciable en relación al término involucrado en la derivada de primer orden. De hecho hay otra forma de garantizar que este término sea lo suficientemente pequeño en la práctica para poder despreciarlo. Esto es porque el término que multiplica a la derivada de segundo orden en la ecuación 4.2.4 es $[y_i - y(x_i|a)]$ y para que se trate de un modelo viable el término debería ser la medida del error aleatorio en cada punto. Este error puede ser positivo o negativo y en general no debería estar relacionado con el modelo. Por lo tanto los términos de la derivada de segundo orden tenderán a cancelarse cuando sean sumados.

Es más, la inclusión del término de la derivada de segundo orden puede causar que el ajuste del modelo no sea estable o quede sesgado por datos espurios que no serán compensados por términos similares de signo contrario. Por todo esto se utilizará la siguiente aproximación que hace tomar a la derivada de segundo orden un valor cero. Esto no afecta al resultado porque la modificación de estos parámetros no altera el resultado final, sólo el camino hacia esta solución. Esto es debido a que la condición en el mínimo de χ^2, $\beta_k = 0, \forall k$, es independiente de como haya sido definido α.

$$\alpha_{kl} = \sum_{i_0}^{N-1} \left(\frac{1}{\sigma_i^2} \left[\frac{\partial y(x_i|a)}{\partial a_k} \frac{\partial y(x_i|a)}{\partial a_l} \right] \right) \tag{4.2.8}$$

4.3. Método Levenberg-Marquardt

El método de Marquardt se basa en una sugestión de Levenberg para variar de manera suave entre los extremos del método de la inversa de la Hessiana 4.2.6 y el método del descenso del gradiente 4.2.7. Este último método se utiliza cuando se está lejos del mínimo, cambiando de manera continua al primero mientras el mínimo es alcanzado.

Este método está basado en dos ideas, elementales pero importantes. Si se considera la "constante" en la ecuación 4.2.7, esta sólo contiene la información sobre el gradiente. No contiene información respecto al orden de magnitud o a la escala, sólo contiene información sobre la pendiente y tampoco hasta donde llega esta pendiente. La primera idea de Marquardt es que las componentes de la Hessiana, incluso si no son usables, dan información sobre el orden de magnitud de la escala del problema.

La función χ^2 es adimensional como se puede observar en 4.2.2. Por otro lado β_k tiene las dimensiones de $\frac{1}{a_k}$ que pueden ser de la forma de cm^{-1} o kwh o similares, de hecho cada componente de β_k podría tener dimensiones diferentes. Por lo tanto la constante de proporcionalidad entre β_k y δa_k deberá tener las dimensiones de a_k^2. Al examinar las componentes de α, se observa que existe una única cantidad obvia con estas dimensiones, esta es $\frac{1}{\alpha_{kk}}$, el recíproco del elemento de la diagonal. Por tanto esto debe establecer la escala de la constante. Pero la escala puede ser en si misma demasiado grande y hay que dividir la constante por algún factor adimensional λ, con la posibilidad de establecer $\lambda \gg 1$ para detener el paso. Expresado de otra manera, se remplaza la ecuación 4.2.7 por

$$\delta a_l = \frac{1}{\lambda \alpha_{ll}} \beta_l \quad or \quad \lambda \alpha_{ll} \delta a_l = \beta_l \tag{4.3.1}$$

La segunda idea de Marquardt es que las ecuaciones 4.3.1 y 4.2.6 pueden ser combinadas si se define una nueva matriz α' con la siguiente definición:

$$\begin{aligned} \alpha'_{jj} &\equiv \alpha_{jj}(1+\lambda) \\ \alpha'_{jk} &\equiv \alpha_{jk} \qquad\qquad (j \neq k) \end{aligned} \tag{4.3.2}$$

y entonces remplazar ambas 4.3.1 y 4.2.6 por

$$\sum_{l=0}^{M-1} \alpha'_{kl} \delta a_l = \beta_k \tag{4.3.3}$$

Cuando λ es muy grande, la matriz α' es forzada a ser diagonal dominante, por tanto la ecuación 4.3.3 tiende a ser idéntica a 4.3.1. Por otro lado cuando λ se aproxima a cero, la ecuación 4.3.3 tiende a ser 4.2.6.

Dada una suposición inicial de los parámetros de ajuste a, la *receta* recomendada por Marquardt es la siguiente:

- Calcular $\chi^2(a)$
- Dar un valor modesto a λ, por ejemplo $\lambda = 0{,}001$
- (†) Resolver el sistema de ecuaciones lineal 4.3.3 para δa y evaluar $\chi^2(a + \delta a)$
- Si $\chi^2(a + \delta a) \geq \chi^2(a)$ aumentar λ por un factor de 10 (u otro factor relevante) y volver a (†)
- Si $\chi^2(a + \delta a) < \chi^2(a)$ disminuir λ por un factor de 10 y actualizar la solución $a \leftarrow a + \delta a$ y volver a (†)

También es necesaria una condición de terminación, ya que al iterar hacia la solución convergente se pueden dar errores de redondeo. No obstante no es necesario llegar a tanta precisión puesto que el mínimo es en el mejor de los casos una estimación conjunto a de parámetros. Y por otro lado cualquier cambio en los parámetros que provoque que χ^2 varíe en una cantidad $\ll 1$ no tiene relevancia estadística.

Además tampoco es infrecuente que los parámetros oscilen entorno al mínimo cuando hay poca pendiente y topografía complicada. Esto ocurre porque el método de Marquardt generaliza el método de las ecuaciones normales (mínimos cuadrados) y por lo tanto sufre el mismo problema de degeneración en la proximidad del mínimo.

Hay otros motivos que pueden evitar que el método funcione de manera correcta, por ejemplo se puede dar el caso de que un pivote sea cero y esto provocaría el fallo de manera inmediata, pero esto es tan poco frecuente que no se tiene en consideración. Por otro lado si que es más habitual que el valor de este pivote sea pequeño, lo que provocaría que la corrección fuese muy grande y por tanto rechazada haciendo que el valor de λ aumentase. Para un valor de λ lo suficientemente elevado la matriz α' es definida positiva y haciendo improbable que tenga pivotes pequeños. Por este motivo el método tiende a evitar los pivotes pequeños pero esto tiene el inconveniente de hacer oscilar el descenso del gradiente en valles degenerados.

Estas consideraciones sugieren que en la práctica sea mejor dejar de iterar con un valor de error superior al mínimo ideal, después de que χ^2 disminuya una cantidad despreciable($\approx$ 0,001). Pero sobre todo no parar después de que χ^2 aumente más que de una manera trivial, esto sólo muestra que λ no se ha autoajustado de manera óptima.

Una vez se ha alcanzado un mínimo aceptable, se suele querer que $\lambda = 0$ y calcular la matriz

$$C \equiv \alpha^{-1} \tag{4.3.4}$$

que es la matriz de covarianza estimada de los errores estándares de los parámetros ajustados a.

4.4. Modelos de deformación y estimación de parámetros

Es necesario definir un modelo de deformación puesto que este define el tipo de transformación espacial que se deberá aplicar entre cualesquiera dos imágenes, I_1 e I_2. Este define una relación geométrica entre cada punto en cada una de las imágenes. La forma general de esta función de mapeo introducida por la deformación es

$$[x, y] = [X(u, v), Y(u, v)]$$

donde $[u, v]$ y $[x, y]$ son los pixels correspondientes en I_1 e I_2, respectivamente, y X e Y.

En el caso del alineamiento geométrico entre I_1 e I_2 se está interesado en en recuperar las funciones de mapeo inverso U y V que transforman I_2 "de vuelta" a I_1:

$$[u, v] = [U(x, y), V(x, y)]$$

En este sentido el proceso de alineamiento geométrico es similar a una corrección geométrica mediante la cual se intenta invertir la distorsión.

La elección del modelo de deformación debe estar relacionada con el proceso de captura de las imágenes, puesto que la complejidad del modelo determina la elección del algoritmo de alineamiento. Por ejemplo, si se sabe que las imágenes difieren en pequeñas traslaciones, sólo es necesaria una técnica simple de correlación. Sin embargo, si difieren en deformaciones elásticas y esto hará necesario que los puntos de correspondencia sean aportados por el usuario. Estos puntos, también conocidos como puntos de control o *fiducial points* o *tie points*, son suministrados por el usuario de manera dispersa e irregular. Puesto que la función de mapeo es conocida de manera precisa sólo en estos puntos, se utiliza un algoritmo de interpolación para datos dispersos para obtener una estimación suave de la función de mapeo en cualquier otro punto de la imagen.

La correspondencia geométrica es lograda determinando la función de mapeo que gobierna la relación entre todos los puntos entre un par de imágenes. Hay varios modelos de funciones de mapeo que son comunes en el ámbito del alineamiento geométrico. Entre ellos se encuentran:

- Transformaciones rígidas

  Son necesarios 3 parámetros para definirlas y son traslación y rotación. Se mantienen los ángulos entre las líneas rectas.

- Transformaciones afines

  Son necesarios 6 parámetros para definirlas y son traslación, rotación, escala y cizalla. Si dos rectas son paralelas lo seguirán siendo en el espacio transformado.

- Transformaciones perspectivas

  Son necesarios 8 parámetros para definirlas. Se trata de un superconjunto de la transformación afín pero es una transformación no lineal que no garantiza que dos rectas paralelas sigan siéndolo en el espacio transformado. La matriz que representa esta transformación se suele llamar homografía.

- Transformaciones locales no rígidas

  En este caso no existe número de parámetros fijo puesto que se trata de un modelo de deformación que puede ser diferente para cada punto de la imagen. Estos modelos son necesarios cuando se trata de obtener una correspondencia densa para tareas como la fotogrametría, reconstrucción de información 3D a partir de 2 o más imágenes.

En el caso que se contempla se van a alinear imágenes entre las que existe una única transformación perspectiva. Se puede suponer que sólo hace falta una transformación perspectiva para alinearlas puesto que las imágenes estarán tomadas desde una distancia (z) mucho mayor que las coordenadas (x,y) y por tanto se asemejarán a imágenes planas. Al tratarse de imágenes planas se garantiza que sólo sea necesaria una única transformación perspectiva. No obstante habría que tener en cuenta aberraciones causadas por el sensor o cualquier otro tipo de deformación que complicaría el modelo, pero se puede suponer que estas deformaciones no son relevantes puesto que no se requiere una correspondencia densa.

Por tanto la función de mapeo para una transformación perspectiva será la siguiente

$$u = \frac{a_0 x + a_1 y + a_2}{a_6 x + a_7 y + 1}$$
$$v = \frac{a_3 x + a_4 y + a_5}{a_6 x + a_7 y + 1}$$

Esto define la transformación, en coordenadas homogéneas, $T\{I\}$ como

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ a_3 & a_4 & a_5 \\ a_6 & a_7 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{u'}{w'} \\ \frac{v'}{w'} \\ \frac{w'}{w'} \end{bmatrix}$$

Mientras que para una transformación afín $a_6 = a_7 = 0$ dando como resultado

$$u = a_0x + a_1y + a_2$$
$$v = a_3x + a_4y + a_5$$

y su transformación correspondiente $T_A\{I\}$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ a_3 & a_4 & a_5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Como criterio para medir la similitud entre las imágenes se utilizará la suma de las diferencias al cuadrado o *sum of squared differences (SSD)* que en un dominio bidimensional se define de la siguiente manera

$$\begin{aligned} \chi^2(a) &= \iint\limits_{u\subset\mathbb{R}^2} [I_1(u) - I_2'(u)]^2 \,\mathrm{d}u \\ &= \iint\limits_{u\subset\mathbb{R}^2} [I_1(u) - T\{I_2(x)\}]^2 \,\mathrm{d}u \\ &= \|I_1(u) - T\{I_2(x)\}\|^2 \end{aligned}$$

donde T es una transformación geométrica aplicada a la imagen I_2 para mapearla desde su sistema de coordenadas (x,y) al sistema de coordenadas (u,v) de I_1

El caso anterior trata un modelo de transformación continuo pero en el caso en el que se trabaja utiliza datos discretos y por tanto un modelo discretizado para dichos datos es el siguiente:

$$\begin{aligned} \chi^2(a) &= \sum_{i=1}^{N} [I_1(u_i, v_i) - I_2(u_i, v_i)]^2 \\ &= \sum_{i=1}^{N} [I_1(u_i, v_i) - T\{I_2(x_i, y_i)\}]^2 \\ &= \sum_{i=1}^{N} \left[I_1(u_i, v_i) - I_2\left(\frac{a_0x_i + a_1y_i + a_2}{a_6x_i + a_7y_i + 1}, \frac{a_3x_i + a_4y_i + a_5}{a_6x_i + a_7y_i + 1} \right) \right]^2 \end{aligned}$$

que en el caso de tratarse de un modelo de transformación afín, T_A, la discretización del modelo resultará de la siguiente forma

$$\begin{aligned}\chi^2(a) &= \sum_{i=1}^{N}[I_1(u_i,v_i) - I_2(u_i,v_i)]^2 \\ &= \sum_{i=1}^{N}[I_1(u_i,v_i) - T_A\{I_2(x_i,y_i)\}]^2 \\ &= \sum_{i=1}^{N}\left[I_1(u_i,v_i) - I_2\left(a_0x_i + a_1y_i + a_2, a_3x_i + a_4y_i + a_5\right)\right]^2\end{aligned}$$

4.5. Newton-Raphson

El mínimo de una función se da en ellos puntos donde la primera derivada de la función es cero. Este se describe para el caso afín, puesto que ha sido usado en dicho caso. Por tanto hay que resolver para:

$$B_k(a) = \frac{\partial\chi^2(a)}{\partial a_k} = -2\sum_{i=1}^{N}\left([I_1(u_i) - I_2'(u_i)]\frac{\partial I_2'(u_i)}{\partial a_k}\right) = 0$$

donde $I_2'(u_i) = T_A\{I_2(x_i)\}$ y $k = 0, 1, \ldots, 5$

Esto supone un sistema de seis ecuaciones no lineales

$$B_k = 0 \quad k = 0, 1, \ldots, 5$$

La parte lineal de la serie de Taylor es la siguiente

$$B_k(a + \Delta a) \approx B_k(a) + \sum_{l=0}^{5}\frac{\partial B_k(a)}{\partial a_l}\Delta a_l$$

Seis incógnitas y seis ecuaciones lineales dan como resultado

$$B(a + \Delta a) = B(a) + H(a)\Delta a$$

Cuando $(a + \Delta a)$ sea la solución, entonces $B(a + \Delta a) = 0$ y por tanto

$$H(a)\Delta a = -B(a)$$

donde $H(a)$ es la hessiana, una matriz 6x6 de derivadas de segundo orden simétrica, $h_{kl} = h_{lk}$

$$\begin{aligned}h_{kl} &= \frac{\partial B_k(a)}{\partial a_l} \\ &= \frac{\partial^2\chi^2(a)}{\partial a_k\partial a_l} \\ &= 2\sum_{i=1}^{N}\left[\frac{\partial I_2'(u_i)}{\partial a_k}\frac{\partial I_2'(u_i)}{\partial a_l} - [I_1(u_i) - I_2'(u_i)]\frac{\partial^2 I_2'(u_i)}{\partial a_k\partial a_l}\right]\end{aligned}$$

Este sistema de ecuaciones se puede resolver para los seis valores de Δa mediante el método de Gauss, este método es adecuado porque converge de manera rápida. Sus inconvenientes son la dificultad de encontrar buenos parámetros iniciales y la convergencia lenta u oscilaciones si la función no está bien balanceada.

Para evitar estas oscilaciones, se puede realizar una aproximación de la misma de tal manera que se desprecien los términos relacionados con la segunda derivada.

$$h_{kl} = \frac{\partial B_k(a)}{\partial a_l} = \frac{\partial^2 \chi^2(a)}{\partial a_k \partial a_l} = 2 \sum_{i=1}^{N} \left[\frac{\partial I_2'(u_i)}{\partial a_k} \frac{\partial I_2'(u_i)}{\partial a_l} \right]$$

4.6. Descenso del gradiente

En cualquier minimización iterativa, la iteración (i+1) está relacionada con la iteración i por la ecuación

$$a_{i+1} = a_i + \Delta a_i$$

Se denomina $\{a_{i+1}\}$ una secuencia de descenso si $\chi^2(a_{a_{i+1}} < \chi^2(a))$.

Una dirección que con seguridad produce un descenso es la dirección del gradiente negativo. Por tanto

$$a_{i+1} = a_i - C \nabla \chi^2(a)$$

donde C controla el tamaño del paso a lo largo de cada parámetro en

$$\begin{aligned} \Delta a_i &= a_{i+1} - a_i \\ &= -C \nabla \chi^2(a) \\ &= -CB(a) \end{aligned}$$

La ventaja de este método es que siempre converge hasta un mínimo local. Por otro lado su desventaja es su convergencia lenta hacia el mismo.

4.7. LMA(Newton-Raphson y Descenso del gradiente)

Este método se basa en la combinación del método de Newton-Raphson y el método del descenso del gradiente.

De Newton-Raphson se tiene

$$H(a) \Delta a = -B(a)$$

y del descenso del gradiente se tiene

$$\Delta a = -CB(a)$$

Si se define

$$C_k = \frac{1}{\lambda h_{kk}}$$

entonces se tiene

$$\lambda h_{kk} \Delta a = -B(a)$$

Con esto, Marquardt probó que el siguiente método híbrido minimiza $\chi^2(a)$ de manera más robusta que cualquiera de ambos por separado.

$$[H(a) + \lambda I]\Delta a = -B(a)$$

donde el parámetro $\lambda \geq 0$ controla el grado en el que el método de comporta como el de Newton-Raphson o como el del descenso del gradiente. En el caso de que la actualización anterior haya conseguido reducir $\chi^2(a)$, entonces λ es reducido para realizar la actualización del método de Newton-Raphson. En el caso contrario, aumenta $\chi^2(a)$, se incrementa λ para realizar la actualización del descenso del gradiente. Por tanto, el método resultante , conocido como Algoritmo Levenberg-Marquard (LMA), es el que se describe a continuación. Este algoritmo requiere dos umbrales T_1 y T_2 para especificar las condiciones de terminación. El algoritmo terminará cuando el cambio de $\chi^2(a)$ caiga por debajo de T_1 o bien cuando λ supere el umbral T_2

```
begin
    Inicializar los parámetros como la matriz identidad
     (la transformación obtenida también es la identidad)
    Inicializar λ con un valor dado, por ejemplo λ = 0,001
    mientras_que ( |Δχ²(a)| > T1  ||  λ < T2 )
        Calcular la Hessiana
        Calcular el vector B
        Resolver el sistema lineal (H(a) + λI)Δa = −B para Δa
        Evaluar χ²(a + Δa)
        si χ²(a + Δa) < χ²(a) entonces
            λ ← λ/10
            a ← a + Δa
        si_no
            λ ← λ * 10
        fin_si
    fin_mientras_que
```

4.8. Estimación perspectiva mediante aproximaciones afines

En este caso se extiende la alineación geométrica afín de tal manera que pueda tratar con transformaciones perspectivas. Para ello hace falta estimar los ocho parámetros que definen dicha transformación perspectiva. Se puede proceder de manera directa como en el caso anterior, el problema es que si la posición inicial no es muy similar a la solución es muy fácil quedarse estancado en un mínimo local. Esto es mucho más grave con las transformaciones perspectivas porque se trata de transformaciones no lineales. Para tratar de evitar este problema se intenta aproximar la transformación perspectiva por transformaciones

afines en distintos trozos de la imagen. De esta manera se utiliza la estimación afín, que es más robusta, y se estima luego la transformación perspectiva en función de estos resultados.

Para poder hacer esto, se hace uso de una aproximación afín local entorno a un punto mediante la serie de Taylor de primer orden, y se define como

$$\begin{aligned} u &= U(x,y) \\ &= U(x_0,y_0) + \frac{\partial U(x_0,y_0)}{\partial x}(x-x_0) + \frac{\partial U(x_0,y_0)}{\partial y}(y-y_0) \\ &= A_0x + A_1y + A_2 \end{aligned} \tag{4.8.1}$$

$$\begin{aligned} v &= V(x,y) \\ &= V(x_0,y_0) + \frac{\partial V(x_0,y_0)}{\partial x}(x-x_0) + \frac{\partial V(x_0,y_0)}{\partial y}(y-y_0) \\ &= A_3x + A_4y + A_5 \end{aligned} \tag{4.8.2}$$

donde

$$\begin{array}{lll} A_0 = \frac{\partial U(x_0,y_0)}{\partial x} & A_1 = \frac{\partial U(x_0,y_0)}{\partial y} & A_2 = U(x_0,x_0) - A_0x_0 - A_1y_0 \\ A_3 = \frac{\partial V(x_0,y_0)}{\partial x} & A_4 = \frac{\partial V(x_0,y_0)}{\partial y} & A_5 = V(x_0,x_0) - A_3x_0 - A_4y_0 \end{array}$$

A continuación se va a mostrar como es el proceso para determinar la transformación afín entre dos tiles y su uso para inferir los parámetros de la transformación perspectiva. Se resume la transformación afín con la transformación de las cuatro esquinas, de esta manera se reduce el número de puntos a tener en cuenta y se evitan errores en los datos.

Aunque la transformación perspectiva se puede estimar de manera directa una vez conocida la correspondencia entre las parejas cada una de las cuatro esquinas, en este caso se trata de encontrar la mejor transformación afín que aproxima este mapeo.

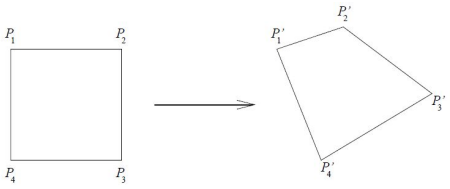


Figura 16: Mapeo de 4 esquinas

Dadas las cuatro esquinas de una tile en la imagen I_2 y sus correspondientes en la de referencia I_1, se puede resolver el mejor ajuste de la transformación afín usando un ajuste por mínimos cuadrados. Por tanto se puede definir el mapeo de la siguiente manera

$$\begin{aligned} u &= a_0x + a_1y + a_2 \\ v &= a_3x + a_4y + a_5 \end{aligned}$$

Se resuelve para los parámetros afines mediante la minimización de la expresión de χ^2 mostrada a continuación

$$\chi^2(a) = \sum_{i=1}^{4} \left[(u_i - a_0 x_i - a_1 y_i - a_2)^2 + (v_i - a_3 x_i - a4 y_i - a_5)^2 \right]$$

Se pueden relacionar estas correspondencias de la forma $U = WA$

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ x_4 & y_4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \\ 0 & 0 & 0 & x_4 & y_4 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}$$

y para terminar de inferir los seis parámetros afines se calcula la solución con el método de la pseudoinversa

$$A = (W^T W)^{-1} W^T U$$

La estimación anterior es para una pareja de tiles, pero lo que se busca es inferir los parámetros afines para cada una de las N parejas en I_1 e I_2 y aplicársela a ellas en el centro de cada tile. Esto establece la correspondencia de N puntos entre I_1 e I_2, haciendo que los ocho parámetros de la transformación perspectiva puedan ser inferidos resolviendo el siguiente sistema de ecuaciones

$$\begin{aligned} u &= a_0 x + a_1 y + a_2 - a_6 ux - a_7 uy \\ v &= a_3 x + a_4 y + a_5 - a_6 vx - a_7 vy \end{aligned}$$

que define el siguiente sistema de ecuaciones sobredeterminado

$$\begin{bmatrix} u_0^i \\ \vdots \\ v_0^i \\ \vdots \end{bmatrix}_{2N \times 1} = \begin{bmatrix} x_0^i & y_0^i & 1 & 0 & 0 & 0 & -u_0^i x_0^i & -u_0^i y_0^i \\ & & & & \vdots & & & \\ 0 & 0 & 0 & x_0^i & y_0^i & 1 & -v_0^i x_0^i & -v_0^i y_0^i \\ & & & & \vdots & & & \end{bmatrix}_{2N \times 8} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix}_{8 \times 1} \tag{4.8.3}$$

donde (u_0^i, v_0^i) y (x_0^i, y_0^i) representan los centros de la tile i en I_1 e I_2 para $1 \leq i \leq N$.

De manera similar al caso afín, se calcula la solución utilizando el método de la pseudoinversa para calcular los ocho parámetros de la transformación perspectiva.

$$A = (W^T W)^{-1} W^T U$$

Estos resultados se pueden mejorar si se añaden restricciones adicionales a la estimación de los mínimos cuadrados. La restricción que se puede imponer es la correspondencia entre los centros de las tiles.

De las ecuaciones 4.8.1 y 4.8.2 se tenía que la aproximación a la transformación afín es de la forma

$$\begin{aligned} u &= A_0x + A_1y + A_2 \\ v &= A_3x + A_4y + A_5 \end{aligned}$$

donde

$$\begin{aligned} A_0 &= \frac{\partial U(x_0,y_0)}{\partial x} = \frac{a_0(a_6x_0+a_7y_0+1) - a_6(a_0x_0+a_1y_0+a_2)}{(a_6x_0+a_7y_0+1)^2} \\ &= \frac{a_0 - a_6u_0}{a_6x_0+a_7y_0+1} = a_0 - a_6(A_0x_0+u_0) - a_7A_0y_0 \\ A_1 &= \frac{\partial U(x_0,y_0)}{\partial y} = \frac{a_1(a_6x_0+a_7y_0+1) - a_7(a_0x_0+a_1y_0+a_2)}{(a_6x_0+a_7y_0+1)^2} \\ &= \frac{a_1 - a_7u_0}{a_6x_0+a_7y_0+1} = a_1 - a_6A_1x_0 - a_7(A_1y_0+u_0) \\ A_2 &= U(x_0,y_0) - A_0x_0 - A_1y_0 = \frac{a_0x_0+a_1y_0+a_2}{a_6x_0+a_7y_0+1} - A_0x_0 - A_1y_0 \\ &= a_0x_0 + a_1y_0 + a_2 - a_6u_0x_0 - a_7u_0y_0 - (u_0 - A_2) \\ A_3 &= \frac{\partial V(x_0,y_0)}{\partial x} = \frac{a_3(a_6x_0+a_7y_0+1) - a_6(a_3x_0+a_4y_0+a_5)}{(a_6x_0+a_7y_0+1)^2} \\ &= \frac{a_3 - a_6v_0}{a_6x_0+a_7y_0+1} = a_3 - a_6(A_3x_0+v_0) - a_7A_3y_0 \\ A_4 &= \frac{\partial V(x_0,y_0)}{\partial y} = \frac{a_4(a_6x_0+a_7y_0+1) - a_7(a_3x_0+a_4y_0+a_5)}{(a_6x_0+a_7y_0+1)^2} \\ &= \frac{a_4 - a_7v_0}{a_6x_0+a_7y_0+1} = a_4 - a_6A_4x_0 - a_7(A_4y_0+v_0) \\ A_5 &= V(x_0,y_0) - A_3x_0 - A_4y_0 = \frac{a_3x_0+a_4y_0+a_5}{a_6x_0+a_7y_0+1} - A_3x_0 - A_4y_0 \\ &= a_3x_0 + a_4y_0 + a_5 - a_6v_0x_0 - a_7v_0y_0 - (v_0 - A_5) \end{aligned}$$

Hay que tener en cuenta que los términos de A_2 y A_5 generan ecuaciones de la forma $u_0 = a_0x_0+a_1y_0+a_2-a_6u_0x_0-a_7u_0y_0$ y $v_0 = a_3x_0+a_4y_0+a_5-a_6v_0x_0-a_7v_0y_0$ respectivamente. Estas son las mismas correspondencias definidas en la ecuación 4.8.3. De forma compacta se puede expresar las ecuaciones anteriores para obtener la relación de los parámetros afines en cada una de las N tiles con

los parámetros desconocidos de la transformación perspectiva buscada.

$$\begin{bmatrix} A_0^i \\ \vdots \\ A_1^i \\ \vdots \\ u_0^i \\ \vdots \\ A_3^i \\ \vdots \\ A_4^i \\ \vdots \\ v_0^i \\ \vdots \end{bmatrix}_{6N\times 1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -(A_0^i x_0^i + u_0^i) & -A_0^i y_0^i \\ & & & & & \vdots & & \\ 0 & 1 & 0 & 0 & 0 & 0 & -A_1^i x_0^i & -(A_1^i y_0^i + u_0^i) \\ & & & & & \vdots & & \\ x_0^i & y_0^i & 1 & 0 & 0 & 0 & -u_0^i x_0^i & -u_0^i y_0^i \\ & & & & & \vdots & & \\ 0 & 0 & 0 & 1 & 0 & 0 & -(A_3^i x_0^i + v_0^i) & -A_3^i y_0^i \\ & & & & & \vdots & & \\ 0 & 0 & 0 & 0 & 1 & 0 & -A_4^i x_0^i & -(A_4^i y_0^i + v_0^i) \\ & & & & & \vdots & & \\ 0 & 0 & 0 & x_0^i & y_0^i & 1 & -v_0^i x_0^i & -v_0^i y_0^i \\ & & & & & \vdots & & \end{bmatrix}_{6N\times 8} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix}_{8\times 1} \tag{4.8.4}$$

donde (u_0^i, v_0^i) y (x_0^i, y_0^i) representan los centros de la tile i en I_1 e I_2 para $1 \leq i \leq N$. Hay que tener en cuenta que la ecuación 4.8.4 es un superconjunto de la ecuación 4.8.3. Y de la misma manera que en el caso anterior, se resuelve utilizando el método de la pseudoinversa. Otra cosa a tener en cuenta si se utiliza la ecuación 4.8.4 es que la solución de la misma no es estable si se calcula en las coordenadas de los pixels de manera directa. Por lo tanto hay que realizar una normalización de las coordenadas de los pixels a un dominio de coordenadas $[0, 1]$

Con todo esto, se va a ver como es el algoritmo que resuelve esta aproximación a la transformación perspectiva.
Considérese una imagen a alinear I_2 y una imagen de referencia I_1, ambas divididas en tiles de forma regular y con las mismas dimensiones para poder calcular χ^2. Para cada tile T_2^i se realiza el alineamiento geométrico afín usando el método Levenberg-Marquardt para encontrar la tile más similar de T_1^i. Esto ofrece una colección de parámetros afines y centros de tiles, con los que se estimarán los parámetros de la transformación perspectiva.

```
Particionar I_2 en N tiles: T_2^i, para 1 ≤ i ≤ N
i ← 1
mientras_que ( i < N )
    Seleccionar la tile T_2^i
    para todas las posiciones (x,y) en I_1
        Seleccionar la tile T_1^i
        Alinear T_2^i con T_1^i usando LMA
        si χ^2(a) es mínimo entonces
            x^i ← x
            y^i ← y
            a^i ← a
        fin_si
    fin_para
    i ← i + 1
fin_mientras_que
Resolver la ecuación 4.8.3 o 4.8.4 para los ocho parámetros
de la transformación perspectiva (mínimos cuadrados lineales)
```

4.9. Transformación afín inversa

Método eficiente del cálculo de la inversa de una matriz 3x3

$$a^{-1} = \begin{bmatrix} a_0 & a_1 & a_2 \\ a_3 & a_4 & a_5 \\ a_6 & a_7 & a_8 \end{bmatrix}^{-1} = \frac{1}{det(a)} \begin{bmatrix} A_0 & A_1 & A_2 \\ A_3 & A_4 & A_5 \\ A_6 & A_7 & A_8 \end{bmatrix}^T = \frac{1}{det(a)} \begin{bmatrix} A_0 & A_3 & A_6 \\ A_1 & A_4 & A_7 \\ A_2 & A_5 & A_8 \end{bmatrix}$$

$$\begin{array}{lll} A_0 = (a_4a_8 - a_5a_7) & A_3 = (a_2a_7 - a_1a_8) & A_6 = (a_1a_5 - a_2a_4) \\ A_1 = (a_5a_6 - a_3a_8) & A_4 = (a_0a_8 - a_2a_6) & A_7 = (a_2a_3 - a_0a_5) \\ A_2 = (a_3a_7 - a_4a_6) & A_5 = (a_6a_1 - a_0a_7) & A_8 = (a_0a_4 - a_1a_3) \end{array}$$
$$det(a) = a_0A_0 + a_1A_1 + a_2A_2$$

Transformación afín $\Leftrightarrow a_6 = a_7 = 0$ y $a_8 = 1$

$$a^{-1} = \begin{bmatrix} a_0 & a_1 & a_2 \\ a_3 & a_4 & a_5 \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \frac{1}{a_0a_4 - a_1a_3} \begin{bmatrix} a_4 & -a_1 & a_1a_5 - a_2a_4 \\ -a_3 & a_0 & a_2a_3 - a_0a_5 \\ 0 & 0 & a_0a_4 - a_1a_3 \end{bmatrix}$$

La siguiente expresión es la forma compacta de expresar las transformaciones afines porque cuando hace uso de ellas se añade la última fila, que es siempre el mismo vector constante [0 0 1]

$$a^{-1} = \frac{1}{a_0a_4 - a_1a_3} \begin{bmatrix} a_4 & -a_1 & a_1a_5 - a_2a_4 \\ -a_3 & a_0 & a_2a_3 - a_0a_5 \end{bmatrix}$$

A continuación se muestra el cálculo de las derivadas de primer orden, que forman el vector B, en función de cada parámetro involucrado en el proceso de la transformación afín.

$$\frac{\partial a^{-1}}{\partial a_0} = \frac{1}{(a_0a_4 - a_1a_3)^2} \begin{bmatrix} -a_4^2 & a_1a_4 & a_2a_4^2 - a_1a_4a_5 \\ a_3a_4 & -a_1a_3 & a_1a_3 - a_2a_3 \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_1} = \frac{1}{(a_0a_4 - a_1a_3)^2} \begin{bmatrix} a_4a_3 & -a_0a_4 & a_0a_4a_5 - a_2a_3a_4 \\ -a_3^2 & a_0a_3 & a_2a_3^2 - a_0a_3a_5 \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_2} = \frac{1}{(a_0a_4 - a_1a_3)^2} \begin{bmatrix} 0 & 0 & -a_4 \\ 0 & 0 & a_3 \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_3} = \frac{1}{(a_0a_4 - a_1a_3)^2} \begin{bmatrix} a_1a_4 & -a_1^2 & a_1^2a_5 - a_1a_2a_5 \\ -a_0a_4 & a_0a_1 & a_0a_2a_4 - a_0a_1a_5 \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_4} = \frac{1}{(a_0a_4 - a_1a_3)^2} \begin{bmatrix} -a_1a_3 & a_0a_1 & a_1a_2a_3 - a_0a_1a_5 \\ a_0a_3 & a_0^2 & a_0^2a_5 - a_0a_2a_3 \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_5} = \frac{1}{(a_0a_4 - a_1a_3)^2} \begin{bmatrix} 0 & 0 & a_1 \\ 0 & 0 & -a_0 \end{bmatrix}$$

A continuación se muestra el cálculo de las derivadas de segundo orden, que forman la matriz simétrica H, en función de cada parámetro involucrado en el proceso de la transformación afín.

$k = \frac{1}{det(a)} = \frac{1}{a_0 a_4 - a_1 a_3}$

$$\frac{\partial a^{-1}}{\partial a_0 \partial a_0} = k^3 \begin{bmatrix} 2a_4^3 & -2a_1 a_4^2 & 2a_4(a_1 a_5 - a_2 a_4) \\ -2a_3 a_4^2 & 2a_1 a_3 a_4 & -2a_3 a_4(a_1 a_5 - a_2 a_4) \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_0 \partial a_1} = k^3 \begin{bmatrix} -2a_3 a_4^2 & a_4(a_0 a_4 + a_1 a_3) & (a_0 a_5 - 2a_2 a_3)a_4^2 + a_1 a_3 a_4 a_5 \\ 2a_3^2 a_4 & -a_3(a_0 a_4 + a_1 a_3) & a_3 a_4(a_0 a_5 - 2a_2 a_3) + a_1 a_3^2 a_5 \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_0 \partial a_2} = k^2 \begin{bmatrix} 0 & 0 & a_4^2 \\ 0 & 0 & -a_3 a_4 \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_0 \partial a_3} = k^3 \begin{bmatrix} -2a_1 a_4^2 & 2a_1^2 a_4 & -2a_1 a_4(a_1 a_5 - a_2 a_4) \\ a_0 a_4^2 + a_1 a_3 a_4 & -a_0 a_1 a_4 - a_1^2 a_3 & (a_0 a_4 + a_1 a_3)(a_1 a_5 - a_2 a_4) \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_0 \partial a_4} = k^3 \begin{bmatrix} 2a_1 a_3 a_4 & -a_0 a_1 a_4 - a_1^2 a_3 & a_1^2 a_3 a_5 + a_1 a_4(a_0 a_5 - a_2 a_3) \\ -a_0 a_3 a_4 - a_1 a_3^2 & 2a_0 a_1 a_3 & -a_0 a_3(2a_1 a_5 + a_2 a_4) - a_1 a_2 a_3^2 \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_0 \partial a_5} = k^2 \begin{bmatrix} 0 & 0 & -a_1 a_4 \\ 0 & 0 & a_1 a_3 \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_1 \partial a_1} = k^3 \begin{bmatrix} 2a_3^2 a_4 & -2a_0 a_3 a_4 & 2a_3 a_4(a_0 a_5 - a_2 a_3) \\ -2a_3^3 & 2a_0 a_3^2 & -2a_3^2(a_0 a_5 - a_2 a_3) \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_1 \partial a_2} = k^2 \begin{bmatrix} 0 & 0 & -a_3 a_4 \\ 0 & 0 & a_3^2 \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_1 \partial a_3} = k^3 \begin{bmatrix} a_0 a_4^2 + a_1 a_3 a_4 & -2a_0 a_1 a_4 & a_1 a_4(2a_0 a_5 - a_2 a_3) - a_0 a_2 a_4^2 \\ -2a_0 a_3 a_4 & a_0^2 a_4 + a_0 a_1 a_3 & -a_0^2 a_4 a_5 + a_0 a_3(a_1 a_5 - 2a_2 a_4) \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_1 \partial a_4} = k^3 \begin{bmatrix} -a_0 a_3 a_4 - a_1 a_3^2 & a_0^2 a_4 + a_0 a_1 a_3 & (a_0 a_4 + a_1 a_3)(a_2 a_3 - a_0 a_5) \\ 2a_0 a_3^2 & -2a_0^2 a_3 & 2a_0 a_3(a_0 a_5 - a_2 a_3) \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_1 \partial a_5} = k^2 \begin{bmatrix} 0 & 0 & a_0 a_4 \\ 0 & 0 & -a_0 a_3 \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_2 \partial a_2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_2 \partial a_3} = k^2 \begin{bmatrix} 0 & 0 & -a_1 a_4 \\ 0 & 0 & a_0 a_4 \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_2 \partial a_4} = k^2 \begin{bmatrix} 0 & 0 & a_1 a_3 \\ 0 & 0 & -a_0 a_3 \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_2 \partial a_5} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_3 \partial a_3} = k^3 \begin{bmatrix} 2a_1^2 a_4 & -2a_1^3 & 2a_1^2(a_1 a_5 - a_2 a_4) \\ -2a_0 a_1 a_4 & 2a_0 a_1^2 & -2a_0 a_1 (a_1 a_5 - a_2 a_4) \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_3 \partial a_4} = k^3 \begin{bmatrix} -a_1(a_0 a_4 + a_1 a_3) & 2a_0 a_1^2 & -a_1^2(2a_0 a_5 - a_2 a_3) - a_0 a_1 a_2 a_4 \\ a_0(a_0 a_4 + a_1 a_3) & 2a_0^2 a_1 & a_0 a_1 (2a_0 a_5 - a_2 a_3) - a_0^2 a_2 a_4 \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_3 \partial a_5} = k^2 \begin{bmatrix} 0 & 0 & a_1^2 \\ 0 & 0 & -a_0 a_1 \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_4 \partial a_4} = k^3 \begin{bmatrix} 2a_0 a_1 a_3 & -2a_0^2 a_1 & 2a_0 a_1 (a_0 a_5 - a_2 a_3) \\ -2a_0^2 a_3 & 2a_0^3 & -2a_0^2 (a_0 a_5 - a_2 a_3) \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_4 \partial a_5} = k^2 \begin{bmatrix} 0 & 0 & -a_0 a_1 \\ 0 & 0 & a_0^2 \end{bmatrix}$$

$$\frac{\partial a^{-1}}{\partial a_5 \partial a_5} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

4.10. Transformación singular

En el método LMA se trabaja con la función $\chi^2(a)$, como medida del error entre las dos imágenes, y sus derivadas de primer y segundo orden para resolver cuanto hay que variar los parámetros de transformación para acercarse al mínimo de la función de error. Esta función de error implica realizar una transformación afín o perspectiva para poder mapear una imagen en el espacio de la otra y así poder medir el error entre ambas de manera directa.

Dicha transformación consiste en multiplicar las coordenadas de cada pixel de la imagen a transformar por la matriz de transformación que contiene los 6 u 8 parámetros si se desea realizar una transformación afín o perspectiva y de esta manera obtener la posición del pixel en el espacio transformado, aunque en la práctica se calcula con la inversa por ser más eficiente.

$$a_{perspectiva} = \begin{bmatrix} a_0 & a_1 & a_2 \\ a_3 & a_4 & a_5 \\ a_6 & a_7 & 1 \end{bmatrix} \quad a_{afín} = \begin{bmatrix} a_0 & a_1 & a_2 \\ a_3 & a_4 & a_5 \\ 0 & 0 & 1 \end{bmatrix}$$

Derivar la función $\chi^2(a)$ implica calcular derivadas de la transformación en función de un parámetro en el caso de la primera derivada (vector B) y en función de dos parámetros en el caso de la segunda derivada (matriz Hessiana). Al calcular estas derivadas se obtiene una nueva transformación y esta es no lineal puesto que se trata de una proyección de la imagen.

Las funciones tanto de OpenCV (*warpPerspective()* y *warpAffine()*) como de Matlab (*imtransform()*) no están concebidas para tratar con este tipo de transformaciones proyectivas que son más comunes en el ámbito de los gráficos por computador puesto que en este ámbito si que es necesario realizar estas proyecciones de un espacio 3D a un espacio 2D para obtener la vista de una cámara. No obstante las funciones con las que se ha trabajado no lo soportan y devuelven un resultado técnicamente correcto pero incorrecto en el ámbito en el que se está trabajando. Esto es a causa de la forma en que trabajan estas funciones ya que para hacer su trabajo de una manera más eficiente trabajan con la inversa de la matriz de transformación y en este caso esta matriz al ser singular no tiene inversa. Visto de otra manera el problema que tienen estas funciones es la interpretación de la proyección, que en el caso de estas funciones es "encoger" la imagen hasta un vector o punto con valor nulo y por lo tanto esto no aporta ninguna información.

Se ha tratado de solucionar este inconveniente en la función *imtransform* mediante el uso de la función *maketform* con la que se pueden generar transformaciones arbitrarias que luego pueden ser interpretadas por *imtransform*. Esta función trabaja con la inversa de la transformación y por lo tanto ha sido necesario implementar una función de transformación inversa general y una función de transformación inversa para cada una de las derivadas de primer orden y de segundo orden, en función de los 8 parámetros de una transformación perspectiva.

A continuación se detalla la implementación de una de estas funciones para que sirva de ejemplo de lo que es necesario realizar para obtener la transformación deseada. Este caso concreto muestra la implementación de la función de transformación inversa de la derivada segunda respecto de los parámetros a_2 y a_7

```
function U = inv_fcn_da27( x , t )
a1 = t.tdata(1); a2 = t.tdata(2); a3 = t.tdata(3); a4 = t
    .tdata(4); a5 = t.tdata(5); a6 = t.tdata(6); a7 = t.
    tdata(7); a8 = t.tdata(8);
% projective
Ux = (-((a5 - a6*a8)*(a1*a5*a6 + a2*a4*a6 - a1*a6^2*a8 -
    a2*a6^2*a7) + a3*(a5 - a6*a8)*(a4*a6*a8 - 2*a4*a5 + a5
    *a6*a7))*x(:,1) + -((a5 - a6*a8)*(a1*a3*a5 - a3^2*a4*
    a8 - a3^2*a5*a7 + a1*a3*a6*a8) + a2*(a5 - a6*a8)*(a3*
    a4 - 2*a1*a6 + a3*a6*a7))*x(:,2) + -(2*(a1*a6 - a3*a4)
    *(a2*a6 - a3*a5)*(a5 - a6*a8)));
Uy = ((2*(a1*a6 - a3*a4)*(a4 - a6*a7)*(a5 - a6*a8))*x
    (:,1) + -((a1*a6 - a3*a4)*(a1*a5 + a2*a4 - a2*a6*a7 -
    a3*a5*a7) - a8*(a1*a6 - a3*a4)*(a1*a6 + a3*a4 - 2*a3*
    a6*a7))*x(:,2) + ((a1*a6 - a3*a4)*(a2*a4*a6 - a1*a6^2*
    a8 - a2*a6^2*a7 + a3*a4*a6*a8) + a5*(a1*a6 - a3*a4)*(
    a1*a6 - 2*a3*a4 + a3*a6*a7)));
Uw = (((a5 - a6*a8)*(a2*a4^2 - a1*a4*a5 + a3*a4^2*a8 - a1
    *a4*a6*a8) - a7*(a5 - a6*a8)*(a2*a4*a6 - 2*a1*a5*a6 +
    a3*a4*a5))*x(:,1) + (a1^2*(a5^2 - a5*a6*a8) - a1*(a4*(
    a2*(a5 - 2*a6*a8) + a3*a6*a8^2) + a3*a5^2*a7 + a2*a5*
    a6*a7 - 2*a3*a5*a6*a7*a8) + a4^2*(a3^2*a8^2 - a2*a3*a8
```

```
    ) + a4*(a2*(2*a3*a5*a7 - a3*a6*a7*a8) - a3^2*a5*a7*a8)
    )*x(:,2) + -((a1*a6 - a3*a4)*(a1*a5^2 - a2*a4*a5 + a3*
    a5^2*a7 - a2*a5*a6*a7) - a8*(a1*a6 - a3*a4)*(a1*a5*a6
    - 2*a2*a4*a6 + a3*a4*a5)));
UxNorm=Ux./Uw;
UyNorm=Uy./Uw;
U = [UxNorm,UyNorm];
end
```

En este ejemplo se observa como se aplica la inversa de la transformación (precalculada) a todos los puntos de la imagen. Esta función para efectuar la transformación de manera eficiente trabaja con las coordenadas x,y separadas en dos vectores $x_i = x(:,1)$ e $y_i = x(:,2)$. Otra cosa que se puede resaltar es que es necesario trabajar en coordenadas homogéneas (x,y,w), esto hace necesario calcular una coordenada adicional. Esta es necesario descartarla para poder volver a un espacio de dos dimensiones y por lo tanto es necesario normalizar el vector en función de ella.(Nota: en matlab un operador precedido por . simboliza que la operación se realiza componente a componente, por ejemplo en UxNorm=Ux./Uw; se realiza la división de cada componente de Ux por la correspondiente en Uw)

Como esta implementación no ha dado el resultado deseado y en el caso que se está tratando es necesario disponer de la información de las derivadas, puesto que se trata de la información básica con la que trabaja el método LMA, esto ha hecho necesario implementar estas funciones. Por un lado hay que decidir como realizar la transformación de una imagen de un espacio a otro, en este caso se ha decidido por simplicidad realizar la transformación directa. Por otro lado ha sido necesario interpretar la proyección, para ello se han estudiado ocho formas distintas de realizarla.

- Suma de la proyección sobre el primer elemento

  Esta interpretación es la más cercana a como se trata en las funciones de Matlab/OpenCv. De los 6 casos posibles que se dan al trabajar con las derivadas de la matriz afín, todos ellos al final se corresponden con una proyección a un vector o a un punto con un significado dependiente de cada caso. En lugar de realizar esta proyección sobre un espacio de tamaño infinitamente pequeño, lo que se hace en su lugar es proyectar sobre un vector de pixels o bien sobre un único pixel. Para efectuar esta proyección se suman todos los valores de los pixels que su proyección se corresponde con las coordenadas proyectadas. De esta manera se obtiene una suma de la intensidad acumulada según la proyección dada por el parámetro correspondiente en la primera fila, o la primera columna, o en el primer elemento. Mientras que el resto de la imagen resultante contendrá valores nulos.

- Suma de la proyección sobre un eje

  En este caso la interpretación es similar al caso anterior puesto que se realiza la proyección sumando los elementos correspondientes. La diferencia es que en este caso la proyección se realiza sobre un eje ideal infinitesimal y cómo se accede a la información contenida en él. Para ello en el caso de

tratarse de una proyección sobre un eje se ignora la coordenada perpendicular al eje, por no afectar, y se accede de manera directa mediante la otra coordenada a la posición del vector donde se ha almacenado la información. No obstante para trabajar de una manera más cómoda en matlab y no tener que implementar tantos casos particulares, se ha decidido replicar los valores en una matriz del tamaño de la imagen y por lo tanto de esta manera se accede de manera directa a la coordenada correspondiente sin tener en cuenta que caso particular está accediendo.

- Media de la proyección sobre el primer elemento y sobre un eje

  En este caso en lugar de obtener la suma de los elementos proyectados se ha obtenido su media. Se ha considerado que esta información podría ser más informativa para el método utilizado que la suma de la intensidad, al tratarse de un valor relativo en lugar de absoluto. Esto por otro lado también es beneficioso puesto que los valores resultantes quedan restringidos al rango de valores que tienen las imágenes y por lo tanto al realizar operaciones entre ellos los resultados tendrán menos errores numéricos.

- Normalización de la proyección sobre el primer elemento y sobre un eje

  En este caso se ha buscado otra manera diferente de obtener la información relativa al valor de cada elemento. Para ello se ha normalizado el vector que ha resultado de la proyección y de esta manera se ha obtenido un vector de pesos que ponderan cuanta importancia tiene cada componente, restringiéndose estos valores al rango [0,1].

- Normalización sobre el máximo de la proyección sobre el primer elemento y sobre un eje

  Este caso es muy similar al anterior, la diferencia radica en que en lugar de realizar la normalización del vector de manera habitual, lo que se hace es normalizar respecto a la componente de mayor tamaño y por tanto las proporciones obtenidas son distintas. Estas proporciones ofrecen una indicación de cuanta diferencia existe con el máximo valor de la imagen.

4.11. Diferencias Centrales

El cálculo de las derivadas se ha aproximado de manera numérica haciendo uso de las diferencias centrales. Esto se podría haber realizado de diferentes maneras pero se ha considerado que esta era la que más se ajustaba al problema. En concreto es más relevante en el caso de la derivada de primer orden puesto que de esta manera no se supone una dirección predominante de avance y se aproxima la derivada en torno centro. Hay que hacer notar que evaluar la función f(i,j) implica realizar la transformación afín/perspectiva de la imagen y calcular la derivada para cada pixel de la imagen y por lo tanto se ha procurado que el número de transformaciones realizadas fuese el mínimo necesario, reutilizándolas cuando ha sido posible.

$$f'(x) \approx \frac{f(x+\Delta) - f(x-\Delta)}{2\Delta}$$

$$f'_x(x,y) \approx \frac{f(x+\Delta,y) - f(x-\Delta,y)}{2\Delta}$$

$$f'_y(x,y) \approx \frac{f(x,y+\Delta) - f(x,y-\Delta)}{2\Delta}$$

$$f''_{xx}(x,y) \approx \frac{f(x+\Delta,y) - 2f(x,y) + f(x-\Delta,y)}{\Delta^2}$$

$$f''_{yy}(x,y) \approx \frac{f(x,y+\Delta) - 2f(x,y) + f(x,y-\Delta)}{\Delta^2}$$

$$f''_{xy}(x,y) \approx \frac{f(x+\Delta,y+\Delta) - f(x+\Delta,y-\Delta)}{4\Delta^2} + \frac{-f(x-\Delta,y+\Delta) + f(x-\Delta,y-\Delta)}{4\Delta^2}$$

4.12. Transformación Log-Polar

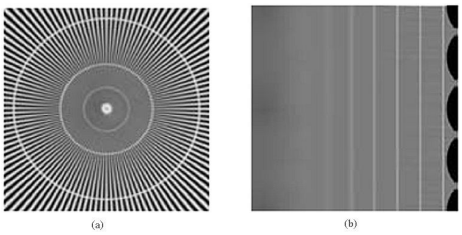
La transformación al dominio log-polar es una transformación no lineal y un muestreo no uniforme del dominio espacial. La no linealidad es debida a el mapeo polar, mientras que el muestreo no uniforme es el resultado del escalado logarítmico.

Considérese el sistema de coordenadas log-polar (r, θ), donde r representa la distancia radial desde el centro (x_c, y_c) en escala logarítmica y θ representa el ángulo. Por lo tanto, cualquier punto (x, y) puede ser representado en coordenadas log-polar de la siguiente manera

$$r = \log_{base}\left(\sqrt{(x-x_c)^2 + (y-y_c)^2}\right)$$

$$\theta = \tan^{-1}(\frac{y-y_c}{x-x_c})$$

Al aplicar una transformación polar a la imagen I se mapean las líneas radiales en el espacio cartesiano a líneas horizontales en el espacio de coordenadas polares. Se denominará I_{lp} a la imagen transformada, asumiendo que r y θ corresponden a los ejes horizontal y vertical respectivamente. Por lo tanto, la imagen I 17a será transformada en la imagen I_{lp} 17b después de una transformación log-polar.



(a) cartesianas (b) log-polar

Figura 17: Transformación a coordenadas Log-Polar

El origen de utilizar transformaciones al dominio log-polar proviene de que esta es la forma en la que muchos organismos biológicos utilizan en su retina. Esta forma captura la información con mayor densidad en el centro y disminuye conforme se aumenta la

distancia al mismo. Este muestreo no uniforme es simulado por la escala logarítmica. Aunque las coordenadas log-polar no son exactamente lo mismo, están aceptadas como modelo de representación de la retina de los primates.

La transformación al dominio log-polar tiene ventajas así como inconvenientes, pero a continuación se van a resaltar las dos ventajas más relevantes. Por un lado, las imágenes resultantes son invariantes a rotación y escala. Por otro lado, la variación espacial del muestreo no uniforme de la retina es la solución para reducir la cantidad de información que tiene que atravesar el nervio óptico, manteniendo una alta resolución en la fóvea y al mismo tiempo un amplio campo de visión.

Sean dos imágenes I_1 e I_2 y sus transformaciones correspondientes en el dominio log-polar I_{1p} e I_{2p}. Si I_2 es una versión rotada de I_1, entonces I_{2p} será una versión trasladada en el eje θ de I_{1p}. Aplicando una correlación cruzada de I_{1p} e I_{2p} se puede encontrar el *offset* entre ambas imágenes. Hay que resaltar que la técnica de correlación cruzada es usada de manera habitual para encontrar *offsets* de traslación entre dos imágenes y este no funciona bien en coordenadas cartesianas, en presencia de rotaciones o escalados. Sin embargo, se ha visto que en el espacio polar encontrar la componente de traslación entre I_{1p} e I_{2p} corresponde a encontrar la rotación entre I_1 e I_2.

Para determinar el cambio de escala el comportamiento es similar. Considérese una ampliación de I_1 de tal manera que sea cuatro veces mayor que I_2. Entonces, todos los puntos (x, y) de I_1 se corresponderán a los análogos $(4x, 4y)$ en I_2. Para determinar el factor de escala se hace uso de las propiedades de los logaritmos, puesto que en el espacio logarítmico $(x, y) \rightarrow (\log x, \log y)$, y $(4x, 4y) \rightarrow (\log 4x, \log 4y) \rightarrow (\log x + \log 4, \log y + \log 4)$. De este modo, se observa de manera explícita como en el espacio logarítmico la introducción de un cambio de escala se corresponde con una traslación en el eje logarítmico (r o $logr$, según la notación empleada) de la imagen transformada.

4.13. Transformada de Fourier-Mellin

El método de alineamiento geométrico de Fourier-Mellin se basa en la correlación de la fase y las propiedades del análisis de Fourier. El método de la correlación de la fase puede encontrar la traslación entre dos imágenes. Mientras que el método de Fourier-Mellin extiende la correlación de la fase para poder alinear imágenes que tienen tanto traslación como rotación [18] [19] [20] [21] [22] [23] [24]. De acuerdo con las propiedades de rotación y traslación de la transformada de Fourier, las transformaciones se representan de la siguiente forma

$$
\begin{aligned}
I_1(x, y) &= I_2(u, v) \\
u &= x \cos\theta_0 + y \sin\theta_0 - x_0 \\
v &= -x \sin\theta_0 + y \cos\theta_0 - y_0 \\
F_1(\omega_x, \omega_y) &= F_2(\omega_u, \omega_v) e^{-j(\omega_x x_0 + \omega_y y_0)} \\
\omega_u &= \omega_x \cos\theta_0 + \omega_y \sin\theta_0 \\
\omega_v &= -\omega_x \sin\theta_0 + \omega_y \cos\theta_0
\end{aligned}
$$

Se observa que la magnitud o amplitud del espectro de frecuencias $|F_1|$ es una réplica rotada de $|F_2|$ y ambos espectros comparten el mismo centro de rotación. Por lo tanto, se puede recuperar esta rotación representando el espectro $|F_1|$ y $|F_2|$ en coordenadas polares

$$|F_1(r, \theta| = |F_2(r, \theta - \theta_0|$$

La amplitud de Fourier en coordenadas polares se diferencia sólo por la traslación. Esto hace que se pueda usar el método de correlación de fase para encontrar esta traslación y estimar θ_0. Este método ha sido extendido para encontrar la escala por medio de mapear la amplitud de Fourier en coordenadas log-polar. Por lo tanto, se encuentra la escala y rotación mediante una correlación de fase, que obtiene la cantidad de desplazamiento en el espacio $(\log r, \theta)$.

La ventaja de este método es que es tolerante al ruido aditivo. Pero por otro lado sólo puede obtener buenos resultados con cambios de escala y rotación moderados. Esto es debido a que cuando se realiza un cambio de escala o rotación importante, se generan efectos en los bordes que pueden alterar de manera dramática los coeficientes de Fourier [21] [25] [26].

4.14. Pirámide de multiples resoluciones

Una pirámide de resoluciones múltiples, o espacio de escalas, consiste en un conjunto de imágenes que representan a una imagen en múltiples resoluciones. La imagen original se sitúa en la base de la pirámide y es reducida por un factor de escala constante en cada dimensión para generar el siguiente nivel. Esto es repetido nivel a nivel hasta que se alcanza la cúspide de la pirámide. Lo habitual es usar un factor de escala 2, y por tanto el tamaño de la imagen en el nivel i es reducido respecto a la original en un factor de 2^i en cada dimensión. Para que los niveles sean más significativos se suele llamar nivel más fino al nivel 0 y nivel más grueso al último nivel.

El uso de esta técnica tiene dos ventajas principales. La primera es que cuando se aplica el método LMA al nivel más grueso de la pirámide, el número de pixels disminuye en un factor de $2^{2(n-1)}$. Esto conlleva a una gran mejora computacional puesto que la mayoría de las iteraciones son ejecutadas en los niveles más gruesos. La segunda es que al disminuir el tamaño de la imagen, esta queda suavizada puesto que es equivalente a realizar un desenfoque gaussiano. Este suavizado de la imagen provoca que χ^2 sea calculada en imágenes más suaves y esta suavidad evita que la minimización quede atrapada en mínimos locales. Como en el nivel más grueso permanecen sólo las características de mayor tamaño, el proceso de alineamiento va desde el nivel más grueso hasta el más fino de manera progresiva. De manera adicional, esta aproximación en el nivel anterior es pasada al siguiente nivel como estimación inicial de los parámetros, aunque para ello es necesario escalar los parámetros de manera correspondiente a lo largo de los niveles. Si el factor de escala entre los niveles es

$$s : x^{i+1} = sx^i$$

$$y^{i+1} = sy^i$$

$$v^{i+1} = sv^i$$

donde

$$u^i = \frac{a_0^i x^i + a_1^i y^i + a_2^i}{a_6^i x^i + a_7^i y^i + 1}$$

$$v^i = \frac{a_3^i x^i + a_4^i y^i + a_5^i}{a_6^i x^i + a_7^i y^i + 1}$$

Sustituyendo las coordenadas del siguiente nivel más fino en las ecuaciones anteriores se obtiene

$$\frac{u^{i+1}}{s} = \frac{a_0^i \frac{x^{i+1}}{s} + a_1^i \frac{y^{i+1}}{s} + a_2^i}{a_6^i \frac{x^{i+1}}{s} + a_7^i \frac{y^{i+1}}{s} + 1}$$

$$\frac{v^{i+1}}{s} = \frac{a_3^i \frac{x^{i+1}}{s} + a_4^i \frac{y^{i+1}}{s} + a_5^i}{a_6^i \frac{x^{i+1}}{s} + a_7^i \frac{y^{i+1}}{s} + 1}$$

Multiplicando ambos lados por s se obtiene

$$\begin{aligned} u^{i+1} &= \frac{a_0^{i+1} x^{i+1} + a_1^{i+1} y^{i+1} + a_2^{i+1}}{a_6^{i+1} x^{i+1} + a_7^{i+1} y^{i+1} + 1} \\ &= \frac{a_0^i x^{i+1} + a_1^i y^{i+1} + s a_2^i}{a_6^i \frac{x^{i+1}}{s} + a_7^i \frac{y^{i+1}}{s} + 1} \end{aligned}$$

$$\begin{aligned} v^{i+1} &= \frac{a_3^{i+1} x^{i+1} + a_4^{i+1} y^{i+1} + a_5^{i+1}}{a_6^{i+1} x^{i+1} + a_7^{i+1} y^{i+1} + 1} \\ &= \frac{a_3^i x^{i+1} + a_4^i y^{i+1} + s a_5^i}{a_6^i \frac{x^{i+1}}{s} + a_7^i \frac{y^{i+1}}{s} + 1} \end{aligned}$$

y por lo tanto la relación entre los parámetros es

$$\begin{array}{lll} a_0^{i+1} = a_0^i & a_1^{i+1} = a_1^i & a_2^{i+1} = s a_2^i \\ a_3^{i+1} = a_3^i & a_4^{i+1} = a_4^i & a_5^{i+1} = s a_5^i \\ a_6^{i+1} = \frac{a_6^i}{s} & a_7^{i+1} = \frac{a_7^i}{s} & \end{array}$$

En el caso habitual, $s = 2$, por lo tanto los parámetros de traslación a_2 y a_5 son multiplicados por 2 y los parámetros a_6 y a_7 son divididos por 2.

4.15. Levenberg-Marquardt Optimizado

Esta modificación está basada en el artículo de [27], donde el alineamiento es realizado en imágenes médicas sujetas a transformaciones de rotación, escala y traslación. Por suerte, esta forma de proceder sigue siendo válida si se extiende al caso de transformaciones perspectivas.

En el método LMA estándar, se calcula en vector $B_{8\times 1}$ y la matriz Hessiana $H_{8\times 8}$ en cada iteración. En realidad el cálculo de la Hessiana es innecesario realizarlo en cada iteración, pudiendo ser realizado una única vez al comienzo. Considérese la siguiente función objetivo que establece una medida de similitud entre I_1 e I_2

$$\chi^2(a) = \|I_1(u) - I_2(T_A\{x\})\|^2$$

donde T_A representa la transformación que provoca la matriz de transformación perspectiva 3×3

Se puede suponer que I_2 se transforma en I_1 tras una serie de transformaciones perspectivas A.

Durante el proceso iterativo, las nuevas estimaciones de A son calculadas de la siguiente manera

$$A_{i+1} = A_i + \Delta A_i$$

donde $A_i = I + \Delta A_1 + \Delta A_2 + \ldots + \Delta_{i-1}$.

Como I_2 es transformada en cada iteración, la matriz Hessiana necesita ser recalculada porque es una representación del gradiente de I_2y por lo tanto la matriz Hessiana es responsable del cálculo de los términos anteriores ΔA.

El propósito de esta modificación es eliminar el cálculo de la matriz Hessiana. Esto se puede conseguir transformando el problema en otro donde I_1 sea transformada en I_2 sin tener que modificar I_2 entre iteraciones sucesivas. Esta aproximación permite calcular la matriz Hessiana una única vez, en la primera iteración.

Para poder determinar las nuevas estimaciones de los parámetros de la transformación perspectiva de A en el método modificado, se debe expresar χ^2 en términos de una transformación tal que mapee I_1 en I_2. La diferencia fundamental entre el método LMA estándar y el modificado es que en el método estándar se actualiza la estimación actual mediante el cambio de los valores iniciales hacia el mínimo global, mientras que en el modificado se trae el mínimo global hacia el valor inicial. La consecuencia de esta definición se puede resumir con la siguiente regla del LMA modificado

$$\begin{aligned} A_{i+1}^{-1} &= (I + \Delta A_i)^{-1}(I + \Delta A_{i-1})^{-1} \ldots (I + \Delta A_2)^{-1}(I + \Delta A_1)^{-1} \\ &= (I + \Delta A_i)^{-1} A_i^{-1} \end{aligned}$$

Una diferencia importante entre el método estándar y el modificado es la forma en la que los parámetros son actualizados en cada iteración. En el método estándar los parámetros de inicio son elegidos como la matriz identidad I como

suposición inicial para el punto P_0. A continuación se calculan las derivadas direccionales de I_2, $g_x = \partial I_2/\partial x$ y $g_y = \partial I_2/\partial y$. Estos procesos son del orden de $O(N \times 3 \times 3)$, donde N es el número de pixels y 3×3 es el tamaño del *kernel* o matriz de convolución. El método estándar proporciona un ΔA que se usa para añadirlo a la suposición inicial I para moverse desde el punto P_0 hasta el P_1 en la curva de $\chi^2(a)$. En la siguiente iteración, a causa de que la imagen I_2 es transformada por $A + \Delta A$, es necesario volver a calcular g_x y g_y para encontrar el nuevo ΔA. Por lo tanto en el método LMA estándar, la solución óptima P_i se desliza por la curva de $\chi^2(a)$. Sin embargo, en el método modificado se cambia la curva de $\chi^2(a)$ hacia la suposición inicial P_0. Esto es conseguido remuestreando I_1 mediante la transformación inversa $(I + \Delta A_i)^{-1} A_i^{-1}$. Por consecuencia, la imagen I_1' es "acercada" a I_2. Ahora, la nueva imagen I_1' y la imagen I_2 son utilizadas para minimizar $\chi^2(a)$. El resultado genera un nuevo ΔA que es añadido siempre a I, la suposición inicial en el punto P_0. Puesto que I_2 no cambia, no es necesario calcular g_x y g_y.

Es útil rescribir la función χ^2 en términos del nuevo mapeo directo, así como el mapeo inverso. Esta descomposición va a permitir aplicar una parte importante de la transformación a $I_1(u)$. Como resultado, la transformación inversa restante es pequeña y al aplicarla a $I_2(x)$ permite prescindir del cálculo de la Hessiana.

Supóngase que se descompone la transformación $T\{\}$ en dos transformaciones más pequeñas $T_A T_{I+\Delta A}\{\}$. De esta manera $T_A\{\}$ es la transformación de la iteración previa y $T_{I+\Delta A}$ es la transformación, pequeña, que minimiza $\chi^2(a)$ en el método Levenberg-Marquardt.

$$\chi^2(a) = \|I_1(u) - I_2(T_A\{T_{I+\Delta A}\{x\}\})\|^2 \tag{4.15.1}$$

$$= \frac{1}{|A|}\|I_1(T_{A^{-1}}\{u\}) - I_2(T_{A^{-1}}\{T_A\{T_{I+\Delta A}\{x\}\}\})\|^2 \tag{4.15.2}$$

$$= \frac{1}{|A|}\|I_1(T_{A^{-1}}\{u\}) - I_2(T_{I+\Delta A}\{x\})\|^2 \tag{4.15.3}$$

$$= \frac{1}{|A(I+\Delta A)|}\|I_1(T_{(I+\Delta A)^{-1}A^{-1}}\{u\}) - I_2(x)\|^2 \tag{4.15.4}$$

$$= \frac{1}{|A(I+\Delta A)|}\|I_1(T^{-1}\{u\}) - I_2(x)\|^2 \tag{4.15.5}$$

Estas ecuaciones muestran los pasos necesarios para transformar $T_A T_{I+\Delta A}\{\}$ desde el sistemas de coordenadas de I_2 en el sistema de coordenadas de I_1 con la normalización correspondiente. En vez de minimizar la función 4.15.1 se va a minimizar la 4.15.3 respecto a los parámetros δA.

En el método LMA modificado es necesario derivar $\partial I_2(T_{I+\Delta A}\{x\})/\partial \Delta a$ y la regla de actualización para cada parámetro de la transformación. Se puede descomponer $\partial I_2(T_{I+\Delta A}\{x\})/\partial \Delta a$ de la siguiente manera

$$\frac{\partial I_2(T_{I+\Delta A}\{x\})}{\partial \Delta a} = \frac{\partial I_2(T_{I+\Delta A}\{x\})}{\partial x}\frac{\partial x}{\partial \Delta a}$$

y puesto que la transformación Δa es lo suficientemente pequeña, se pueden realizar las siguientes aproximaciones

$$I_2(T_{I+\Delta A}\{x\}) \approx I_2$$

$$\frac{\partial x}{\partial \Delta a} \approx \frac{\partial u}{\partial \Delta a}$$

de esto se deduce que

$$\frac{\partial I_2(T_{I+\Delta A}\{x\})}{\partial \Delta a} \approx \frac{\partial I_2}{\partial x}\frac{\partial x}{\partial \Delta a} \approx \frac{\partial I_2}{\partial x}\frac{\partial u}{\partial \Delta a}$$

donde $\frac{\partial I_2}{\partial x}$ es el gradiente de I_2. Entonces, $\frac{\partial I_2(x)}{\partial \Delta a_k}$ para los ocho parámetros de la transformación perspectiva resultan

$$\begin{aligned}
\frac{\partial I_2}{\partial \Delta a_0} &= \frac{\partial I_2}{\partial x}\frac{\partial u}{\partial \Delta a_0} = g_x\frac{x}{w} \\
\frac{\partial I_2}{\partial \Delta a_1} &= \frac{\partial I_2}{\partial x}\frac{\partial u}{\partial \Delta a_1} = g_x\frac{y}{w} \\
\frac{\partial I_2}{\partial \Delta a_2} &= \frac{\partial I_2}{\partial x}\frac{\partial u}{\partial \Delta a_2} = g_x \\
\frac{\partial I_2}{\partial \Delta a_3} &= \frac{\partial I_2}{\partial y}\frac{\partial v}{\partial \Delta a_3} = g_y\frac{x}{w} \\
\frac{\partial I_2}{\partial \Delta a_4} &= \frac{\partial I_2}{\partial y}\frac{\partial v}{\partial \Delta a_4} = g_y\frac{y}{w} \\
\frac{\partial I_2}{\partial \Delta a_5} &= \frac{\partial I_2}{\partial y}\frac{\partial v}{\partial \Delta a_5} = g_y \\
\frac{\partial I_2}{\partial \Delta a_6} &= \frac{\partial I_2}{\partial x}\frac{\partial u}{\partial \Delta a_6} + \frac{\partial I_2}{\partial y}\frac{\partial v}{\partial \Delta a_6} \\
&= -x\left(g_x\frac{a_0x + a_1y + a_2}{(a_6x + a_7y + 1)^2} + g_y\frac{a_3x + a_4y + a_5}{(a_6x + a_7y + 1)^2}\right) \\
\frac{\partial I_2}{\partial \Delta a_7} &= \frac{\partial I_2}{\partial x}\frac{\partial u}{\partial \Delta a_7} + \frac{\partial I_2}{\partial y}\frac{\partial v}{\partial \Delta a_7} \\
&= -y\left(g_x\frac{a_0x + a_1y + a_2}{(a_6x + a_7y + 1)^2} + g_y\frac{a_3x + a_4y + a_5}{(a_6x + a_7y + 1)^2}\right)
\end{aligned}$$

En el método LMA estándar, la regla de actualización es la siguiente

$$A_{new} = A_{old} + \Delta A$$

mientras que en el LMA modificado es

$$\begin{aligned}
A_{new} &= A_{old}(I + \Delta A) \\
&= \begin{bmatrix} a_0 & a_1 & a_2 \\ a_3 & a_4 & a_5 \\ a_6 & a_7 & 1 \end{bmatrix} \times \begin{bmatrix} \Delta a_0 + 1 & \Delta a_1 & \Delta a_2 \\ \Delta a_3 & \Delta a_4 + 1 & \Delta a_5 \\ \Delta a_6 & \Delta a_7 & 1 \end{bmatrix}
\end{aligned}$$

A continuación se muestra un esquema de como deberá ser el Algoritmo Levenberg-Marquardt modificado

```
Construir las pirámides de resoluciones para I_1 e I_2
Inicializar los parámetros como la matriz identidad
Inicializar λ con un valor dado, por ejemplo λ = 0,001
para i = L hasta 0 //L el nivel más grueso de la pirámide
    Calcular los gradientes direccionales: ∂I_2^i/∂x y ∂I_2^i/∂y
    Calcular la matriz Hessiana 8 × 8
    mientras_que ( |Δχ²(a)| > T_1 || λ < T_2 )
        Aplicar la transformación T_{A^-1} sobre I_1 en el nivel i
        Calcular el vector B
        Resolver el sistema lineal (H(a) + λI)Δa = −B para Δa
        Evaluar χ²(a + Δa)
        si χ²(a + Δa) < χ²(a) entonces
            λ ← λ/10
            a ← a + Δa
        si_no
            λ ← λ * 10
        fin_si
    fin_mientras_que
fin_para
```

4.16. Lenguajes contemplados

4.16.1. C++

Este lenguaje se posiciona como el mejor candidato por motivos de eficiencia y la disponibilidad de múltiples *frameworks* para la visión por computador, en especial OpenCV.

La gran ventaja de este lenguaje es que la gran mayoría de los desarrollos que existen en este ámbito están realizados en C++, lo que proporciona una buena base para trabajar y documentación disponible de la misma. Esto último es lo que nos interesa de manera principal de este lenguaje, ya que salvo por el hecho de cargar las imágenes y su manejo eficiente se pretende implementar el resto de funcionalidad, salvo funciones concretas utilizadas de manera habitual.

4.16.2. R

Se trata de un lenguaje cuyo propósito es el análisis estadístico y dispone de gran parte de la instrumentación matemática necesaria en el ámbito de la visión por computador, aunque no de manera específica.

Los inconvenientes encontrados son que tiene una sintaxis y modo de trabajo diferentes al que estamos acostumbrados y hay poca documentación o trabajos realizados en este ámbito.

4.16.3. Python

Se trata de un lenguaje que permite un prototipado rápido porque es de muy alto nivel e incorpora manejo de memoria dinámico entre otras cosas. Además existe un *wrapper* para OpenCV que hace que trabaje de manera muy eficiente manteniendo el manejo dinámico de la memoria.

El inconveniente encontrado es que el *wrapper* tiene algunas particularidades que lo hacen diferir de como trabaja python y esto provoca que para poder utilizar algunas funcionalidades haya que hacerlo de manera especial y se pierda mucho tiempo en estos detalles, y este tiempo es el que se esperaba ganar al utilizarlo como lenguaje de prototipado.

Pyhton también cuenta con una librería propia para tratamiento de imágenes, *PIL (Python Image Library)*, esta tiene un modo de trabajo muy consistente y cómodo pero las pruebas realizadas con esta librería han sido un orden o dos de magnitud más lentos que con OpenCV y no en las tareas más costosas computacionalmente.

Se ha contemplado también el uso de Cython, que es una variante de Python compatible pero que añadiendo algunas directivas, sobre todo de tipado, permite compilarlo a código C haciéndolo comparable en eficiencia a este. No se ha profundizado en esta aproximación puesto que no es un proyecto maduro y se ha preferido evitar tener problemas por este motivo.

4.16.4. Java

Java dispone de un *wrapper* para OpenCV a través de *JNI (Java Native Interface)* lo que en principio lo haría bastante eficiente.

Se ha contemplado el uso de este lenguaje porque es el que se emplea de manera mayoritaria en los desarrollos del grupo y es interesante desde el punto de vista de la integración en otras aplicaciones.

4.16.5. Matlab/Octave

Ambos son lenguajes de muy alto nivel que permiten un prototipado rápido de los algoritmos. Así mismo son lenguajes orientados al tratamiento matemático, lo que nos es de bastante utilidad. Además en teoría también deberían ser capaces de acceder a OpenCV mediante un *wrapper*. También hay que tener en cuenta que la sintaxis de ambos es muy similar y salvo algunos detalles se puede decir que son compatibles.

La ventaja principal de Octave es que su licencia es GPL y su mayor inconveniente encontrado es que tiene algunos errores en su implementación que hacen desperdiciar tiempo en solucionarlos. Por ejemplo, no cargar una imagen por un fallo en el algoritmo que compone las rutas (se suministró a los desarrolladores de Octave la versión corregida). Por contra, el entorno y el flujo de trabajo en Matlab está mucho más pulido, es más cómodo de trabajar y no presenta errores de este tipo.

4.16.6. OpenCV

OpenCV (Open Source Computer Vision) es una librería cuyo propósito es proporcionar funciones para la visión por computador en tiempo real, bajo licencia BSD. De esta librería lo que realmente nos interesa es usarla para delegar en ella el manejo en memoria de las imágenes, puesto que lo hace de manera muy eficiente y ofrece interfaces para C++, C, C#, Ruby, Python y Java manteniendo la eficiencia y permitiendo el manejo de las mismas como memoria dinámica.

También vamos a hacer uso de algunas de sus funciones como por ejemplo las transformaciones de las imágenes mediante la matriz de homografía ya que es una funcionalidad muy probada y no produce artefactos en la imagen.

4.17. Acrónimos y abreviaturas

FFT Fast Fourier Transform

GMAPS Google MAPS

GPL General Public License

GPU Graphics processing unit

HOG Histograms of Oriented Gradients

IDE Infraestructuras de Datos Espaciales

JNI Java Native Interface

LMA Levenberg-Marquardt Algorithm

OpenCV Open source Computer Vision

PCA Principal Component Analysis

PIL Python Image Library

PNOA Plan Nacional de Ortofotografía Aérea

RANSAC RANdom SAmple Consensus

SSD Sum of Squared Differences

SIFT Scale-invariant feature transform

SURF Speeded Up Robust Feature

SVT Scalable Vocabulary Tree

Índice de figuras

Referencias

[1] G. W. Siavash Zokai, "Image registration using log-polar mappings for recovery of large-scale similarity and projective transformations," *IEEE Transactions on Image Processing*, vol. 14, October 2005. 1.1, 2.2, 2.2, 3

[2] G. Wolberg and S. Zokai, "Image registration for perspective deformation recovery," in *in Proc. SPIE Conf. Automatic Target Recognition X*, p. 12, 2000. 1.1, 2.2

[3] L. G. Brown, "A survey of image registration techniques," *ACM Comput. Surv.*, vol. 24, pp. 325–376, December 1992. 1.3, 2.2

[4] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *In ECCV*, pp. 404–417, 2006. 2.1

[5] G. Bradski and A. Kaehler, *Learning OpenCV: computer vision with the OpenCV library*. O'Reilly Series, O'Reilly, 2008. 2.1

[6] R. Laganičre, *OpenCV 2 Computer Vision Application Programming Cookbook*. Packt Publishing, Limited, 2011. 2.1

[7] R. Szeliski, *Computer Vision: Algorithms and Applications*. Texts in Computer Science, Springer, 2010. 2.1

[8] R. Szeliski, "Image alignment and stitching: A tutorial," tech. rep., MSR-TR-2004-92, Microsoft Research, 2004, 2005. 2.1

[9] B. Zitová and J. Flusser, "Image registration methods: a survey," *Image and Vision Computing*, vol. 21, pp. 977–1000, 2003. 2.2

[10] D. G. Lowe, "Object recognition from local scale-invariant features," 1999. 2.2

[11] Y. Dufournaud, C. Schmid, and R. Horaud, "Matching images with different resolutions," in *In Proceedings of the Conference on Computer Vision and Pattern Recognition, Hilton Head Island, South*, pp. 612–618, 2000. 2.2

[12] G. Wolberg and S. Zokai, "Robust image registration using log-polar transform," in *In Proc. IEEE Int. Conf. image processing*, pp. 493–496, 2000. 2.2

[13] G. Ritter and J. Wilson, "Handbook of computer vision algorithms in image algebra," *Computer*, p. 417 pages, 2000. 2.3

[14] F. M. Candocia, "Jointly registering images in domain and range by piecewise linear comparametric analysis," 2003. 2.3

[15] W. H. Press, W. T. Vetterling, S. A. Teukolsky, and B. P. Flannery, *Numerical Recipes in C++: the art of scientific computing*. New York, NY, USA: Cambridge University Press, 2nd ed., 2002. 2.3

[16] N. Paragios, Y. Chen, and O. Faugeras, *Handbook of Mathematical Models in Computer Vision*, vol. XXXIII. Springer, 2006. 2.3

[17] S. Mann, "Comparametric equations with practical applications in quantigraphic image processing," 2000. 2.3

[18] D. Casasent and D. Psaltis, "Position, rotation, and scale invariant optical correlation," *Appl. Opt.*, vol. 15, pp. 1795–1799, Jul 1976. 4.13

[19] E. De Castro and C. Morandi, "Registration of translated and rotated images using finite fourier transforms," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-9, pp. 700 –703, sept. 1987. 4.13

[20] Q.-S. Chen, M. Defrise, and F. Deconinck, "Symmetric phase-only matched filtering of fourier-mellin transforms for image registration and recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, pp. 1156 –1168, dec 1994. 4.13

[21] B. S. Reddy and B. N. Chatterji, "An fft-based technique for translation, rotation, and scale-invariant image registration," *Image Processing IEEE Transactions on*, vol. 5, no. 8, pp. 1266–1271, 1996. 4.13

[22] S. hsu Chang, T. F. hsuan Cheng, W. hsing Hsu T, and G. zua Wu T, "Fast algorithm for point pattern matching: Invariant to translations rotations and scale changes," *Pattern Recognition*, pp. 311–320, 1997. 4.13

[23] L. Lucchese and G. Cortelazzo, "Noise-robust estimation of planar rototranslations with high precision," in *Image Processing, 1997. Proceedings., International Conference on*, vol. 1, pp. 699 –702 vol.1, oct 1997. 4.13

[24] L. Lucchese, G. M. Cortelazzo, and C. Monti, "High resolution estimation of planar rotations based on fourier transform and radial projections," *IEEE International Symposium on Circuits and Systems*, 1997. 4.13

[25] H. Stone, M. Orchard, and E.-C. Chang, "Subpixel registration of images," in *Signals, Systems, and Computers, 1999. Conference Record of the Thirty-Third Asilomar Conference on*, vol. 2, pp. 1446 –1452 vol.2, 1999. 4.13

[26] M. M. Harold S. Stone and, Bo Tao and, "Analysis of image registration noise due to rotationally dependent aliasing," *J. Visual Communication and Image Representation*, vol. 14, no. 2, pp. 114–135, 2003. 4.13

[27] P. Thevenaz, U. Ruttimann, and M. Unser, "A pyramid approach to subpixel registration based on intensity," *Image Processing, IEEE Transactions on*, vol. 7, pp. 27 –41, jan 1998. 4.15