

UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE INGENIERIA ELECTRÓNICA

**SISTEMA DE GENERACIÓN DE MOSAICOS 2D
PARA ROBOTS MÓVILES A PARTIR DE VIDEO
MONOCULAR**

Por:

Victor Yovanni Garcia Carmona

Realizado con la asesoría de:
José de la Cruz Cappelletto Fuentes
PROYECTO DE GRADO

Presentado ante la Ilustre Universidad Simón Bolívar
como requisito parcial para optar al título de
Ingeniero Electrónico

Sartenejas, Marzo de 2018



UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE INGENIERÍA ELECTRÓNICA

ACTA FINAL PROYECTO DE GRADO

SISTEMA DE GENERACIÓN DE MOSAICOS 2D PARA ROBOTS
MÓVILES A PARTIR DE VIDEO MONOCULAR

Presentado por:
Victor Yovanni Garcia Carmona

Este Proyecto de Grado ha sido aprobado por el siguiente jurado examinador:

José de la Cruz Cappelletto Fuentes

Novel Antonio Certad H.

Gerardo Fernandez López

Sartenejas, @día de Mayo de 2018



UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE INGENIERÍA ELECTRÓNICA
**SISTEMA DE GENERACIÓN DE MOSAICOS 2D PARA ROBOTS
MÓVILES A PARTIR DE VIDEO MONOCULAR**
PROYECTO DE GRADO
PRESENTADO POR:
Victor Yovanni Garcia Carmona, Carnet: 12-10738

RESUMEN

Al realizar tareas de exploración para el análisis del suelo en espacios aéreos o en fondo marino, es muy común emplear sistemas de adquisición basados en captura de videos para su posterior análisis. En la actualidad, el incremento de la tecnología sobre el procesamiento de datos, ha permitido que los algoritmos de visión por computadora coloquen a la cámara como principal sensor para la reconstrucción de entornos recorridos por vehículos móviles. El presente trabajo se encuentra enfocado al análisis e implementación de distintos algoritmos para la reconstrucción de un mosaico 2D (dos dimensiones) del suelo que recorre un robot, a partir de la información proveniente de una cámara monocular ubicada en la parte inferior de este. El robot en cuestión puede realizar recorridos aéreos para realizar la adquisición del video, o incluso trayectorias mas desafiantes como serían las aplicaciones subacuáticas. Para esto, se implementarán distintos algoritmos usando técnicas de procesamiento de imágenes y visión por computadora, para la elaboración de un sistema automatizado que permita generar un mapa en dos dimensiones de la trayectoria recorrida por el vehículo móvil, mejorando la detección de puntos clave y optimizando el cálculo de las matrices de transformación para la alineación de imágenes en el mosaico, logrando así un mapa con la menor distorsión posible. Además de realizar análisis sobre el error de proyección de dichas imágenes en el mapa del suelo generado.

Palabras clave: mosaico, video monocular, puntos clave, matriz de transformación.

Agradecimientos

Índice general

Resumen	I
Agradecimientos	I
Índice de Figuras	IV
Lista de Tablas	VI
Acrónimos y Símbolos	VII
1. Introducción	1
1.1. Antecedentes	1
1.2. Justificación y planteamiento del problema	2
1.3. Objetivos	4
1.3.1. Objetivo General	4
1.3.2. Objetivos Específicos	4
1.4. Estructura del trabajo	5
2. Sistemas de generación de mosaico	7
2.1. Estado del arte	8
2.2. Esquema propuesto	16
2.3. Librería de desarrollo	17
2.3.1. OpenCV	17
3. Extracción de puntos característicos	19
3.1. Introducción	19
3.2. Revisión teórica	20
3.2.1. Detectores y descriptores de características	20
3.2.2. Emparejadores de puntos característicos	29
3.3. Módulo comparativo	30
3.3.1. Metodología	31
3.3.2. Resultados	34
3.4. Resumen	39

4. Alineación de imágenes	41
4.1. Introducción	41
4.2. Revisión teórica	41
4.2.1. Transformaciones geométricas	42
4.2.2. Estimación de Homografía	48
4.3. Generación de sub-mosaicos	52
4.3.1. Matriz de transformación promedio	53
4.3.2. Selección de puntos característicos	56
4.3.3. Seguimiento de puntos	58
4.3.4. Relación con vecinos	59
4.4. Corrección euclíadiana	61
4.5. Resultados	63
4.6. Resumen	63
5. Unión de imágenes	64
5.1. Introducción	64
5.2. Línea de costura	65
5.2.1. Corte por grafo	65
5.3. Corrección de color	67
5.3.1. Método de Reinhard	68
5.4. Fusión de imágenes	69
5.4.1. Fusión ponderada	70
5.4.2. Fusión ponderada piramidal	71
5.5. Resultados	73
5.6. Resumen	73
6. Conclusiones y trabajos futuros	74
A. Instalación de librería OpenCV y dependencias	78
A.1. @sección	78
A.1.1. @subsección	78

Índice de figuras

1.1.	Robot móvil OpenROV	2
1.2.	Efectos en el cambio del punto de vista	3
1.3.	Logo del software Hugin	3
2.1.	Proceso básico para la generación de mosaico	9
2.2.	Clasificación de los algoritmos de generación de mosaico	10
2.3.	Esquema propuesto para la construcción del mosaico	17
2.4.	Logo de la librería OpenCV	18
3.1.	Caracterización de regiones en una imagen	21
3.2.	Ventana de búsqueda para de detección de esquinas	22
3.3.	Efecto del escalado sobre las esquinas	22
3.4.	Detector SIFT	24
3.5.	Descriptor SIFT	25
3.6.	Detector y descriptor SURF	25
3.7.	Detector y descriptor ORB	26
3.8.	Filtro no lineal propuesto por KAZE	28
3.9.	Emparejamiento de puntos característicos	29
3.10.	Combinación de algoritmos para el análisis de rendimiento	31
3.11.	Estiramiento del histograma	34
3.12.	imágenes representativas del conjunto <i>Chuspa</i>	35
3.13.	imágenes representativas del conjunto <i>ScottReef 25</i>	36
3.14.	imágenes representativas del conjunto <i>Mochima</i>	37
3.15.	imágenes representativas del conjunto <i>Grava</i>	38
4.1.	Tipos de transformaciones geométricas	47
4.2.	Relacion de homografías	48
4.3.	Mejor modelo - RANSAC	51
4.4.	Generación de sub mosaicos	52
4.5.	Estimación de homografía promedio	54
4.6.	Distorsion de imagen	54
4.7.	Selección de mejores parejas	56
4.8.	Selección de mejores puntos, ejemplo con plano	57
4.9.	Seguimiento de puntos	59
4.10.	Superposición de imágenes vecinas, mismo sentido	60
4.11.	Superposición de imágenes vecinas, cambio de sentido	60

4.12. Corrección euclíadiana	62
5.1. Corte por grafo	67
5.2. Espacio de color CIELAB	68
5.3. Fusión ponderada simple	70
5.4. Construcción de pirámide de gaussianas	71
5.5. Construcción de pirámide de laplacianas	72
5.6. Ejemplo de fusión piramidal	73

Índice de Tablas

3.1.	Comparación de rendimiento usando imágenes del conjunto <i>Chuspa</i>	35
3.2.	Comparación de rendimiento usando imágenes del conjunto <i>Chuspa</i> , luego de aplicar estiramiento del histograma	35
3.3.	Comparación de rendimiento usando imágenes submarinas del con- junto <i>ScottReef 25</i>	36
3.4.	Comparación de rendimiento usando imágenes submarinas del con- junto <i>ScottReef 25</i> , luego de aplicar estiramiento del histograma . .	36
3.5.	Comparación de rendimiento usando imágenes del conjunto <i>Mochima</i>	37
3.6.	Comparación de rendimiento usando imágenes del conjunto <i>Mochi- ma</i> , luego de aplicar estiramiento del histograma	37
3.7.	Comparación de rendimiento usando imágenes del conjunto <i>Grava</i> .	38
3.8.	Comparación de rendimiento usando imágenes del conjunto <i>Grava</i> , luego de aplicar estiramiento del histograma	38

Acrónimos y Símbolos

Dedicatoria

A @personasImportantes, por @razonesDedicatoria.

Capítulo 1

Introducción

La navegación y exploración en áreas de difícil acceso mediante el uso de robots, es una tarea que se ha venido desarrollando en el Grupo de Investigación y Desarrollo en Mecatrónica de la USB (*GIDM*) desde hace mucho tiempo. Donde una de las aplicaciones mas demandadas, es la tarea de reconstruir un mapa 2D de la superficie mapeada por los robots utilizados. En el presente capítulo se pretende introducir los trabajos previos y avances que se han tenido en el desarrollo de este tipo aplicaciones, específicamente en el *GIDM*, que dieron origen y motivación para la realización del proyecto. Además de postular un serie de problemas que el presente trabajo busca solucionar.

1.1. Antecedentes

En el *GIDM* se han realizado grandes avances en el desarrollo de equipos y plataformas robóticas para actividades de investigación, exploración e inspección de ambientes no estructurados. Usualmente cuando se opera en este tipo de ambientes, en busca de realizar exploraciones mas eficientes y a mayor escala, se emplean vehículos operados remotamente *ROV* (del inglés: Remotely Operated Vehicles) equipados con cámaras de vídeo. O bien, para el caso de aplicaciones subacuáticas también se suelen utilizar vehículos autónomos submarinos *AUV* (del inglés: Automated Underwater Vehicles), mientras que para exploraciones aéreas se hace uso de vehículos aéreos no tripulados *UAV* (del inglés: Unmanned Aerial Vehicle).

En este sentido, se tienen proyectos como el presentado por *Danilo, D.* [1], cuyo trabajo de grado consistió en el desarrollo en un sistema de operación remota para un robot submarino (*ROV*), con la finalidad de implementarlo en tareas de exploración. Con objetivos similares, *Said, A.* [2] basó su proyecto de grado en la instrumentación y control de un robot cuadricóptero volador(*AUV*).

Adicional a los proyectos antes mencionados, en el el *GIDM* se cuenta con un vehículo submarino llamado OpenROV¹, el cual es un robot maniobrado remotamente de baja envergadura, diseñado especialmente para operar bajo el agua.

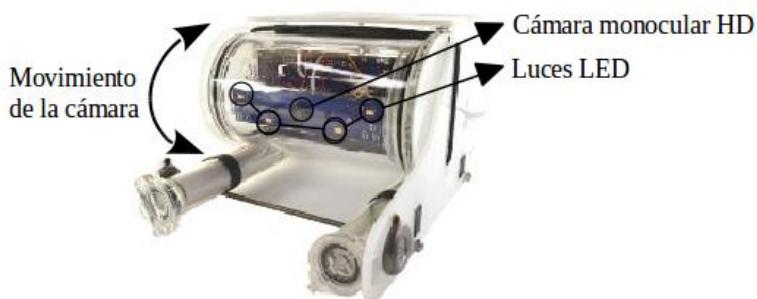


FIGURA 1.1: Robot móvil OpenROV

Si bien se cuenta con un conjunto de plataformas robóticas adaptadas para tareas de exploración, hasta el momento no se han desarrollados sistemas basados en visión que permitan incluirse en la etapa de navegación y mapeo de dichos vehículos, siendo la presente investigación la primera en abordar la tarea de la reconstrucción del suelo recorrido haciendo uso únicamente de una cámara de vídeo, sensor presente en todas las plataformas robóticas antes mencionados.

1.2. Justificación y planteamiento del problema

Cuando se habla de construir un mosaico 2D, se hace referencia al proceso de recortar y alinear imágenes, de tal forma que puedan ser representadas todas juntas en una sola gran imagen. Es importante considerar que las imágenes para este tipo de aplicaciones son capturadas desde diferentes ubicaciones de la cámara, a diferencia del proceso para elaborar imágenes panorámicas, en las cuales esta

¹ <https://www.openrov.com/products/openrov28/>

ubicación es una constante. Esta característica trae consigo uno de los principales problemas en la construcción de mosaicos, y se debe al efecto paralaje. Este efecto está asociado a la diferencia entre las posiciones aparentes de los objetos, según el punto desde donde se observa.

En la figura 1.2 se aprecia un ejemplo ilustrativo de esta definición, en la cual, si nos fijamos en el punto de vista **A**, se observa el triángulo a la izquierda del círculo, mientras que desde le punto **B** este orden se encuentra invertido.

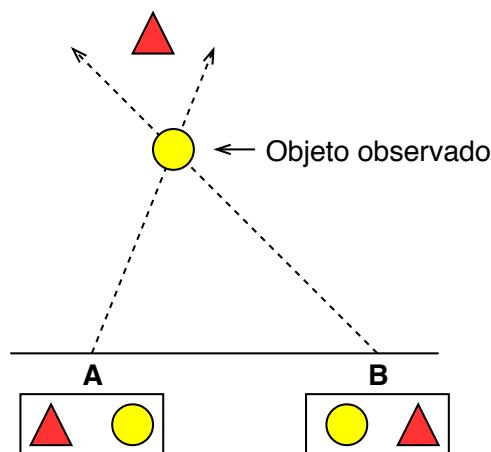


FIGURA 1.2: Efectos en el cambio del punto de vista.

Si bien, este problema afecta en gran medida la construcción del mosaico, no es el único presente, y se intensifican en aplicaciones de mapeo submarino, en las cuales, se evidencian efectos de distorsión de los objetos, absorción y cambios en la dirección de la luz, producto de pequeñas partículas suspendidas en el agua.

En el *GIDM* actualmente se utilizan mecanismos manuales para la elaboración de estos mapas, en específico, se hace uso de *softwares* como Hugin². Este es un programa de código abierto y gratuito bajo licencia GPL³, el cual esta dedicado a la generación de imágenes panorámicas, incluyendo funciones para el recorte, alineación y corrección de color; además de algoritmos para la optimización de parámetros en la cámara, y corrección de distorsión. Si



FIGURA 1.3: Logo del software Hugin

²hugin.sourceforge.net/

³<http://www.gnu.org/copyleft/gpl.html>

bien este software esta diseñado para la creación de imágenes panorámicas, permite el uso de varios tipos de proyecciones cartográficas, entre estas la rectangular, proyectando las imágenes sobre un plano recto.

Esta practica manual, además de limitar el alcance de los sistemas embebidos para el uso en navegación automática, requiere de una inversión de tiempo importante por medio del usuario en el proceso de selección y alineación de imágenes.

Atendiendo a esta necesidad, es necesario contar con un sistema que permita realizar la reconstrucción del suelo con la menor interacción posible del ser humano. Asimismo, con el fin de poder realizar operaciones de mapeo y localización simultanea *SLAM* (del inglés: Simultaneous Localization and Mapping), haciendo uso de las herramientas y robots existentes en el laboratorio, se requiere contar con un sistema basado en visión, que genere de forma automática un mapa 2D de la superficie sobre la que navega o sobrevuela el vehículo remoto, y que logre lidiar de manera efectiva ante los problemas previamente planteados.

1.3. Objetivos

1.3.1. Objetivo General

Analizar e implementar un sistema automatizado que permita la reconstrucción de un mapa en dos dimensiones, del suelo recorrido por robot, aéreo o submarino, a través de la información capturada por una cámara ubicada en su parte inferior.

1.3.2. Objetivos Específicos

- Análisis comparativo de métodos vigentes en la reconstrucción de mosaicos 2D, a partir de imágenes y videos de entrada.
- Implementación de modulo de pre-procesamiento y corrección de entrada.
- Análisis comparativo de métodos de detección y descripción de puntos característicos.

- Implementación de módulo de alineación de imágenes mediante la detección de puntos característicos.
- Cuantificar el error de proyección y distorsión en los modelos 2D generados.

1.4. Estructura del trabajo

Luego de presentar el planteamiento del problema y la descripción del proyecto, la presente investigación se encuentra dividida en 5 capítulos, organizados de la siguiente manera:

En el **Capítulo 2** se presenta una revisión del estado del arte sobre los algoritmos de generación de mosaico, en el cual se exponen los trabajos recientes y avances importantes en esta área de investigación. Al mismo tiempo, se describen los módulos principales que componen este tipo de sistemas. Luego, en base a los algoritmos y técnicas estudiadas se propone un esquema para un sistemas de generación de mosaico. Para finalizar, se presenta la librería de procesamiento de imágenes que se planteó utilizar para la implementación de los algoritmos propuestos.

El **Capítulo 3** inicia una revisión teórica en la cual se describe el funcionamiento de los algoritmos detectores, descriptores y emparejadores de características; y posteriormente se presentan resultados de pruebas comparativas entre los mas usados para este tipo de aplicaciones.

El módulo encargado de la alineación de imágenes en el mosaico, es descrito en el **Capítulo 4**. Al igual que el capítulo anterior, se presenta una revisión teórica de los conceptos necesarios para su implementación. Luego, se introduce el modelo de sub mosaicos, y la implementación de un conjunto de correcciones geométricas sobre este nuevo modelo. Finalmente se muestran los resultados de los algoritmos aplicados en esta sección, seguidos de sus respectivos análisis.

En el **Capítulo 5** se describe el modulo final del sistema, en donde se explica el funcionamiento de los algoritmos que corrigen visualmente el mosaico final. De

igual forma se muestran los resultados de su implementación, seguidos de una conclusión final sobre estos.

Finalmente, en el ***Capítulo 6*** se presentan las conclusiones finales, además de propuestas sobre recomendaciones y posibles implementaciones que pueden aportar mejoras y/o permitir la continuación del proyecto aquí planteado.

Capítulo 2

Sistemas de generación de mosaico

En este capítulo se presenta una revisión teórica del estado actual de las investigaciones que se han desarrollado en el área de procesamiento de imágenes, aplicado a la construcción de mosaicos, además de una reseña histórica de la evolución de dichos métodos. Debido a que la construcción de mosaicos ha sido y sigue siendo un área de investigación muy activa, existe una gran variedad de métodos y técnicas que se han empleado para este fin. Con el objetivo de recopilar esta información de manera estructurada, se presenta una clasificación de estos algoritmo en base al modo de abordar los módulos principales para la elaboración de estos mapas.

Con esto se pretende recuperar y trascender el conocimiento acumulado en esta área de estudio, además de familiarizar al lector con los conceptos básicos, necesarios para la comprensión del presente trabajo. Seguidamente se presenta el modelo del sistema de generación de mosaico propuesto en base a los algoritmos y técnicas que se han utilizado en las investigaciones mas recientes. Finalmente se introduce la librería de procesamiento de imágenes que se seleccionó para la implementación de todos los módulos necesarios.

2.1. Estado del arte

La elaboración de mosaicos para la construcción de mapas del suelo, se ha desarrollado incluso antes desde la era digital de las computadoras. Desde que el proceso de registrar fotografías ha existido, se comenzaron a usar para elaborar mapas topográficos [3], donde imágenes adquiridas a partir de globos aerostáticos o altas colinas eran unidas manualmente. Posteriormente, producto de los avances en materia de aeronáutica, el interés por la aerofotografía se incrementó en gran medida. En este mismo sentido se utilizaban aviones para el registro de imágenes a mayores altitudes, logrando cubrir grandes áreas en menor cantidad de tiempo. Pero debido a que no se alcanzaba suficiente altura, y se mantenía la necesidad de registrar grandes áreas, era requerido que los mapas se construyan mediante fotografías que se superpongan, de igual forma esta tarea se llevaba a cabo mediante técnicas manuales por medio de expertos.

La necesidad de registrar áreas aún más grandes siguió avanzando, motivado por la llegada de los satélites que eran capaces de enviar a tierra la información que obtenían de las cámaras. Los avances tecnológicos en materia de computación, y el creciente aumento de datos para esta aplicación, promovieron el desarrollo de técnicas de procesamiento digital de imágenes para dar solución a este tipo de problemas.

Con el desarrollo de cámaras cada vez más pequeñas y portátiles, así como también la llegada de vehículos no tripulados más compactos —ROV, UAV, AUV—, se produjeron grandes avances y nuevas técnicas por parte de centros de investigación en el área de la física, robótica y visión por computadora, que buscaron aportar soluciones para la realización automática de mosaicos. Esta vez, con un creciente enfoque en las aplicaciones más desafiantes como los ambientes submarinos.

Para explicar el proceso en el que consiste construir un mapa del suelo a partir de fotografías, lo podemos separar en tres simples pasos. Si bien, éste se ilustran gráficamente en la figura 2.1, a continuación se explica cada una de estas etapas.

- **Registro:** De su término en inglés *image-registration*, consiste en establecer la correspondencia geomántica entre las imágenes que componen la misma

escena. Para esto, es necesario estimar la transformación geométrica que logra alinear dichas imágenes en un mismo plano.

- **Alineación:** También llamada proyección, consiste en alinear las imágenes registradas en un sistema de referencia común, es decir, con respecto a un plano de referencia. En este caso se utiliza la transformación geométrica calculada en el paso anterior.
- **Fusión:** En este paso se busca corregir los errores fotométricos o discontinuidades, presentes en el mosaico luego del proceso de alineación. Estos errores aparecen, producto de errores en la estimación de las transformaciones o a cambios en la perspectiva de los objetos observados.

Si bien, se han propuesto una gran cantidad de algoritmos por parte de distintos grupos de investigación en todo el mundo, esta tarea aun sigue siendo desafiante, debido mayormente a los procesos de registro y fusión de las imágenes.

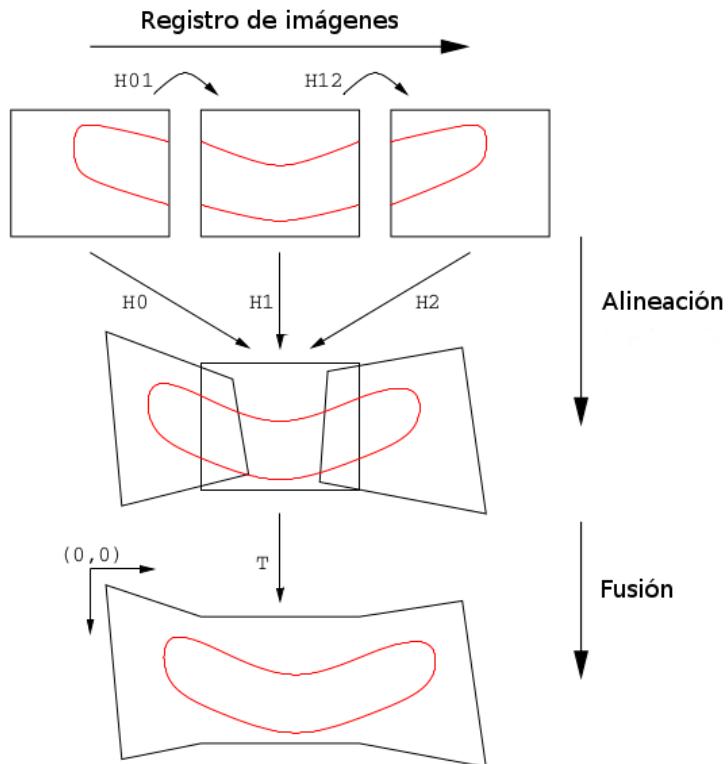


FIGURA 2.1: Proceso básico para la generación de mosaico, adaptado de [4]

Específicamente la etapa para estimar correspondencias entre las imágenes es un problema complicado, en principio debido a la naturaleza no plana de los suelos

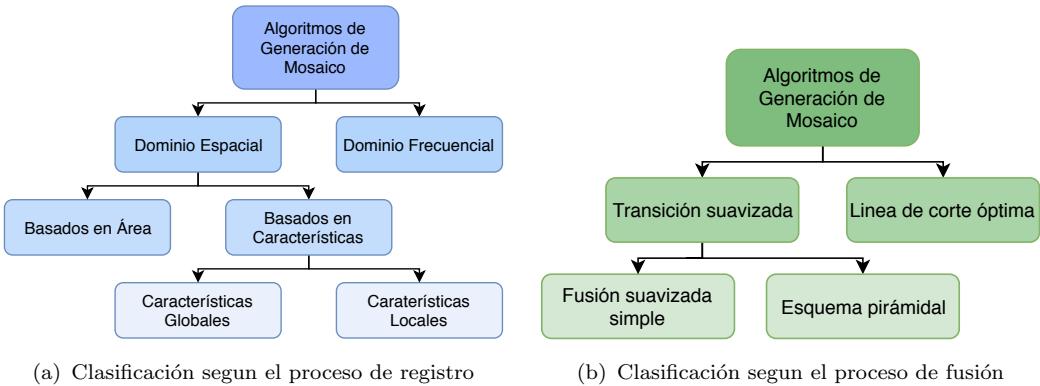


FIGURA 2.2: Clasificación de los algoritmos de generación de mosaico

estudiados; por otro lado la reducción de discontinuidades, o inconsistencias entre imágenes consecutivas sigue siendo una tarea desafiante, en este caso debido a fuertes cambios de exposición o errores en la etapa de registro. Es por esto que la mayoría de avances e implementaciones en esta área, están encaminados en resolver estos dos problemas principales, o bien mejorar los resultados de trabajos previos.

De esta forma se propone una clasificación de los algoritmos de mosaicos, basada en como estos abordan los procesos de registro y fusión. Además, para cada clasificación se realiza una breve revisión teórica de cada categoría, así como también los diferentes métodos y modificaciones que han aplicado diversos desarrolladores. en la figura 2.2 se puede apreciar esta clasificación.

Clasificación basada en el registro de imágenes

Este proceso es muy importante para la creación de mosaicos, y básicamente es la base para estos. Cuando se registran imágenes, primero lo que se busca es encontrar la relación, o la correspondencia entre estas, teniendo en cuenta que pudieron haber sido capturadas desde distintos puntos de vista, distintos instantes de tiempo, distinta perspectiva, o incluso distintas cámaras. Luego de encontrar las zonas o puntos correspondientes, se busca estimar una matriz de transformación geométrica que permita alinearlas todas en un sistema de referencia común. Se puede decir que el registro ha sido exitoso, si se logra estimar una matriz de transformación tal que todos los puntos correspondientes se puedan unir.

Las relaciones entre las imágenes se pueden establecer utilizando distintos métodos, ya sea, emparejando puntos coincidentes, regiones enteras, o bien usando la propiedad de correlación de fase en el dominio de la frecuencia. Si bien se cuenta con algoritmos de generación de mosaico que realizan el registro de imágenes basados en datos de navegación, como el GPS (del inglés: Global Positioning System), o instrumentos de medición inercial IMU (del inglés: Inertial Measurement Unit), para estimar el movimiento de los vehículos y así establecer una relación entre las imágenes basado en el movimiento de la cámara, en este trabajo solo se discutirán técnicas que utilicen únicamente una cámara monocular como sensor de entrada. Dicho esto, estos métodos para establecer correspondencias son discutidos a continuación.

Algoritmos en el dominio espacial

Los algoritmos en esta categoría utilizan la información de los píxeles para establecer la relación entre imágenes, es decir, se utiliza el valor de los píxeles (intensidad) y se trata de establecer la correspondencia de estos según la ubicación en la que se encuentran. Estos se pueden separar en dos técnicas principales: basados en área o en características.

Los algoritmos basados en área, buscan relacionar dos ventanas o regiones en dos imágenes que correspondan a la misma escena. El concepto principal consiste en mover la región de interés desde la primera imagen hacia la segunda, buscando que la diferencia entre las intensidades sea la menor posible, es decir, se trata de estimar la mejor matriz de transformación que logre reducir la diferencia de intensidades al alinear las regiones estudiadas [5]. Algunos trabajos importantes en esta clasificación utilizaron algoritmos como NCC (siglas del inglés: Normalized Cross Correlation) [6], y el MI (siglas del inglés: Mutual Information) [7], donde estos proporcionan una métrica de igualdad entre dos imágenes.

En el primer caso, el NCC mide la similitud entre las regiones estudiadas según los valores de intensidades, mientras que el MI la mide en base a la cantidad de información que comparten estas imágenes en términos de entropía. Al emplear esta técnica se logra emparejar las imágenes a nivel de píxel. Si bien se logran buenos resultados, este tipo de algoritmo requiere un alto nivel de superposición

entre las imágenes de entrada. Además, el proceso de iterar para optimizar los parámetros de transformación y el cálculo del error para cada píxel sobre las regiones, se convierte computacionalmente costoso.

Para reducir el tiempo de computo, se utilizan algoritmos basados en la relación de características, en los cuales se tratan de detectar puntos o regiones en distintas imágenes, que correspondan con la misma ubicación. Estas características detectadas se pueden evidenciar en forma de puntos aislados, curvas continuas o regiones conectadas. Luego se puede encontrar la transformación geométrica que relaciona las características de origen con las de destino, en muchos casos resolviendo un sistema de ecuaciones. En este caso, el proceso de registro así como también el resultado del mosaico, será tan bueno como el algoritmo de detección que se utilice.

Tal y como se mencionaron los tipos de características que se pueden extraer, podemos clasificar de forma general los algoritmos de detección. Ya sea si trabajan con características locales, como lo son aquello que detectan puntos aislados. o más globales como los basados en detección de contornos.

Detectores locales

Al usar este tipo de algoritmos se busca encontrar la relación entre una serie de puntos dispersos que se corresponden entre dos imágenes, donde las características locales más comunes que se suelen detectar serían esquinas, bordes, manchas, entre otros. Posteriormente el proceso de registro se completa al estimar la transformación geométrica con dicha relación de puntos, en este caso resolviendo un sistema de ecuaciones.

Una de las ventajas principales de esta técnica para la generación de mosaicos, es que puede trabajar con imágenes consecutivas que no posean un alto nivel de cobertura. Siempre y cuando la cantidad de puntos detectados, y correctamente emparejados entre el par de imágenes supere el mínimo necesario para la solución del sistema de ecuaciones.

Diversos algoritmos detectores de características locales se han venido desarrollando desde hace mucho tiempo [8–15], y en la actualidad estos avances han

permitido que el uso de este método traiga consigo muchas ventajas sobre el resto, desde variedad de aplicaciones, robustez ante distinto tipo de escenas, y velocidad de computo (siempre en función del detector a utilizar), lo que lo convierte en uno de los mas usados para la construcción de mosaicos.

Detectores globales

Al utilizar este tipo de detectores, se busca encontrar formas, contornos, texturas, o regiones sobresalientes que se mantengan invariantes ante cambios del punto de vista o iluminación [16]. Al igual que con los detectores locales, aquí se busca extraer tanto la posición, como el tamaño y la orientación de estas regiones.

El resto del proceso para completar la etapa de registro se mantiene igual con este método, donde la transformación geométrica se obtiene a partir de la correspondencia entre las posiciones y orientaciones de las regiones de interés que se lograron extraer. Si bien tienen buen rendimiento ante cambios de movimiento desafiantes, su uso implica un aumento en el tiempo de computo.

Algoritmos en el dominio frecuencial

Ya se ha visto que los algoritmos que operan en el dominio espacial cubren la mayoría de las aplicaciones e investigaciones. Sin embargo se pueden encontrar métodos que obtienen los parámetros óptimos para la transformación a partir de cálculos en el dominio de la frecuencia.

Estos algoritmos utilizan la propiedad de correlación de fase para lograr su objetivo. Teniendo un par de imágenes que se encuentran relacionadas por una simple traslación, el funcionamiento consiste en calcular la correspondiente transformada de *Fourier*, luego el espectro de la potencia cruzada entre ambas. De aquí, se asegura que la fase del espectro de la potencia cruzada corresponde con la diferencia de traslación entre las dos imágenes. Finalmente el proceso continua similarmente a los métodos anteriores, alineando las imágenes según la transformación obtenida, seguido del proceso de fusionarlas.

Tal y como se explicó este proceso para una traslación, se tienen diversos trabajos como [17] que añaden modificaciones para permitir otro tipo de transformaciones como la rotación, e incluso otros que admiten cambios en la escala [18]. Si bien, los trabajos mencionados presentaron importantes, se requiere de un buen porcentaje de cobertura entre las imágenes, y además presentan limitaciones en los grados de libertad de las transformaciones geométricas que se pueden estimar.

Clasificación basada en la fusión de imágenes

Si bien el proceso de registro de imágenes es fundamental para lograr un mosaico correcto, el paso final de unir las imágenes también es de gran importancia. Teniendo en cuenta que se busca aparentar que todas las imágenes componen una sola, es vital que se logre un mapa final sin inconsistencias o discontinuidades producto de los cambios en iluminación, objetos en movimiento, entre otros.

Debido a la importancia de este paso, numerosos métodos para lidiar con este tipo de problemas se han desarrollado, lo que nos permite también clasificar los algoritmos de generación de mosaico según como aborden este problema. Entre los métodos mas utilizados encontramos: fusionar las imágenes mediante cambios suavizados o búsqueda de la mejor linea de corte.

Transición suavizada

Los algoritmos de esta categoría tratan de minimizar la diferencia entre dos imágenes, buscando que el cambio entre los bordes de estas sea imperceptible. El método mas simple para fusionar dos imágenes bajo esta técnica, consiste en realizar una suma ponderada sobre el área de superposición entre ambas, ponderando la intensidad de cada imagen a la mitad. Al realizar esta operación se suelen tener efectos indeseados como el efecto fantasma, en el cual se pueden observar duplicados del mismo objeto con cierto nivel de desvanecimiento. Esto de sebe a errores en el proceso de alineación de las imágenes, diferencia en la iluminación, o incluso a objetos móviles.

Para evitar esto, se utiliza un proceso de fusión que consiste en realizar una suma ponderada entre ambas imágenes, pero dando mayor peso a las regiones que se encuentren mas cerca del centro de la imagen, y menor a aquellas que se encuentren cerca el borde. Si bien, se logra reducir posibles discontinuidades entre los bordes originales, el efecto fantasma aun se puede apreciar para imágenes con fuertes problemas de alineación.

Considerando este problema, y con el objetivo de realizar una unión mas robusta, se desarrolló un esquema piramidal de fusión ponderada. El proceso consiste en obtener una imagen laplaciana para distintos tamaños de escala, formando así una pirámide, al mismo tiempo se va creando para cada nivel de la pirámide una mascara difuminada por el efecto del filtro gaussiano de dicha escala. Luego para cada nivel de la pirámide se aplica el algoritmo de fusión ponderada descrito previamente donde la mascara difuminada pondera el valor de cada píxel. Entre los trabajos que aplican este algoritmo obteniendo resultados notables se tiene [19], logrando reducir en gran medida el efecto duplicado en las regiones de superposición.

Línea de corte óptima

En lugar de buscar reducir las posibles discontinuidades a partir de una suave transición a través del borde entre dos imágenes, tal y como lo hacen los algoritmos previos, en este tipo de algoritmos se busca modificar este borde. Es decir, se trata de encontrar la linea de corte en el área de superposición que logre reducir la discontinuidad de texturas entre ambas imágenes. La diferencia principal de este método sobre los anteriores, es que se toma en cuenta la información presente en la región que se desea fusionar, permitiendo que se logre remover errores producidos por el efecto paralaje, o debido a objetos móviles dentro de la escena. Por otra parte, como las regiones resultantes luego de la linea de corte no se comparten información, se pueden presentar discontinuidades producto de grandes diferencias de iluminación.

En esta sección podemos destacar el trabajo en [20], que presenta resultados robustos ante imágenes con alto efecto paralaje. En este se busca el área para el cual ambas imágenes presentan mejor similitud —llamada área de costura-local—,

y se busca una linea de corte que bordee esta región. Por otro lado tenemos trabajos como el presentado en [21], donde se busca la linea de corte que minimiza la diferencia entre los gradientes en el área de superposición, a través de un modelado por grafos, y resolviendo el algoritmo flujo-máximo/mínimo-corte [22] para encontrar la línea deseada.

2.2. Esquema propuesto

En base a los métodos estudiados en la sección anterior, se plantea el siguiente esquema para la construcción automática de mosaicos:

En primer lugar, se propone utilizar un esquema basado en características para la etapa de registro de imágenes, específicamente, utilizar algoritmos detectores de características locales. El uso de esta técnica permite que se pueda trabajar con imágenes consecutivas que no tengan un elevado nivel de superposición. Además, la gran variedad de algoritmos presentes de esta categoría permite trabajar bajo un gran número de ambientes, presentando robustez ante variados tipos de condiciones.

Por otro lado, evaluando visualmente los resultados obtenidos en las distintas técnicas de fusión, se pretende implementar una combinación en los algoritmos que fueron mas eficientes bajo condiciones de alto efecto paralaje, presencia de objetos móviles, cambios en la iluminación, entre otros. En este caso, se buscaría la linea de corte óptima y luego se aplica una fusión bajo un esquema piramidal.

Si bien el registro y la fusión constituyen las etapas mas importantes, diversos métodos para lograr corregir errores de distorsión y proyección se han desarrollado de la mano de muchos grupos de investigación. Evaluando entre los presentes, se plantea implementar el modelo de sub mosaicos propuesto por *F. Bellavia et. al.* en [23, 24], el cual se presentará detalladamente en el siguiente capítulo. Explicado brevemente, estos algoritmos buscan reducir errores provenientes de la sección de registro, y toman en consideración algunos aspectos importantes que logran reducir distorsiones geométricas en el mosaico final.

Finalmente, tomando en cuenta algunas observaciones y recomendaciones de los trabajos consultados, se propone implementar un módulo de corrección de color con el objetivo de reducir las diferencias de intensidades entre los bordes de las imágenes consecutivas. Adicionalmente en el esquema se agregan algunas técnicas propuestas en [25] para añadir robustez al sistema propuesto. El esquema propuesto se puede observar en la figura 2.3, donde cada uno de estos módulos serán explicados en detalle en los capítulos siguientes.

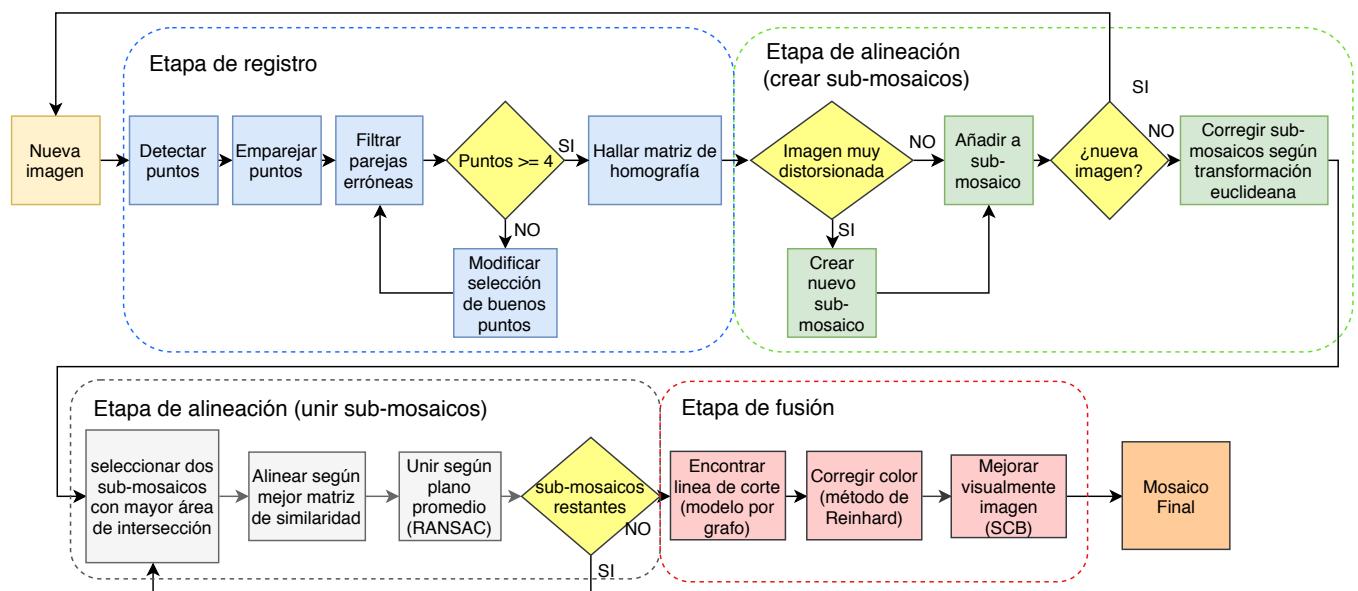


FIGURA 2.3: Esquema propuesto para la construcción del mosaico

2.3. Librería de desarrollo

2.3.1. OpenCV

Para la implementación de los módulos previamente descritos, es necesario el uso de librerías y entornos de trabajo, que permitan un manejo eficiente de las imágenes. Además, el uso de plataformas que se encuentren estandarizadas en esta área de estudio, facilita que el desarrollo del presente trabajo siga avanzando de la mano de futuros desarrolladores. En base a esto, se seleccionó la librería OpenCV¹ para la implementación de los módulos necesarios en el sistema de generación de mosaico.

¹<http://opencv.org/>

OpenCV (del inglés: *Open Source Computer Vision*) es una librería de procesamiento de imágenes desarrollada por la empresa Intel² en el año 1999. Esta librería ofrece un gran numero de algoritmos optimizados (actualmente mas de 2.500), el cual proporciona un entorno de desarrollo altamente eficiente para aplicaciones de procesamiento de imágenes. Asimismo presenta una gran aceptación por parte de los usuarios en el mundo académico y comercial, con mas de 47 mil usuarios activos, y un numero de descargas que supera los 14 millones.

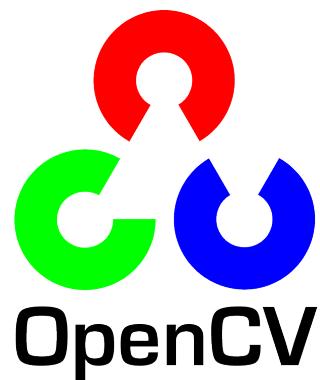


FIGURA 2.4: Logo de la librería OpenCV

Esta plataforma tiene soporte para distintos sistemas operativos, como *Windows, Linux, Mac OS, iOS y Android*. Además de tener la posibilidad de trabajarla con diversos lenguajes de programación como: C++, Python, JavaScript. La motivación de el presente trabajo se encuentra orientada al desarrollo de una aplicación que en un futuro pueda ser embebida en un sistema de navegación automático, con lo cual el soporte de un lenguaje de bajo nivel como C++, puede permitir el desarrollo de un algoritmo con suficiente velocidad de computo para este fin.

Por otra parte, esta librería presenta soporte para trabajar con la arquitectura de cálculo paralelo *CUDA* (del inglés: *Compute Unified Device Architecture*) de la empresa NVIDIA³, con la cual se puede aprovechar el uso de las unidades de procesamiento gráfico para acelerar el rendimiento del algoritmo que se implemente. Cabe destacar que los equipos presentes en el laboratorio del *GIDM* tienen disponible tarjetas gráficas con este soporte.

²<http://www.intel.com>

³<http://www.nvidia.com>

Capítulo 3

Extracción de puntos característicos

3.1. Introducción

En el capítulo anterior se valuaron los distintos tipos de técnicas de establecer la relación para el registro de imágenes, y en base a estos se planteó el uso de características locales para establecer la correspondencia entre imágenes. En el presente capítulo se pretende explicar en detalle el funcionamiento de los algoritmos de detección mas usados en estas aplicaciones, así como también la importancia de los avances que cada uno introduce. Después se presentan los algoritmo de emparejamiento, así como también una explicación de su funcionamiento. Finalmente, con el objetivo de caracterizar cada uno de los detectores y emparejadores, se implementa un módulo para comparar el rendimiento de estos bajo distintas condiciones. Esto nos permitirá luego seleccionar el mejor algoritmo para cada aplicación.

3.2. Revisión teórica

3.2.1. Detectores y descriptores de características

Antes mencionar la evolución de los algoritmos de detección de puntos de interés, primero es necesario definir que son estos. Los puntos de interés, puntos clave, o *"features"* (en español: características) como son comúnmente llamados, son regiones en una imagen que contienen patrones específicos, lo que hace que puedan ser fácilmente seguidos o ubicados en otra imagen. Tuytelaars y Mikolajczyk [26] definen un punto característico local como *"un patrón en la imagen que difiere de su vecindario directo"*. De esta forma, se considera que los puntos característicos deben proporcionar la posibilidad de ser identificados en diferentes imágenes con el objetivo de emparejarlos.

Para alcanzar este objetivo los detectores y extractores de puntos característicos deben cumplir con ciertas propiedades que les permita funcionar bajo distintas condiciones, en concreto se busca que estos algoritmos cumplan con las siguientes propiedades:

- **Robustez:** El algoritmo debe ser capaz de detectar la misma ubicación del punto característico independientemente ante cambios en la escala, rotación, traslación, iluminación, transformaciones geométricas, artefactos de compresión y ruido.
- **Repetibilidad:** El algoritmo debe ser capaz de detectar el mismo punto característico de la misma escena bajo cambios en el punto de vista.
- **Exactitud:** El detector debe localizar el punto característico de manera precisa (misma ubicación de píxel). Especialmente para tareas de alineación de imágenes.
- **Generalidad:** El algoritmo debe ser capaz de detectar puntos que pueden ser usadas en distintas aplicaciones, es decir, que detecte varios tipos de características (esquinas, burbujas, etc.)

- **Eficiencia:** El algoritmo debe ser capaz de detectar puntos característicos en nuevas imágenes a gran velocidad, para soportar aplicaciones en tiempo real.
- **Cantidad:** El algoritmo debe detectar todos, o casi todos los puntos característicos presentes en la imagen.



FIGURA 3.1: Caracterización de regiones en una imagen¹

Llegados a este punto es necesario definir el funcionamiento de los algoritmos detectores, y que consideran estos como puntos característicos, basados en la definición previamente planteada. Atendiendo a la imagen 3.1, se puede observar que se caracterizan seis áreas de interés. Analizando estos segmentos, vemos que **A** y **B** corresponden con superficies planas, lo que hace que sea muy difícil identificar la ubicación exacta de estas superficies en la imagen original. Por otro lado, tenemos las regiones **C** y **D**, las cuales corresponden con bordes en la imagen, si bien, se puede limitar en gran medida el área de búsqueda hacia toda las regiones del mismo bordes, sigue siendo difícil acertar con la ubicación correcta. Por ultimo, analizando las regiones **E** y **F** tenemos que corresponden a esquinas de la imagen original, en este caso se puede identificar fácilmente la ubicación exacta de la región en la imagen.

A partir de esta idea, en la cual se consideran las esquinas como regiones fácilmente identificables en una imagen, en 1988 nace el primer algoritmo de detección

¹https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_features_meaning/py_features_meaning.html

de puntos de interés llamado Detector de esquinas de Harris [8] (nombre original en inglés: Harris Corner Detector), y como su nombre lo indica está basado en la detección de esquinas.

Retomando el concepto planteado previamente, este detector busca la diferencia de intensidad de una región con su entorno directo, es decir, se detectará una esquina para aquellas regiones que presenten una alta variación de intensidad, al desplazar la ventana estudiada en cualquier dirección. En la figura 3.2 se puede apreciar visualmente como funciona esta ventana de búsqueda.

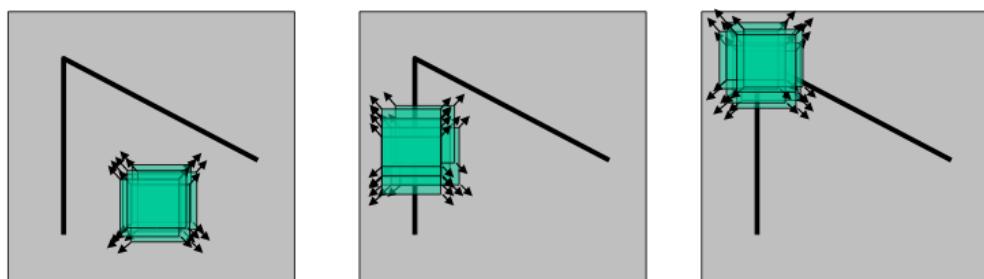


FIGURA 3.2: Ventana de búsqueda para de detección de esquinas, adaptado de²

Cuando se trabajan con detectores de características, se desea que estos sean invariantes ante la mayor cantidad de variables posibles, tal y como se menciona en la propiedad de robustez que deben tener estos algoritmos. Si bien el detector presentado anteriormente es invariante ante la traslación y la rotación (ya que las esquinas se mantienen como esquinas si son rotadas o desplazadas), no funciona de la misma forma ante cambios de escala. Como se observa en la figura 3.3, una región considerada como esquina, se podría considerar plana si es ampliada.

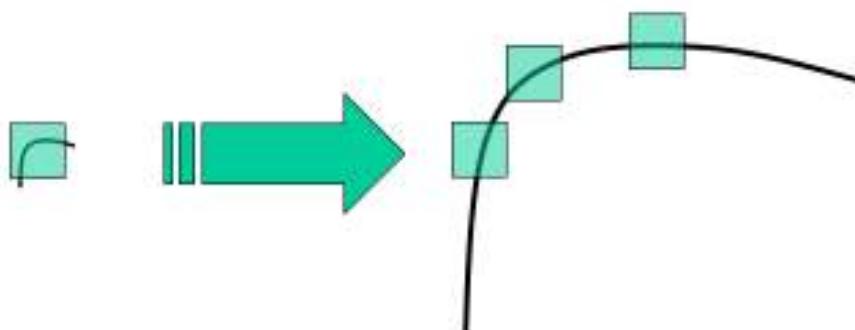


FIGURA 3.3: Efecto del escalado sobre las esquinas³

²<https://dsp.stackexchange.com/questions/14338/>

Con el fin de conseguir detectar los mismos puntos ante cambios en la escala de la imagen, Lindeberg, T. [27] propone un algoritmo detector de “manchas” multi-escala a través de la búsqueda de un máximos en el espacio de escala, el cual se crea utilizando un operador laplaciano. El Laplaciano de Gaussianas *LoG* (del inglés: Laplacian-of-Gaussian), es una combinación lineal de segundas derivadas utilizado para detectar burbujas o manchas en una imagen. El funcionamiento es el siguiente: Dada una imagen de entrada, la representación para cada escala $-s-$ de la imagen se define como la convolución de la imagen con un filtro Gaussiano con desviación estándar s .

Este resultado brinda una fuerte respuesta positiva para burbujas oscuras y respuestas fuertes negativas para burbujas claras, ambas de un tamaño $2s$, donde s es la escala. De esta forma las características detectadas presentan una fuerte relación entre el tamaño de las estructuras en la imagen y el grado de difusión del filtro gaussiano. Donde la desviación estándar del filtro se usa para controlar la escala cambiando que tanto se difumina la imagen.

Llegados aquí, una vez se haya detectado la ubicación de los puntos característicos en la imagen, la información de la localidad de este debe ser codificada y almacenada, y de esta forma lograr tener un descriptor único de la región con el objetivo final de ubicarlo en otra imagen. Con este fin se desarrollaron los algoritmos descriptores, los cuales una vez tengan la ubicación de los puntos característicos se encargan de convertir la información de su alrededor en una serie de números, o un vector que permita diferenciar un punto clave de otro. Esta información también es necesaria que sea invariante ante las variables mencionadas previamente, para lograr una identificación eficiente del mismo punto en distintas imágenes bajo distintas condiciones.

Partiendo de estos problemas, y del hecho que el cálculo del operador *LoG* es computacionalmente costoso, en 2004 *D. Lowe* crea el detector y descriptor *SIFT* [9] (del inglés: Scale Invariant Feature Transform), en el cual el espacio de escala es construido en forma piramidal con la diferencia de gaussianas *DoG* (del inglés: Difference of Gaussians). En este sentido, El operador *DoG* ofrece una aproximación al *LoG*, donde se calcula sin convolución restando niveles de escala

³https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html

adyacentes de una pirámide gaussiana. El proceso para la detección y descripción de puntos de interés de este algoritmo, consta de cuatro pasos principales:

En primer lugar, realiza una detección de máximos en el espacio de la escala aplicando la diferencia gaussiana *DoG*. Para esto, se aplica el filtro gaussiano con distintos tamaños de media (se tienen distintas escalas), luego restando estas imágenes para distintos pares de escalas se logra la diferencia de gaussiana. Posteriormente se buscan los máximos locales a lo largo del espacio (coordenadas X,Y) para cada correspondiente escala. Este proceso de detección se puede visualizar en la figura 3.4.

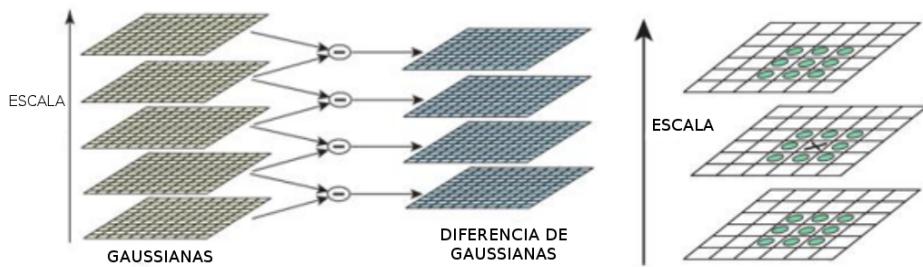


FIGURA 3.4: Detección de máximos en el espacio de la escala DoG, adaptado de [9]

En segundo lugar para la localización de puntos de interés, se descartan los puntos encontrados en el paso anterior que no superen cierto valor de umbral, es decir, que no estén lo suficientemente contrastados con su entorno. Con esta etapa el algoritmo solo toma en cuenta los puntos claves mas fuertes por cada escala. Además, con el objetivo de eliminar los bordes suficientemente contrastados que no correspondan con esquinas, el algoritmo usa una matriz hessiana para calcular las curvaturas principales, y así quedarse solo con esquinas.

Para garantizar la invarianza con respecto a la rotación, se toman los píxeles vecinos al punto clave y se calcula la magnitud y dirección del gradiente en esa región. Con esto se hace un histograma de la magnitud del gradiente en cada dirección, donde el pico mayor del histograma indica la orientación. En el caso que exista un pico mayor al 80 % del pico principal, este se utiliza para crear otro punto de interés en la misma posición pero con la distinta rotación.

Finalmente para crear el vector descriptor por cada punto clave se crea una matriz de 16x16 alrededor de este, dividida en 4 subregiones de 4x4 píxeles con un

histograma de orientaciones para cada uno. Seguidamente, el descriptor del punto será el vector con los valores de los histogramas de las regiones 4x4 concatenados. La figura 3.5 la representación del descriptor de SIFT.

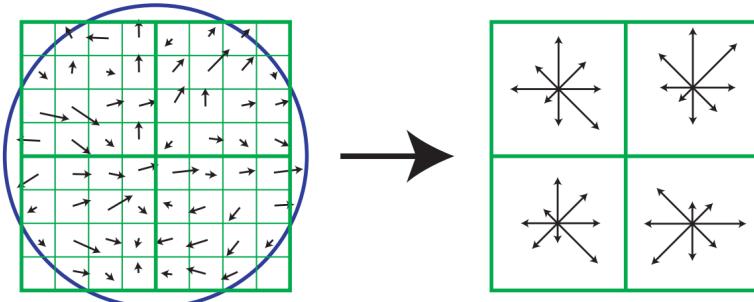


FIGURA 3.5: Izquierda: imagen de gradientes. Derecha: descriptor del punto clave. de [9]

En el año 2006, un grupo de tres personas Bay, H., Tuytelaars, T. and Van Gool, L. desarrollan *SURF* [10], el cual es un detector y descriptor de características basado en SIFT, pero con modificaciones que aumentan su velocidad de detección. Si bien, sacrifica un poco de rendimiento y precisión, lo hace mas provechoso para aplicaciones embebidas que demanden mayor velocidad de computo y menor uso de recursos, como por ejemplo *SLAM*. El proceso para la extracción de características por parte de este algoritmo se compone de los siguientes pasos:

Como primer paso, en lugar de aproximar el laplaciano de Gauss *LoG* (del inglés: Laplacian of Gaussians) con la diferencia de Gaussianas (DoG) como lo hace SIFT, este algoritmo aproxima LoG con cuadrados para promediar la imagen. La ventaja de aplicar filtros con cuadrados es que con la ayuda de imágenes integrales el cálculo computacional se reduce en gran medida.

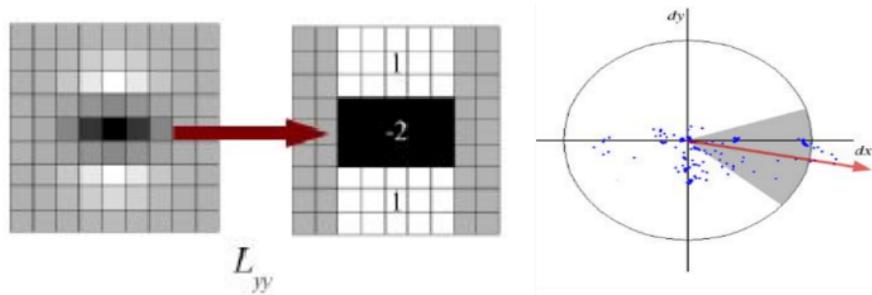


FIGURA 3.6: Izquierda: aproximación a la derivada de segundo orden del filtro gaussiano (derivada parcial en el eje y) y su aproximación con un filtro cuadrado. Derecha: vector de orientación del descriptor. Adaptado de [10]

En función de identificar la orientación, el algoritmo utiliza la respuesta wavelet Haar en horizontal y vertical en un vecindario de $6s$ (donde s es la escala evaluada) píxeles al rededor del punto de interés. Luego estas respuestas son representadas como puntos en el espacio, para luego calcular la orientación dominante con la suma de todos los resultados dentro de una ventana deslizante de apertura 60° . En la figura 3.6 se puede visualizar en el lado izquierdo, la aproximación que realiza de la derivada de segundo orden del filtro gaussiano, y su aproximación con un filtro cuadrado. Del lado derecho se ilustra el vector de orientación en función a la distribución de puntos estudiados.

El siguiente avance importante en los algoritmos de detección aparece en el año 2011 con *ORB* [13] (del inglés: Oriented FAST and Rotated BRIEF), este utiliza una combinación del detector FAST (del inglés: Features from Accelerated Segment Test) y del descriptor BRIEF (del inglés: Binary Robust Independent Elementary Features), este nuevo algoritmo esta caracterizado por su alta velocidad de procesamiento manteniendo un buen rendimiento, gracias al uso de un descriptor binario.

Como se mencionó utiliza el algoritmo FAST, el cual consiste en encontrar esquinas evaluando los píxeles en un perímetro circular, de esta forma, un punto será detectado como esquina si la cantidad de píxeles de color opuesto al evaluado, supera cierto valor de umbral (ver izquierda en la figura 3.7), posteriormente con el fin de aumentar la robustez, es aplicado el algoritmo de clasificación de esquinas de *Harris*. De igual forma se realiza con una estructura piramidal evaluando varias escalas (al igual que SIFT).

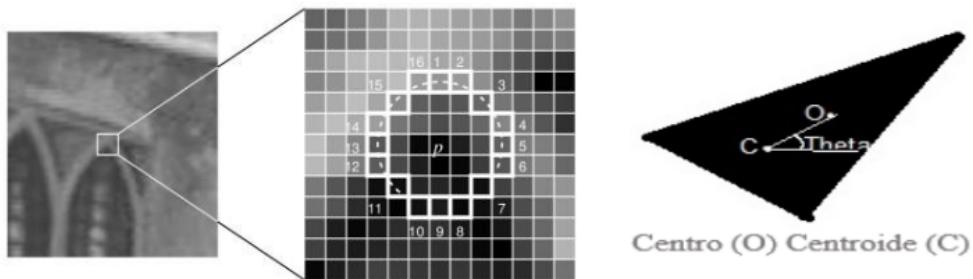


FIGURA 3.7: izquierda: detección de esquinas usando FAST, de [11]. derecha: descriptor basado en BRIEF, adaptado de⁴

⁴<https://gilscvblog.com/2013/10/04/>

Como el algoritmo FAST no toma en cuenta la orientación, en el ORB se modificó para que calculara la orientación de la siguiente forma: Se considera una región ubicada en el centro del punto estudiado, luego se calcula el centroide de la región en función a la intensidad de los puntos. De esta forma, la dirección del vector desde el punto central hasta el centroide es asignado como vector de orientación. Observando a la derecha en 3.7 se aprecia un ejemplo del lugar del centroide (C) y del centro (O) para una región en particular.

Para el descriptor utiliza BRIEF, a diferencia de los anteriores (SIFT y SURF) este es un descriptor binario y no vectorial. El descriptor BFIEF produce una palabra de n -bits usando el algoritmo *Local Binay Tests* (LBT), el problema de esta representación es que no es muy robusta ante cambios en la rotación. Para resolver esto ORB utiliza la información de la orientación previamente calculada en el paso de detección para aplicar LBT en esa orientación.

Los algoritmos de detección que se mencionaron hasta este momento tienen una característica en común, y es que cuando trabajan con el esquema piramidal lo hacen bajo el espacio de escala Gaussiano, el cual es una instancia particular de difusión lineal. De esta forma, al utilizar este filtro no se respetan los límites naturales de los objetos y se difumina del mismo nivel toda la región de la imagen cuando se avanza entre niveles de escala.

Enfocándose en esta característica, en el año de 2012 se desarrolla el detector y descriptor llamado KAZE [14] por parte de *Pablo Fernández Alcantarilla*. Este novedoso algoritmo opera completamente en un espacio de escala no lineal, y para ello utilizan un esquema de división de operadores aditivos (*AOS*, del inglés: Additive Operator Splitting), que les permite obtener espacios de escala no lineales de forma eficiente. De este modo se puede realizar un difuminado localmente adaptativo, posibilitando que se remueva el ruido en las imágenes, manteniendo información importante sobre los bordes de los objetos al avanzar en el espacio de escala. En la figura 3.8 se puede observar como afecta en los bordes de los objetos el aplicar un filtro de difusión lineal, y uno que no lo es, bajo el esquema propuesto por este algoritmo.

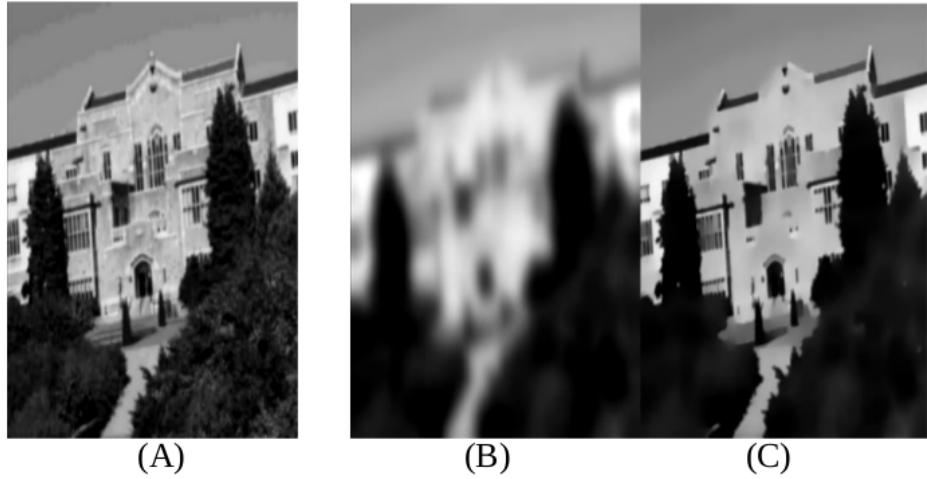


FIGURA 3.8: (A): imagen original, (B) filtro lineal Gaussiano, (C) filtro no lineal propuesto en KAZE, adaptada de [14]

Bajo este mismo esquema de difusión no lineal, el mismo autor en el año 2013 desarrolla la versión acelerada de este algoritmo que recibe el nombre de *A-KAZE* [15] (del inglés: Accelerated KAZE). Esta mejora se utiliza un esquema basado en difusión explícita rápida *FED* (del inglés: Fast Explicit Diffusion) en lugar de *AOS*, el cual es un nuevo esquema piramidal que incrementa en gran medida la velocidad de computo para construir el espacio de escala no lineal.

Para el cálculo de la orientación el primer algoritmo *KAZE* utiliza un descriptor para la orientación similar al que emplea *SURF*. Este encuentra la orientación dominante en un área circular de radio $6s$ (s corresponde con la escala), y para cada muestra del círculo se calcula la derivada de primer orden en las direcciones X e Y , y se ponderan con una gaussiana centrada en el punto de interés. Luego, las respuestas de estas derivadas son representadas como puntos en un espacio vectorial, donde la orientación dominante se haya sumando las respuestas dentro de un segmento de círculo deslizante con apertura de 60° .

Por otro lado, la versión acelerada *A-KAZE* emplea un descriptor basado en una versión modificada del algoritmo de diferencia local binaria *LDB* [28] (del inglés: Local Difference Binary), llamado M-LBD (del inglés: Modified Local Difference Binary), el cual aprovecha al máximo la información del espacio de escala no lineal. La modificación consiste en hacer un sub-muestreo de cada región que divide la zona del descriptor, en lugar de calcular el promedio de todos los píxeles de la región, es decir, se tienen muestras de cada subdivisión para distintas escalas.

3.2.2. Emparejadores de puntos característicos

En este punto ya hemos estudiado los distintos algoritmos que permiten encontrar y clasificar puntos de interés en las imágenes. Ahora bien, es necesario identificar cuales de estos puntos corresponden con la misma ubicación, tal y como podemos observar en la figura 3.9. Para establecer esta relación se utilizan los algoritmos emparejadores de características, los cuales relacionan estos puntos en base a los descriptores de dichos puntos.

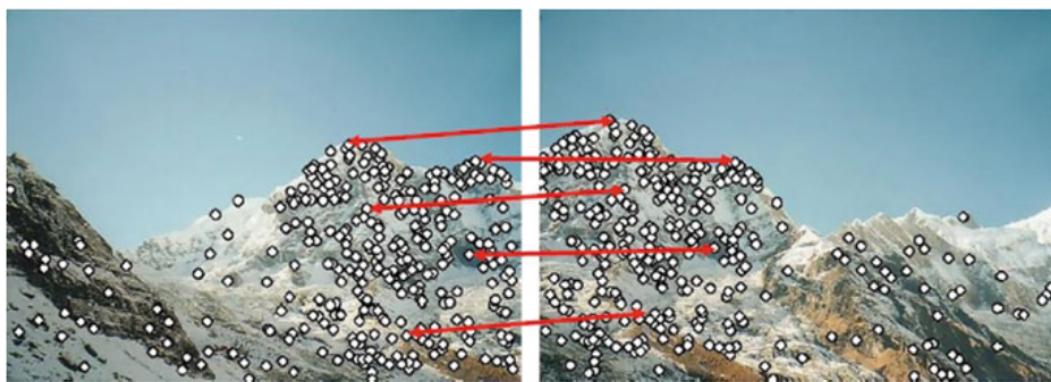


FIGURA 3.9: Emparejamiento de puntos que corresponden a la misma ubicación, de [29]

El proceso para realizar el emparejamiento consiste en el siguiente: Teniendo un punto característico P_1 perteneciente a la imagen 1, y por otro lado se teniendo un punto P_2 perteneciente a la imagen 2, Se calcula la distancia entre los descriptores D_1 y D_2 . En el caso de descriptores vectoriales, esta distancia corresponde con la distancia *Euclídea*, dada por la siguiente expresión:

$$D = \sqrt{(v_1 - q_1)^2 + (v_2 - q_2)^2 + \cdots + (v_n - q_n)^2}$$

Donde v_n corresponden con los componentes del vector descriptor $D1$, q_n con los componentes del vector descriptor $D2$, y n en el numero de componentes de ambos vectores. Por otro lado, para los descriptores binarios, se calcula la distancia *Hamming*, dada por la siguiente expresión:

$$D = ||D_1 \oplus D_2||$$

Este proceso se repite hasta tener la distancia de cada punto de la imagen 1 con todos los puntos de la imagen 2, y viceversa. Al tener estas distancias, los puntos se emparejarán si y solo si se cumplen las siguientes condiciones:

- (I) El punto P_1 presenta la mejor distancia con P_2 , en relación a todos los puntos de la imagen 2.
- (II) El punto P_2 presenta la mejor distancia con P_1 en relación a todos los puntos de la imagen 1.

Este proceso es mejor conocido como emparejamiento por fuerza bruta, ya que se compara entre todos los puntos por la mejor pareja posible. Si bien se asegura obtener el mejor emparejamiento, siendo viable para trabajar con pocos datos, el hecho de probar todas las posibles cuando se tiene una gran cantidad de puntos, incrementa en gran medida el tiempo de computo. Para efectuar este proceso de una forma eficiente se desarrollaron algoritmos basados en la búsqueda de vecinos más cercanos. En este sentido se cuenta con el algoritmo *kd-forest* (abreviado del inglés: k-dimensional forest), el cual es una mejora del algoritmo *kd-tree* (abreviado del inglés: k-dimensional tree) para mejor desempeño al usar vectores multidimensionales, implementado en la librería para la rápida aproximación de vecinos más cercanos FLANN [30] (del inglés: Fast Library for Approximate Nearest Neighbors).

3.3. Módulo comparativo

Una vez se conocen los algoritmos que se proponen utilizar, es necesario comparar su rendimiento bajo distintas condiciones, de modo que se pueda seleccionar el indicado para cada tipo de aplicación. En este sentido, se pretende estudiar el rendimiento en base a los siguientes parámetros:

- **Tiempo de ejecución:** Tiempo en el que se detectan y describen las características en dos imágenes con las mismas dimensiones.
- **Cantidad de puntos detectados:** Cantidad total de puntos detectados en dos imágenes.

- **Cantidad de puntos emparejados:** Cantidad total de puntos emparejados correctamente luego de descartar parejas erróneas.

Adicionalmente, se propone aplicar algunos métodos que permiten aumentar el rendimiento de los algoritmos de detección, así como también implementar un proceso que permita reducir o eliminar falsos positivos al emparejar puntos de interés.

3.3.1. Metodología

El proceso planteado para la comparación, consiste en evaluar los parámetros antes mencionados para todas las combinaciones de extractores y emparejadores que se tienen, tal y como se ilustra en la figura 3.10.

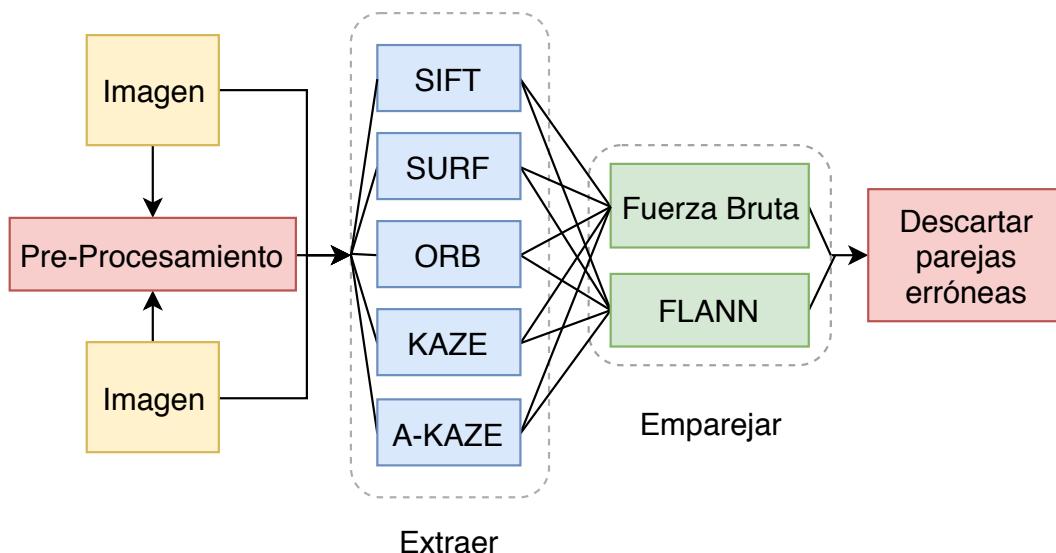


FIGURA 3.10: Combinaciones posibles del módulo comparativo

Cuando se utiliza un algoritmo para emparejar características, es muy común que existan parejas erróneas, puesto que al tener una gran cantidad de datos, varios pares de descriptores pueden tener la similitud necesaria para ser considerados como el mismo punto. Por esta razón es importante emplear una etapa que permita filtrar dichas parejas. Como ya se mencionó, se tienen distintos tipos de emparejadores, y en función a cada uno es necesario realizar el descarte de manera distinta. A continuación se describe el proceso en función a cada caso.

En el caso del algoritmo de fuerza bruta: Se obtiene la distancia de la mejor pareja, luego se descartan todas las parejas cuya distancia sea mayor que la mejor obtenida multiplicada por un factor de umbral. El proceso planteado se muestra en el algoritmo 1.

Algoritmo 1: Selección de buenas parejas - Fuerza bruta

```

1  $P_i \equiv i$ -ésima pareja
2  $D_i \equiv$  Distancia de la  $i$ -ésima pareja
3  $N \equiv$  Número de puntos emparejados
4  $U \equiv$  Umbral para descartar erróneos
5 inicio
6    $U = 0,8;$ 
7   para cada  $i \in n=1,2,\dots,N$  hacer
8     si  $D_i < mejorDistancia$  entonces
9        $mejorDistancia = D_{p_i};$ 
10      fin
11    fin
12    para cada  $i \in n=1,2,\dots,N$  hacer
13      si  $D_i > mejorDistancia \cdot U$  entonces
14        eliminar  $P_i;$ 
15      fin
16    fin
17 fin

```

En el caso del algoritmo bajo el esquema de vecinos mas cercanos: Se descarta cada pareja cuya distancia esté muy cercana a la distancia del vecino mas próximo. El proceso planteado se muestra en el algoritmo 2.

Algoritmo 2: Selección de buenas parejas - Vecinos mas cercanos

```

18  $P_i \equiv i$ -ésima pareja
19  $D_i \equiv$  Distancia de la  $i$ -ésima pareja
20  $V_i \equiv$  Distancia del vecino mas cercano de la  $i$ -ésima pareja
21  $N \equiv$  Número de puntos emparejados
22  $U \equiv$  Umbral para descartar erróneos
23 inicio
24    $U = 0,5;$ 
25   para cada  $i \in n=1,2,\dots,N$  hacer
26     si  $D_i > V_i \cdot U$  entonces
27       eliminar  $P_i;$ 
28     fin
29   fin
30 fin

```

En segundo lugar, con el objetivo de mejorar la detección se implementa un proceso de pre-procesamiento en las imágenes de entrada. Tal y como se estudió en la sección teórica, para que un punto característico sea detectado, éste debe contener información que lo diferencie de su entorno cercano, en este caso esta diferencia es medida en función a su intensidad. En base a esta definición, se propone un algoritmo que permite mejorar el contraste en la imagen mediante la técnica de estirar su histograma. De esta forma cada imagen tendrá el máximo rango de excursión sobre las intensidades de sus puntos, permitiendo que los valores de umbral al detectar regiones, se superen con más facilidad.

Es importante destacar que para este proceso todas las imágenes fueron almacenadas usando variables de 8 bits, es decir, que cada valor es representado en un rango comprendido entre 0 y 255.

Para el cálculo del histograma se obtiene la frecuencia de aparición de cada valor de intensidad en la imagen. Luego, se obtiene el valor de la intensidad para la cual, la cantidad de píxeles cuyo valor sea menor o igual a ésta, no supere el 1% de la cantidad total de píxeles en la imagen, este punto es llamado percentil bajo. A continuación se repite este proceso para ubicar el percentil alto, el cual corresponde con la intensidad para la cual la cantidad de píxeles inferiores a ésta, no supera el 99% de la cantidad total de píxeles. Al tener estos valores se aplican los siguientes criterios:

- Todos los píxeles cuyo valor se encuentre por debajo del percentil bajo es igualado a 0.
- Todos los píxeles cuyo valor se encuentre por encima del percentil alto es igualado a 255.
- Todos los valores cuyo valor se encuentre entre el percentil bajo y el alto, es reescalado usando la siguiente ecuación:

$$I_{pixel} = \frac{(I_{pixel} - P_{bajo}) \cdot 255}{P_{alto} - P_{bajo}}$$

En la figura 3.11 se puede apreciar el cambio, tanto en el histograma como en la misma imagen, luego de aplicar el proceso previamente descrito.

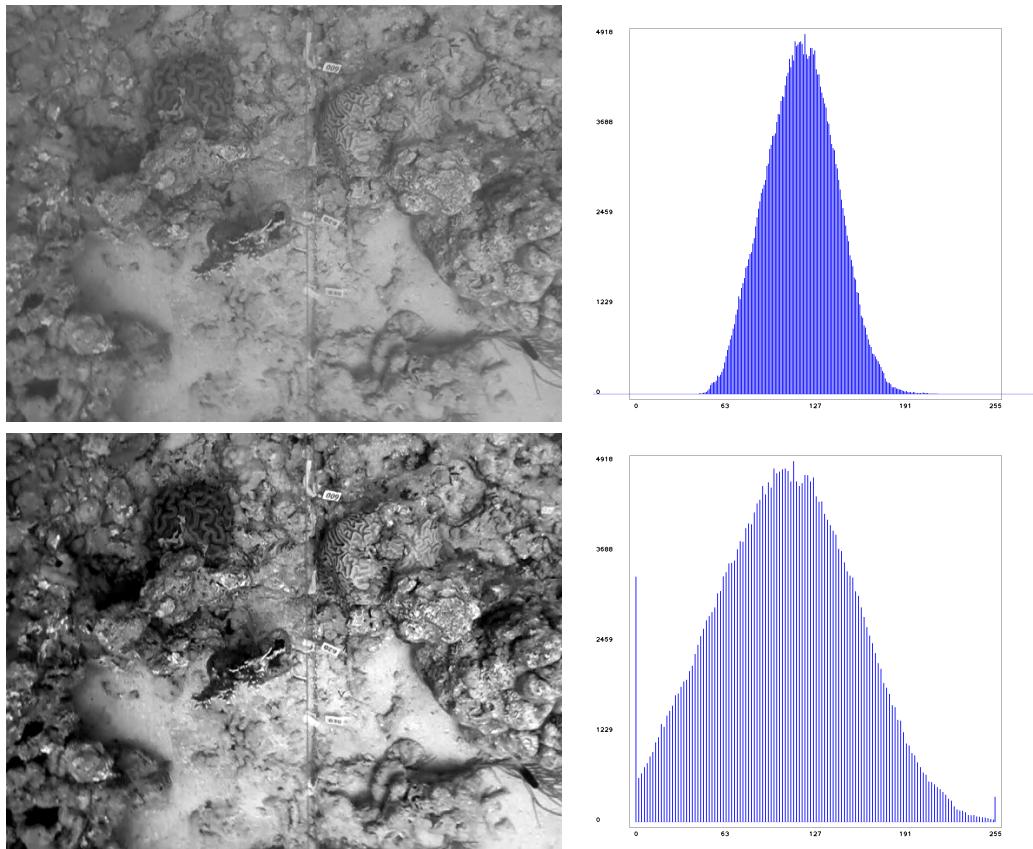


FIGURA 3.11: Arriba: imagen original y su respectivo histograma a su derecha. Abajo: Imagen luego de aplicar el estiramiento del histograma, y su respectivo histograma a su derecha.

3.3.2. Resultados

A continuación se presentan los resultados, para todas las combinaciones de extractores y emparejadores, bajo distintas condiciones de escena. Todas las pruebas mostradas en esta sección se realizaron utilizando un equipo con las siguientes características: *Intel® Core 2 Duo CPU E8400 @ 3.00Ghz*.

En el cuadro 3.1 se observan los resultados para 61 pares de imágenes del fondo marino en la región de Chuspa (conjunto *Chuspa*), ubicada en el estado Vargas, Venezuela. Adicionalmente, en la figura 3.12 se ilustran cuatro imágenes representativas del conjunto estudiado. Para esta prueba se utilizaron imágenes de 640×480 píxeles.

En el cuadro 3.3 se observan los resultados para 42 pares de imágenes. Estas

CUADRO 3.1: Comparación de rendimiento usando imágenes del conjunto *Chuspa*

		SIFT	SURF	ORB	KAZE	A-KAZE
Fuerza Bruta	Total parejas	68679	32888	29637	6818	6896
	Buenas parejas	12601	3764	224	1407	520
	Precisión (%)	18.34	11.44	0.70	20.63	7.54
	Tiempo (s)	38.27	13.60	2.32	69.09	16.99
FLANN	Total parejas	68679	32888	29637	6818	6896
	Buenas parejas	13221	3895	282	1416	553
	Precisión (%)	19.25	11.84	0.95	20.76	8.01
	Tiempo (s)	38.27	13.60	2.32	69.09	16.99

CUADRO 3.2: Comparación de rendimiento usando imágenes del conjunto *Chuspa*, luego de aplicar estiramiento del histograma

		SIFT	SURF	ORB	KAZE	A-KAZE
Fuerza Bruta	Total parejas	235284	98402	30500	59061	62957
	Buenas parejas	29200	7461	200	11812	3115
	Precisión (%)	12.41	7.58	0.65	19.99	4.94
	Tiempo (s)	113.29	28.81	3.44	75.40	21.66
FLANN	Total parejas	235284	98402	30500	59061	62957
	Buenas parejas	31917	7461	257	12522	3617
	Precisión (%)	13.53	8.17	0.84	21.20	5.74
	Tiempo (s)	63.81	25.79	3.78	75.20	20.58



FIGURA 3.12: imágenes representativas del conjunto *Chuspa*

imágenes fueron proporcionadas por el centro Australiano de Robótica de Campo ACFR (del inglés: Australian Center for Field Robotics) y pertenecen al conjunto de datos *ScottReef 25*. Adicionalmente, en la figura 3.13 se ilustran cuatro imágenes representativas del conjunto estudiado. Para esta prueba se utilizaron imágenes de 640×480 píxeles.

En el cuadro 3.5 se observan los resultados para 51 pares de imágenes del fondo marino del parque nacional Mochima (conjunto *Mochima*), ubicado en el estado Sucre, Venezuela. Adicionalmente, en la figura 3.12 se ilustran cuatro imágenes representativas del conjunto estudiado. Para esta prueba se utilizaron imágenes de

CUADRO 3.3: Comparación de rendimiento usando imágenes submarinas del conjunto *ScottReef 25*

		SIFT	SURF	ORB	KAZE	A-KAZE
Fuerza Bruta	Total parejas	100620	48955	21000	17952	19430
	Buenas parejas	3631	1383	55	1511	239
	Precisión (%)	3.60	2.82	0.26	8.41	1.23
	Tiempo (s)	49.87	16.97	5.90	52.74	15.05
FLANN	Total parejas	100620	48955	21000	17952	19430
	Buenas parejas	3987	1480	65	1604	282
	Precisión (%)	3.96	3.02	0.30	8.93	1.45
	Tiempo (s)	36.82	15.70	5.95	51.48	15.22

CUADRO 3.4: Comparación de rendimiento usando imágenes submarinas del conjunto *ScottReef 25*, luego de aplicar estiramiento del histograma

		SIFT	SURF	ORB	KAZE	A-KAZE
Fuerza Bruta	Total parejas	231242	115193	21000	134065	127859
	Buenas parejas	5987	2317	62	7621	796
	Precisión (%)	2.58	2.01	0.29	5.68	0.62
	Tiempo (s)	125.38	36.427	7.08	78.11	35.47
FLANN	Total parejas	231242	115193	21000	134065	127859
	Buenas parejas	6847	2542	65	9034	1052
	Precisión (%)	2.96	2.20	0.30	6.73	0.82
	Tiempo (s)	60.10	29.59	7.02	66.69	26.72



FIGURA 3.13: imágenes representativas del conjunto *ScottReef 25*

640 × 480 píxeles.

En el cuadro 3.7 se observan los resultados para 50 pares de imágenes aéreas de una cantera de grava (conjunto *Grava*), ubicado en Suiza. Estas imágenes fueron proporcionadas por la empresa SenseFly⁵ y capturadas por un dron eBee⁶. Adicionalmente, en la figura 3.15 se ilustran cuatro imágenes representativas del conjunto estudiado. Para esta prueba se utilizaron imágenes de 1280 × 960 píxeles.

⁵<https://www.sensefly.com/education/datasets/>

⁶<https://www.sensefly.com/drone/ebee-mapping-drone/>

CUADRO 3.5: Comparación de rendimiento usando imágenes del conjunto *Mochima*

		SIFT	SURF	ORB	KAZE	A-KAZE
Fuerza Bruta	Total parejas	116840	62472	25382	35934	34733
	Buenas parejas	14659	6068	207	6760	1888
	Precisión (%)	12.58	9.71	0.81	18.81	5.43
	Tiempo (s)	53.24	18.37	2.30	61.14	15.37
FLANN	Total parejas	116840	62472	25382	35934	34733
	Buenas parejas	15690	6426	242	7175	2136
	Precisión (%)	13.42	5.48	0.95	19.96	6.14
	Tiempo (s)	39.23	17.66	2.44	62.07	15.07

CUADRO 3.6: Comparación de rendimiento usando imágenes del conjunto *Mochima*, luego de aplicar estiramiento del histograma

		SIFT	SURF	ORB	KAZE	A-KAZE
Fuerza Bruta	Total parejas	146453	75342	25440	61107	57797
	Buenas parejas	16743	6879	208	12105	3310
	Precisión (%)	11.43	3.13	0.81	19.80	5.72
	Tiempo (s)	68.42	22.05	2.80	66.45	18.69
FLANN	Total parejas	146453	75342	25440	61107	57797
	Buenas parejas	18006	7328	244	12890	3773
	Precisión (%)	12.29	9.72	0.95	21.09	6.52
	Tiempo (s)	45.01	20.27	2.88	64.77	17.57

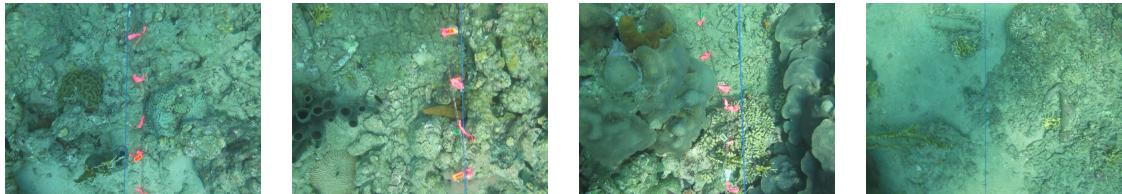


FIGURA 3.14: imágenes representativas del conjunto *Mochima*

Presentadas las pruebas, analizaremos el rendimiento de todas las combinaciones estudiadas en función a los parámetros descritos al inicio de la sección.

Es evidente la inversa relación entre la cantidad de puntos detectados y el tiempo que le toma ese proceso. En función a la detección, podemos acotar que los algoritmos SIFT y KAZE presentan la mayor cantidad de parejas correctas, siendo SIFT el que permanece con el mayor numero a lo largo de todas las pruebas. Este resultado está relacionado con el uso de descriptores vectoriales, que permiten una descripción mas robusta de los puntos detectados. Además que en el caso de SIFT no se realiza ninguna aproximación para la obtención de las diferentes escalas de

CUADRO 3.7: Comparación de rendimiento usando imágenes del conjunto *Grava*

		SIFT	SURF	ORB	KAZE	A-KAZE
Fuerza Bruta	Total parejas	336189	222251	25000	108819	121914
	Buenas parejas	21006	5853	126	10290	3240
	Precisión (%)	3.24	2.63	0.5	9.45	2.65
	Tiempo (s)	282.76	116.38	28.84	309.29	92.71
FLANN	Total parejas	336189	222251	25000	108819	121914
	Buenas parejas	22610	6525	143	10861	3749
	Precisión (%)	6.72	2.93	0.57	9.98	3.07
	Tiempo (s)	153.12	90.27	28.53	299.59	81.52

CUADRO 3.8: Comparación de rendimiento usando imágenes del conjunto *Grava*, luego de aplicar estiramiento del histograma

		SIFT	SURF	ORB	KAZE	A-KAZE
Fuerza Bruta	Total parejas	335418	221287	25000	108124	121122
	Buenas parejas	20943	5812	127	10313	3239
	Precisión (%)	6.24	2.62	0.58	9.53	2.67
	Tiempo (s)	278.97	114.86	29.54	300.12	92.31
FLANN	Total parejas	335418	221287	25000	108124	121122
	Buenas parejas	22555	6509	149	10880	3754
	Precisión (%)	6.72	2.94	0.59	10.06	3.09
	Tiempo (s)	153.71	88.25	29.44	294.61	83.68

FIGURA 3.15: imágenes representativas del conjunto *Grava*

la imagen.

Por el contrario, Aquellos algoritmos que utilizan descriptores binarios presentan un menor rendimiento en base a parejas emparejadas. Sin embargo, esta característica les permite realizar la extracción con una velocidad mucho mayor, siendo este un patrón que se mantiene a lo largo de todas las pruebas.

Analizando los resultados del extractor *KAZE*, Si bien no ofrece la mayor cantidad de parejas correctas, presenta el mejor cociente entre parejas totales y correctas. Lo que indica un gran nivel de robustez en su descriptor, ya que un alto porcentaje de las parejas encontradas en un principio, efectivamente corresponden con el mismo punto.

Comparando el rendimiento de los emparejadores, podemos observar que para la mayoría de los casos el algoritmo FLANN aumenta la cantidad de parejas correctas, y al mismo tiempo disminuye el tiempo computacional. Si bien para la mayoría de las pruebas no se tiene un mejora significativa en términos de tiempo, es debido a que este algoritmo ofrece mayores beneficios cuando la cantidad de parejas a estudiar es mucho mas elevada. Este punto se evidencia mayormente en los resultados del algoritmo SIFT, el cual ofrece mayor cantidad de parejas, y por lo tanto mayor conjunto de puntos a emparejar.

Por ultimo podemos destacar el importante aumento en la cantidad de parejas detectadas, para los casos en los que se aplicó el pre-procesamiento en la entrada. Como ya se mencionó, se debe al aumento en el contraste, lo que produce que la diferencia entre un punto característico y su vecindario se incremente, aumentando al mismo tiempo la posibilidad de ser detectado y luego emparejado.

3.4. Resumen

El proceso de establecer la relación entre las imágenes, es un paso muy importante en la etapa de registro. De esta forma, la selección del algoritmo para la extracción de características, así como también la técnica que se use para establecer las correspondencias determinará la calidad final del mapa. Tal y como se presentó, se dispone de una gran cantidad de algoritmos para la extracción de características locales, así como también métodos eficientes para emparejarlos. Esta diversidad permite que se cuente con algoritmos que se comporten de manera eficiente para cada tipo de aplicación, o en el caso de mapeo, para cada tipo de escena.

En el presente capítulo se logró establecer una clasificación sobre los algoritmos de generación, en función a como aborden las etapas mas importantes de éste.

Tomando en cuenta los distintos métodos y técnicas, se planteó un sistema que combina aquellas que presentan los mejores resultados, considerando los requerimientos del presente proyecto. Al mismo tiempo se describió el funcionamiento de todos algoritmos que se plantean implementar en el sistema, permitiendo que se puedan plantear técnicas para mejorar el rendimiento en la extracción y emparejamiento.

Tras los resultados mostrados, se logró evidenciar claramente el rendimiento de cada combinación en función a los parámetros de tiempo y cantidad de parejas. Lo que permite una selección en función de los requerimientos de la aplicación, y debido a la modularidad del esquema planteado es posible reemplazar cualquiera de los algoritmo presentes para extraer y emparejar sin afectar el resto del proceso para la construcción del mosaico.

Capítulo 4

Alineación de imágenes

4.1. Introducción

De la sección anterior estudiamos la primera etapa del proceso de registro de imágenes, estableciendo la correspondencia de puntos entre estas. Ahora bien, en esta sección se continúa la etapa de registro pero abordando el tema de las transformaciones geométricas, en primer lugar presentando una descripción y explicación de su estructura y seguido del proceso de su estimación a partir de los puntos previamente obtenidos. Una vez la etapa de registro este completa, se profundiza en el proceso para la alineación de imágenes en un plano común, presentando los algoritmos utilizados para su implementación, así como también técnicas para reducir el error acumulado de las etapas anteriores. Finalmente se presentan resultados por separado de cada proceso previamente descrito, con su respectivo análisis.

4.2. Revisión teórica

A continuación se expone una revisión de los conceptos básicos que involucran una transformación geométrica, donde se plantea una jerarquía de los distintos niveles de transformaciones basadas en sus propiedades geométricas. Una vez se tengan claros estos conceptos, se presentan los métodos matemáticos que permiten obtener dichas transformaciones.

4.2.1. Transformaciones geométricas

El siguiente paso en el proceso de registro, luego de establecer la correspondencia entre las imágenes, es encontrar la transformación geométrica que permite alinearlas. Para comprender su funcionamiento primero es necesario introducir el concepto de espacio proyectivo, sobre el cual se efectuarán dichas transformaciones. En base a esto comenzaremos presentando la notación homogénea para puntos y líneas en este espacio.

Una línea en el espacio representada por la ecuación $ax + by + c = 0$, se puede expresar en forma vectorial de la forma (a, b, c) . Así, los vectores $k(a, b, c)$ representan todo el conjunto de rectas que son equivalentes — $(ka)x + (kb)y + (k)c = 0$ —, para cualquier $k \neq 0$. De esta forma, se conoce como un vector homogéneo al vector particular que pertenezca a un grupo de vectores bajo esta relación de equivalencia. En base a este concepto se introduce la definición 4.1 de espacio proyectivo.

Definición 4.1. El espacio proyectivo \mathbb{P}^2 está conformado por el grupo de rectas o vectores equivalentes en el conjunto $\mathbb{R}^2 - \{0\}$, es decir, excluyendo la recta $(0, 0, 0)$.

Conociendo el espacio proyectivo, y la notación de vectores homogéneos, es importante introducir el concepto de puntos homogéneos, y establecer una relación que nos permita transformarlos a su espacio euclíadiano original.

Teniendo un punto $\mathbf{x} = (x, y)$, este pertenece a la recta $\mathbf{l} = (a, b, c)$ si logra satisfacer su ecuación — $ax + by + c = 0$ —. Si lo escribimos en forma vectorial, tendríamos $(x, y, 1)(a, b, c) = 0$, esto es añadiendo un 1 como tercer componente del punto \mathbf{x} . Similar al concepto previo de vectores homogéneos, se tiene que el conjunto de puntos $k(x, y, 1)$ son representaciones equivalentes del mismo punto, ya que todo ese conjunto pertenece a la misma recta — $k(x, y, 1)(a, b, c) = 0 \rightarrow (ka)x + (kb)y + (k)c = 0$ —. Así, añadiendo esta coordenada, tenemos que los puntos son representados como vectores homogéneos y pertenecen al espacio \mathbb{P}^2 . De esta forma se introduce la siguiente definición 4.2.1 que relaciona la siguiente conversión: $\mathbb{P}^2 \rightarrow \mathbb{R}^2$.

Definición 4.2. La representación de un vector homogéneo es de la forma $\mathbf{x} = (x_1, x_2, x_3)$, representando el punto $(x_1/x_3, x_2/x_3)$ en \mathbb{R}^2 , con $x_3 \neq 0$.

Teniendo claros los conceptos de representación homogénea, se presenta formalmente la definición de una homografía (4.3) o transformación geométrica en el espacio proyectivo.

Teorema 4.3. *Una transformación $h : \mathbb{P}^2 \rightarrow \mathbb{P}^2$ es una homografía si y solo si existe una matriz no singular \mathbf{H}_{3x3} , para la cual, cualquier punto en \mathbb{P}^2 representado por un vector \mathbf{x} se cumple que $h(\mathbf{x}) = \mathbf{H}\mathbf{x}$.*

Existen varias formas de describir las transformaciones geométricas, la primera es algebraica, en la cual se muestra la estructura de la matriz de transformación, y la segunda es analizando las variables que se preservan, o que se mantienen invariantes luego de aplicar dicha transformación. Adicionalmente, cada transformación geométrica está caracterizada por los grados de libertad, o la cantidad de parámetros que cada una puede variar sobre la imagen de entrada. Esta definición se introduce a continuación (4.4).

Definición 4.4. Los grados de libertad *DoF* (del inglés: Degrees of Freedom), son la cantidad mínima de parámetros independientes que pueden especificar el movimiento de un objeto. En otras palabras, es la mínima cantidad de movimientos independientes que puede realizar un objeto. En el caso de transformaciones geométricas, corresponde con el numero de componentes "libres" que permiten especificar dicha transformación.

En base a los grados de libertad, a las propiedades geométricas y al tipo de invarianza que presenten, se tienen cuatro niveles de transformaciones geométricas: isometría, similaridad, afín y perspectiva. Estos son descritos a continuación.

ISOMETRÍA

Este termino proviene del griego iso (igual) metria (medida), y tal como su nombre lo indica, ésta transformación se caracteriza fuertemente por mantener

iguales las longitudes entre todos los puntos. Ésta se compone por una combinación de rotaciones y translaciones — transformaciones euclidianas — donde se modela el movimiento de un objeto rígido. En la ecuación 4.1 se puede observar su representación matricial.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & tx \\ \sin \theta & \cos \theta & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (4.1)$$

$$(x', y', 1) \rightarrow (x', y')$$

Esta matriz es posible expresarla por bloques, como se muestra a continuación:

$$\mathbf{x}' = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{x}$$

Donde \mathbf{R} sería una matriz ortogonal de rotación 2×2 , \mathbf{t} es un vector columna de translación de 2 componentes y $\mathbf{0}$ es un vector fila nulo de 2 componentes.

- **Invarianza:** Distancia entre puntos, ángulo entre líneas y área.
- **3 Grados de libertad:** Ángulo de rotación, traslación en eje x , traslación en eje y . Puede ser calculada a partir de dos puntos correspondientes.

SIMILARIDAD

Esta transformación es una combinación de una isometría con un escalado, en este caso el escalado es de igual magnitud en los ejes x e y . La representación matricial se observa en la ecuación 4.2.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} s \cos \theta & -s \sin \theta & tx \\ s \sin \theta & s \cos \theta & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (4.2)$$

$$(x', y', 1) \rightarrow (x', y')$$

Esta matriz es posible expresarla por bloques, como se muestra a continuación:

$$\mathbf{x}' = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{x}$$

Donde \mathbf{R} sería una matriz ortogonal de rotación 2×2 , s es el factor de la escala, \mathbf{t} es un vector columna de translación de 2 componentes y $\mathbf{0}$ es un vector fila nulo de 2 componentes.

- **Invarianza:** Cociente de longitudes, Ángulo entre líneas y área.
- **4 Grados de libertad:** Ángulo de rotación, traslación en eje x , traslación en eje y . Puede ser calculada a partir de dos puntos correspondientes.

AFÍN

La transformación afín (también llamada afinidad) está compuesta por una transformación lineal (rotación, sesgo, homotecia), seguida de una translación. En esta se tiene una transformación más compleja que las anteriores, ya que se incluyen algunas deformaciones que no permiten conservar la forma original de los objetos. De igual forma puede incluirse un escalado, pero a diferencia de la anterior, éste puede ser de distinta magnitud en los ejes x e y . La representación matricial se observa en la ecuación 4.3.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & tx \\ a_{21} & a_{22} & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (4.3)$$

$$(x', y', 1) \rightarrow (x', y')$$

Esta matriz es posible expresarla por bloques, como se muestra a continuación:

$$\mathbf{x}' = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{x}$$

Donde \mathbf{A} sería una matriz ortogonal 2×2 no singular, \mathbf{t} es un vector columna de translación de 2 componentes y $\mathbf{0}$ es un vector fila nulo de 2 componentes.

- **Invarianza:** Lineas paralelas, cociente de longitudes de lineas paralelas.
- **6 Grados de libertad:** 4 componentes de la matriz lineal, traslación en eje x , traslación en eje y . Puede ser calculada a partir de tres puntos correspondientes.

PROYECTIVA

Por último se presenta la transformación proyectiva u homografía. Las transformaciones mostradas anteriormente tienen en común la ultima fila de la matriz que la describe — $(0, 0, 1)$ —, lo cual hace que el tercer componente de la representación homogénea nunca cambie de 1. Por esta razón se dice que estas transformaciones son de coordenadas no-homogéneas. El caso de la homografía es distinto, ya que se tiene una transformación lineal de coordenadas homogéneas (definición 4.2.1). La representación matricial se puede ver en la ecuación 4.4.

$$\begin{pmatrix} x' \\ y' \\ w \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (4.4)$$

$$(x', y', w) \rightarrow (x'/w, y'/w), w \neq 0$$

Esta matriz es posible expresarla por bloques, como se muestra a continuación:

$$\mathbf{x}' = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v} & u \end{bmatrix} \mathbf{x}$$

Donde \mathbf{A} sería una matriz ortogonal 2×2 no singular, \mathbf{v} es un vector fila de 2 componentes, \mathbf{t} es un vector columna de translación de 2 componentes, $\mathbf{0}$ es un vector fila nulo de 2 componentes y u es el factor de escala de la transformación.

- **Invarianza:** cociente cruzado (cociente de cocientes de longitudes), puntos de contacto (intersecciones).
- **8 Grados de libertad:** Si bien la matriz cuenta con 9 componentes, al escalar todos los componentes por u , la transformación quedaría especificada por los 8 valores restantes.

En la figura 4.1 se ilustra gráficamente los tipos de transformaciones descritas previamente.

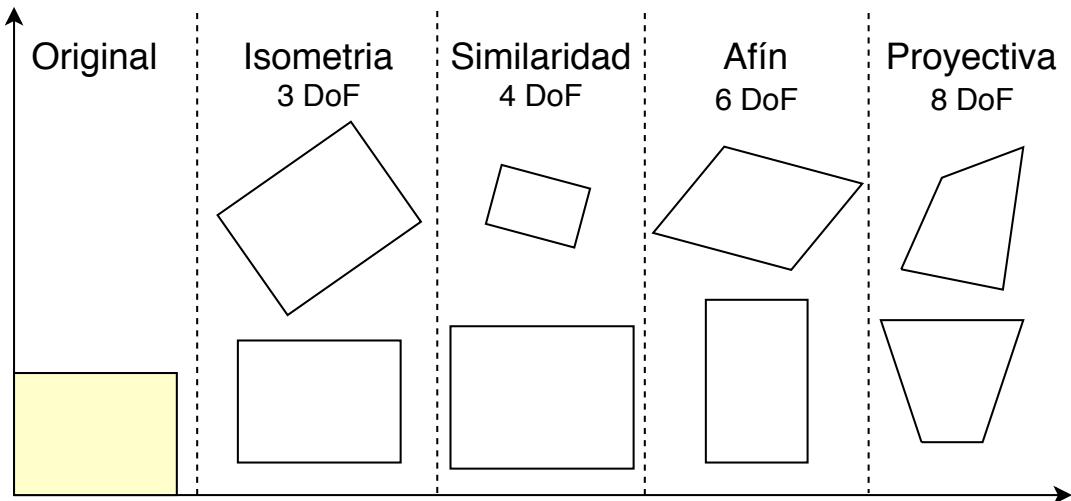


FIGURA 4.1: Tipos de transformaciones geométricas

Las transformaciones proyectivas, al igual que las anteriores puede ser descompuesta en una serie de transformaciones de menor jerarquía, tal y como se muestra en la ecuación 4.5 para la proyectiva.

$$H = H_S H_A H_P = \begin{bmatrix} sR & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & 0 \\ v & u \end{bmatrix} = \begin{bmatrix} A & t \\ v & u \end{bmatrix} \quad (4.5)$$

Donde los sub-índices de las matrices corresponden con S = similaridad, A = Afín, P = Proyectiva. R es una matriz de rotación (no singular) 2×2 , K es una matriz 2×2 triangular superior normalizada tal que $\det K = 1$ (para remover la escala y rotación), t es un vector de translación de 2 componentes, I es una matriz identidad 2×2 , v es un vector de 2 componentes, A es una matriz no singular y $u \neq 0$.

Las transformaciones proyectivas al estar compuestas por un grupo de transformaciones también cumplen con las siguientes aserciones:

- (i) El inverso de una homografía, es también una homografía.
- (ii) La composición de dos homografía es una homografía.

En base a esto, siempre es posible encontrar una matriz de transformación que pueda relacionar dos planos.

4.2.2. Estimación de Homografía

El ultimo nivel de transformación estudiadas posee ocho grados de libertad (*8-DoF*), con lo cual es posible estimar los parámetros de dicha transformación con tan solo 8 valores independientes. Por otro lado, conociendo que la homografía es una transformación que preserva la naturaleza de los objetos — transforma puntos en puntos, lineas en lineas y planos en planos —, solo es posible relacionar dos imágenes mediante una homografía, siempre y cuando la escena que capturen corresponda con un plano. ver figura 4.2.

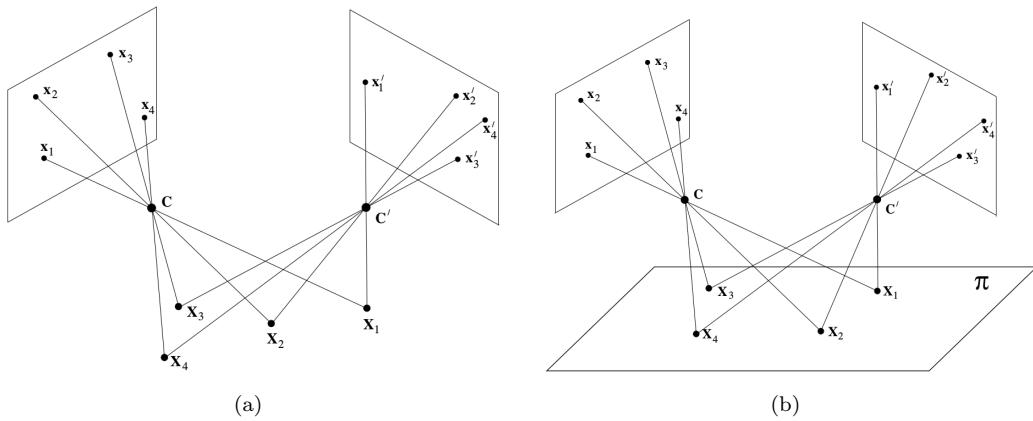


FIGURA 4.2: Los puntos \mathbf{C} y \mathbf{C}' corresponden con los centros de la cámara en las distintas posiciones, Los puntos \mathbf{x}_i y \mathbf{x}'_i son la intersección de los puntos de la escena \mathbf{X}_i en el plano de la imagen (formación de la imagen). Si la cámara se mueve no se podrá relacionar los planos de las imágenes con una homografía (izquierda), a menos que los puntos de la escena se encuentren todos en un mismo plano (derecha). Adaptado de [31].

El primer paso para la estimación de la transformación consiste en encontrar 4 pares correspondientes de puntos, como cada punto en 2D posee 2-DoF (x e y del plano euclíadiano de la imagen) con 4 puntos se tienen los 8-DoF. Luego por cada par de puntos se tienen (a partir de la ecuación 4.4) las siguientes relaciones:

$$x'_i = \frac{x'}{w} = \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}}, \quad y'_i = \frac{y'}{w} = \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}}$$

Como se observa cada par de puntos genera dos ecuaciones lineales, mostradas a continuación luego de simplificar:

$$x'_i(h_{31}x_i + h_{32}y_i + h_{33}) = h_{11}x_i + h_{12}y_i + h_{13}$$

$$y'_i(h_{31}x_i + h_{32}y_i + h_{33}) = h_{21}x_i + h_{22}y_i + h_{23}$$

Escrito en forma matricial se puede expresar de la siguiente forma:

$$\mathbf{A}_i \mathbf{h} = \mathbf{0}$$

Donde

$$\mathbf{A}_i = \begin{bmatrix} -x_i & -y_i & -1 & 0 & 0 & 0 & x'_i x_i & x'_i y_i & x'_i \\ 0 & 0 & 0 & -x_i & -y_i & -1 & y'_i x_i & y'_i y_i & y'_i \end{bmatrix}$$

Y

$$\mathbf{h}^T = (h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33})$$

Dado que 1 pareja arroja 2 ecuaciones lineales, 4 parejas arrojarán 8 ecuaciones, y se puede resolver el sistema para un factor multiplicativo dado por h_{33} ($h_{33} = 1$). La única restricción que tiene esta sistema de ecuaciones para asegurar un resultado único, es que no se tengan 3 puntos que residan en la misma linea. Con esto se tiene el siguiente sistema de ecuaciones:

$$\mathbf{A} \mathbf{h} = \begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x'_1 x_1 & x'_1 y_1 & x'_1 \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & y'_1 x_1 & y'_1 y_1 & y'_1 \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x'_2 x_2 & x'_2 y_2 & x'_2 \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & y'_2 x_2 & y'_2 y_2 & y'_2 \\ -x_3 & -y_3 & -1 & 0 & 0 & 0 & x'_3 x_3 & x'_3 y_3 & x'_3 \\ 0 & 0 & 0 & -x_3 & -y_3 & -1 & y'_3 x_3 & y'_3 y_3 & y'_3 \\ -x_4 & -y_4 & -1 & 0 & 0 & 0 & x'_4 x_4 & x'_4 y_4 & x'_4 \\ 0 & 0 & 0 & -x_4 & -y_4 & -1 & y'_4 x_4 & y'_4 y_4 & y'_4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0$$

Añadiendo la condición de $|\mathbf{H}| = 1$ para evitar la solución trivial $\mathbf{h} = 0$. Luego se puede resolver mediante el método de descomposición en valores singulares SVD (del inglés: Singular Value Decomposition) con $\mathbf{A} = \mathbf{U} \mathbf{S} \mathbf{V}^T$, donde \mathbf{U} y \mathbf{V} son matrices ortogonales; y \mathbf{S} es una matriz formada con los valores singulares de \mathbf{A} en su diagonal principal, ordenados en orden descendente. Finalmente el vector singular unitario correspondiente con el menor valor singular es la solución \mathbf{h} .

En este punto, la matriz encontrada se puede usar para transformar $\mathbf{x} \rightarrow \mathbf{x}'$, o bien se puede usar la inversa (\mathbf{H}^{-1}) para realizar el mapeo en sentido contrario.

Dada la naturaleza no plana de las escenas del mundo real, o bien debido a la inexactitud al medir los puntos cuando se captura una imagen, es muy poco

probable que los puntos correspondientes radiquen en un mismo plano. Por otra parte es muy común que para aplicaciones de mapeo automático se cuenten con una gran cantidad de puntos correspondientes, con lo cual se tiene un sistema sobre determinado. En este caso es necesario realizar aproximaciones que permitan encontrar ya no la única homografía, sino la mejor, en función a los datos con los que se cuenten. Para esto es usual el uso de algoritmos que estiman la matriz buscando minimizar alguna función de coste.

Uno de los algoritmos mas usados para encontrar el mejor modelo está basado en el método RANSAC (del inglés: Random Sample Consensus), el cual es explicado a continuación.

RANSAC

Este es un método iterativo que permite calcular los parámetros de un modelo, a partir de un conjunto de datos que presentan valores atípicos. En nuestro caso, se tiene un conjunto sobre determinado de pares de puntos correspondientes, de los cuales se deben seleccionar los mejores 4 pares para estimar la mejor matriz de transformación, siendo la mejor aquella que logre minimizar la función de coste planteada.

Como ya se ha mencionado, las escenas que se estudian no son completamente planas, lo que provoca una inexactitud al estimar la transformación. Para lograr establecer un modelo, el método RANSAC considera solo los mejores datos que se logren ajustar a él, en este sentido se definen los *inliers* y *outliers*, donde los *inliers* son aquellos valores que logran encajar en el modelo deseado, mientras que los *outliers* son aquellos que no logran hacerlo. Por ejemplo: si se tienen un conjunto de puntos cuya distribución corresponden con el plano del suelo, estos serían los *inliers*, y por otro lado aquellos puntos que no pertenecen a este plano serían los *outliers*, ya sea por un emparejamiento erróneo o que pertenezcan a un segmento de distinta altitud de la escena. En la figura 4.3 se ilustra esta distinción.

Ahora bien es necesario establecer la función de costo que buscará minimizar el algoritmo. En este caso se intenta disminuir el error cuadrático medio, donde el error es medido con la distancia de los puntos (*inliers*) correspondientes. Tal y como se observa en la figura 4.3 el método RANSAC establece un perímetro que

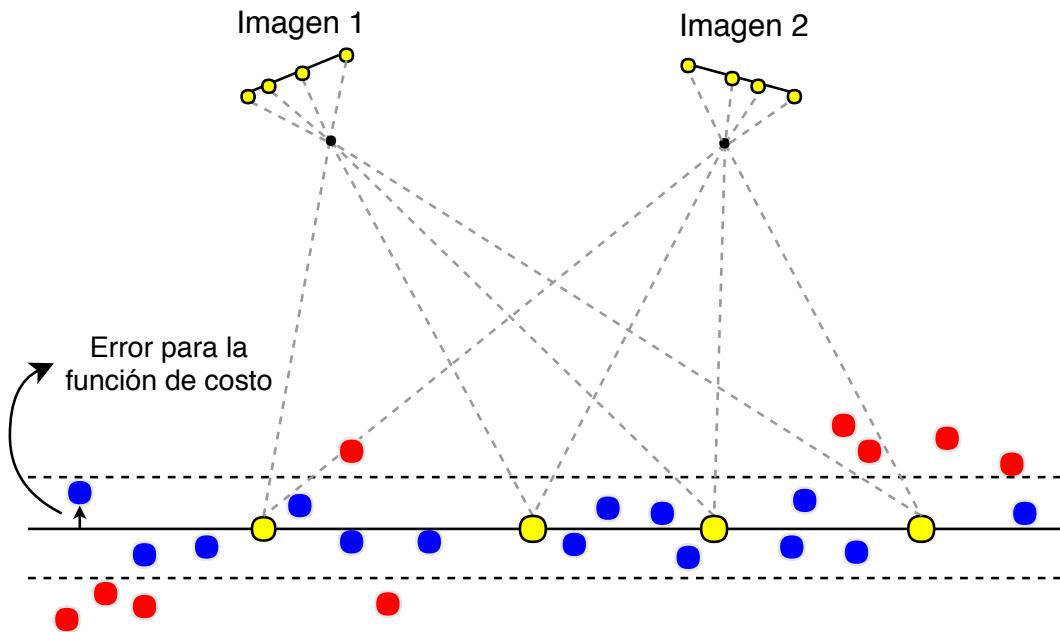


FIGURA 4.3: Representación de un plano visto lateralmente. Los puntos amarillos corresponden con los utilizados para establecer el modelo (para calcular la matriz). Los puntos azules corresponden con los *inliers*, mientras que los rojos con los *outliers*. La distancia para la función de coste se mide desde cada punto hasta el plano modelo, considerando únicamente los *inliers*.

le permite distinguir *inliers* de *outliers*, ahora bien una vez se logran distinguir, el algoritmo solo considera los mejores para obtener el costo o peso asociado al modelo estimado, calculando para esto la distancia desde cada inlier hasta el plano. Este proceso se puede apreciar en el algoritmo 3.

Algoritmo 3: Estimación de matriz de transformación RANSAC

```

31 inicio
32   mientras No se llegue al maximo de iteraciones hacer
33     Se toman 4 parejas correspondientes;
34     Se calcula la matriz de homografía;
35     Se aplica la matriz sobre los puntos de origen;
36     Se calcula el error basado en distancia; // Solo considerando inliers
37     si Error < MejorError entonces
38       Guardar matriz como la mejor;
39       Guardar Error como el mejor;
40     fin
41   fin
42 fin

```

4.3. Generación de sub-mosaicos

El proceso mas simple para la generación de un mosaico consiste en añadir imágenes siguiendo los procesos planteados previamente, estableciendo el registro, y luego aplicando la matriz de transformación para lograr referenciarlas en el plano del mosaico. Ahora bien para el caso del primer par de imágenes, esta práctica implica que se hallo la transformación desde la nueva añadida hacia la primera, tomando esta primera como plano de referencia. Esto se conoce como el proceso básico para la elaboración de mosaicos, en el cual se considera la primera imagen como referencia global para la construcción del mapa. La selección errónea de una imagen de referencia puede generar grandes distorsiones en el mosaico (especialmente para mapas grandes), y considerando el sistema simple, si no se cuenta con una primera imagen que se encuentre muy bien alineada con el plano del suelo, se comprometerá gravemente el resultado final del mosaico.

En busca de resolver este problema, *Bellavia et. al.* en [23, 24] plantean el modelo de sub mosaicos para lograr reducir el error geométrico producto de la mala selección de la imagen de referencia. Este sistema propone una estrategia para la selección del mejor plano de referencia que logre minimizar el error de distorsión, mediante la construcción de pequeños segmentos de mosaico que se alinean bajo el esquema básico, pero que poseen localmente poca distorsión geométrica.

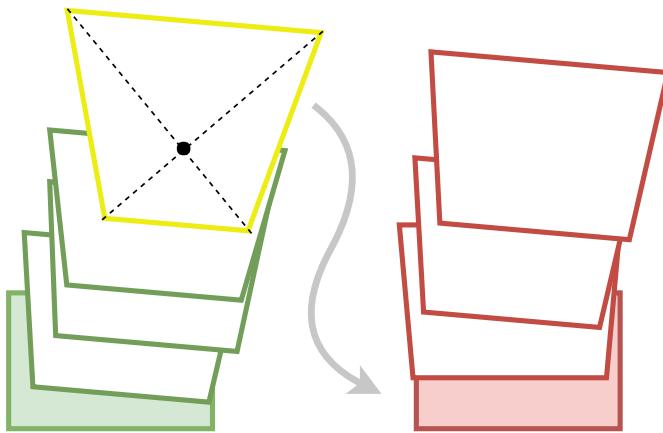


FIGURA 4.4: Los cuadros verde y rojo son dos sub mosaico temporalmente continuos. El cuadro amarillo se convierte en el rojo relleno por presentar mucha distorsión.

La generación de los sub mosaicos consiste en el siguiente proceso: refiriéndonos a la figura 4.4, se selecciona como referencia la primera imagen (cuadro verde lleno), se añaden imágenes al sub mosaico sucesivamente hasta que la siguiente

presente mucha distorsión (cuadro amarillo), en este punto se crea un nuevo sub mosaico (cuadros rojos) y se añade aquella que presentó mucha distorsión como referencia para el nuevo sub mosaico. Donde el error es medido en función al área y el cociente entre semi-diagonales.

Siguiendo este proceso se van construyendo sub mosaicos hasta que no se cuenten con mas imágenes. Una vez se han creado todos los sub-mosaicos es necesario unirlos. En este punto cada sub mosaico considera la primera imagen como el plano de referencia, por lo tanto es necesario definir cual será la referencia bajo la cual éstos deben unirse. Para esto se utiliza la homografía que logre disminuir el error de distorsión sobre todos las imágenes del sub mosaico, esta transformación se define como *homografía promedio*. El proceso para su obtención se detalla a continuación.

4.3.1. Matriz de transformación promedio

Refiriéndonos a la figura 4.5, se observan los dos sub mosaicos que se desean unir, los cuales son temporalmente continuos, en la imagen superior se observan los sub mosaicos S_j , S_k (marcos verde oscuro y verde claro) alineados y considerando la imagen de referencia del primero (verde lleno) como imagen de referencia global. En el segundo caso se muestran ambos alineados pero considerando la imagen de referencia del segundo (rojo lleno) como imagen de referencia global.

Para encontrar la homografía promedio se aplica un algoritmo basado en el método RANSAC, el cual consiste en obtener 4 pares de puntos correspondientes ($x_i^1 \leftrightarrow x_i^2$) entre los dos sub mosaicos (verde y rojo) para un número fijo de iteraciones, seguidamente para dichos puntos se calcula el punto medio (x_i^a), luego se calcula la matriz que transforma los puntos del primer sub mosaico hacia los puntos medios ($x^a = Hx^1$), después se aplica la homografía en el primer sub mosaico y se obtiene la distorsión para dicha transformación. Finalmente la matriz que logre minimizar la distorsión global en el nuevo sub mosaico es seleccionada como la homografía promedio. Éste proceso se muestra mejor estructurado en el algoritmo 4.

El criterio para determinar el grado de distorsión de la imagen se observa en la ecuación 4.6, tal y como se propone en [24]:

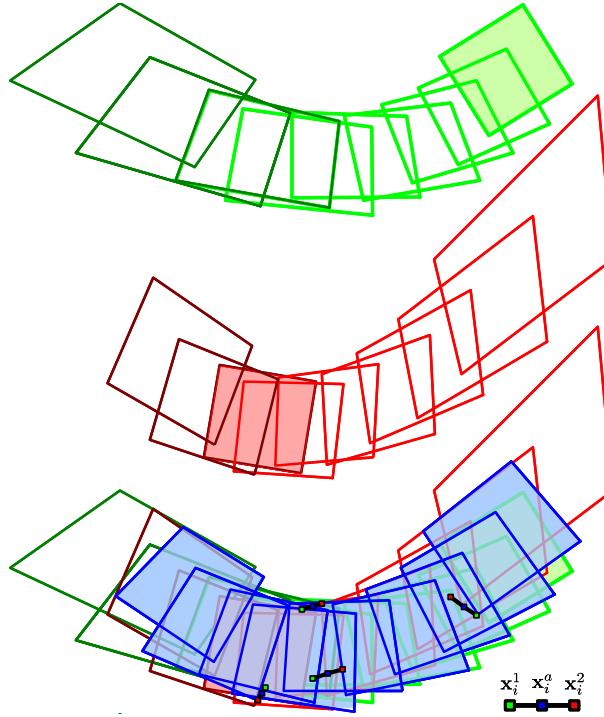


FIGURA 4.5: Estimación de homografía promedio en base a 4 pares de puntos correspondientes, de [24]

$$E = \max_{I_i \in S_j \cap S_k} (E^O + E^C + E^A + E^\alpha) \quad (4.6)$$

Siendo I_i una imagen perteneciente al par de sub mosaicos S_j, S_k . Con lo cual se considera la imagen que presente mas distorsión en el sub mosaico que se intenta unir. El error de cada tipo de error se encuentra normalizado en el rango $0 \rightarrow 1$.

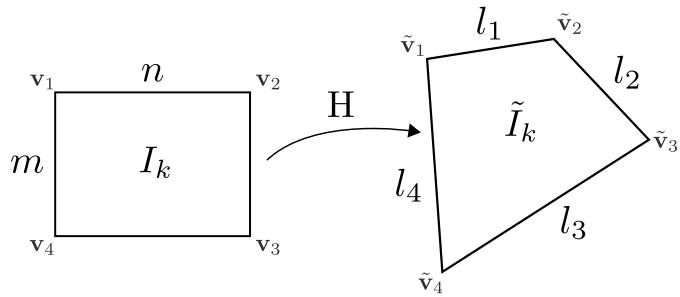


FIGURA 4.6: Parámetros para el calculo de distorsión de una imagen, de [24]

Considerado los parámetros mostrados en la figura 4.6 se describen los componentes del error en la ecuación 4.6.

Algoritmo 4: Calculo de matriz de homografia promedio

```

43 inicio
44   mientras no se alcanza el maximo de iteraciones hacer
45     Seleccionar 4 puntos aleatorios del primer sub-mosaico;
46     Seleccionar los 4 puntos correspondientes en el segundo sub-mosaico;
47     Salcular el punto medio para cada par de puntos correspondientes;
48     Calcular la transformacion desde los puntos del primer sub-mosaico hasta
        los puntos medios;
49     Aplicar transformacion en el primer sub-mosaico;
50     Calcular error de distorsión en el primer sub-mosaico;
51     si el error es menor que el mas bajo obtenido entonces
52       Guardar el error como el mas bajo;
53       Guardar la matriz de transformación como la mejor;
54     en otro caso
55       Restaurar valores del primer sub-mosaico;
56   fin
57 fin
58   Aplicar mejor matriz al primer sub-mosaico;
59 fin

```

E^O mide el error sobre lados opuestos de la imagen.

$$E^O = 1 - \frac{1}{2} \left(\frac{\min(l_1, l_3)}{\max(l_1, l_3)} + \frac{\min(l_2, l_4)}{\max(l_2, l_4)} \right) \quad (4.7)$$

E^C mide el error sobre lados consecutivos, siendo $m > n$.

$$E^C = 1 - \frac{\min(r, \frac{m}{n})}{\max(r, \frac{m}{n})} \quad (4.8)$$

Donde r es el mínimo cociente entre lados consecutivos, expresado como:

$$r = \left(\frac{l_1}{l_2}, \frac{l_2}{l_3}, \frac{l_3}{l_4}, \frac{l_4}{l_1} \right)$$

E^A mide el error del área

$$E^A = 1 - \frac{\min(A_{\tilde{I}_k}, mn)}{\max(A_{\tilde{I}_k}, mn)} \quad (4.9)$$

E^α mide el error sobre los ángulos internos.

$$E^\alpha = (\max(\cos \alpha_{12}, \cos \alpha_{23}, \cos \alpha_{34}, \cos \alpha_{41}))^5 \quad (4.10)$$

4.3.2. Selección de puntos característicos

Según se estudió en el método RANSAC, la estimación de una buena homografía depende de la distribución de los puntos en la escena. Es posible contar con una gran cantidad de puntos extraídos y emparejados en una sección de la escena que no corresponde con el mejor modelo del plano de ésta, lo que indica que la matriz de homografía se ajuste a dicha región por ofrecer menor peso en la función de coste. En cambio, el resultado esperado es que se tenga una mejor distribución de puntos en la escena de modo que se tenga una ponderación de igual magnitud para todas las regiones del suelo.

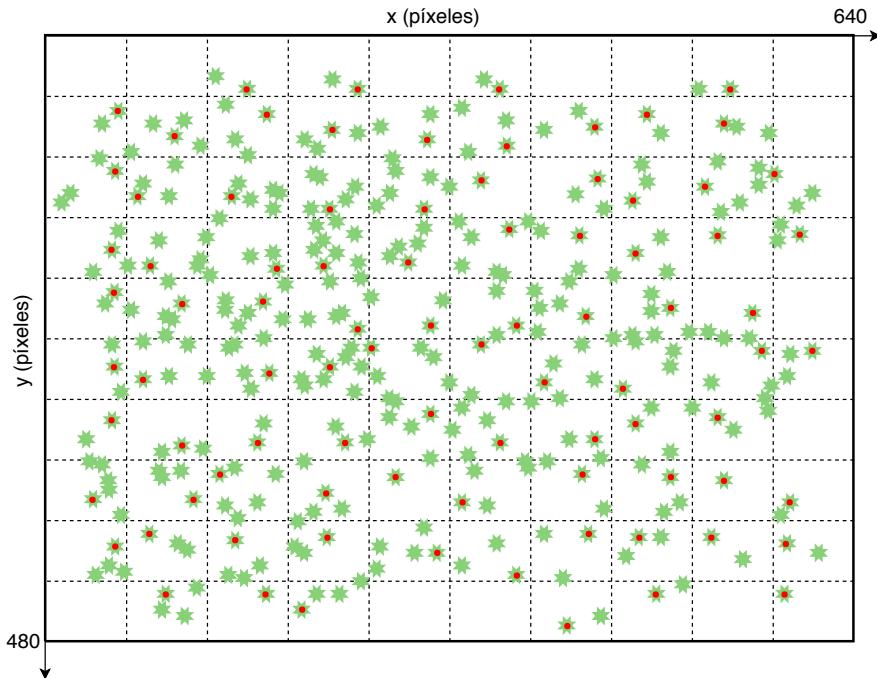


FIGURA 4.7: Selección de mejores parejas en cada celda. Los puntos verdes corresponden con las parejas detectadas, y los puntos rojos serían las mejores parejas por cada celda.

Para solucionar este problema, *Eibol et. al.* en [25] propone una segmentación espacial de los puntos emparejados entre dos imágenes, seguida de un proceso de selección con el objetivo de mejorar la distribución, y de esta forma ponderar

de igual forma todas las secciones del plano. Básicamente el proceso consiste en dividir la imagen en $n \times m$ celdas, y por cada celda seleccionar la mejor pareja de puntos. Teniendo como resultado un máximo de $n \times m$ parejas entre imágenes. Esta división y selección se puede observar en la figura 4.7.

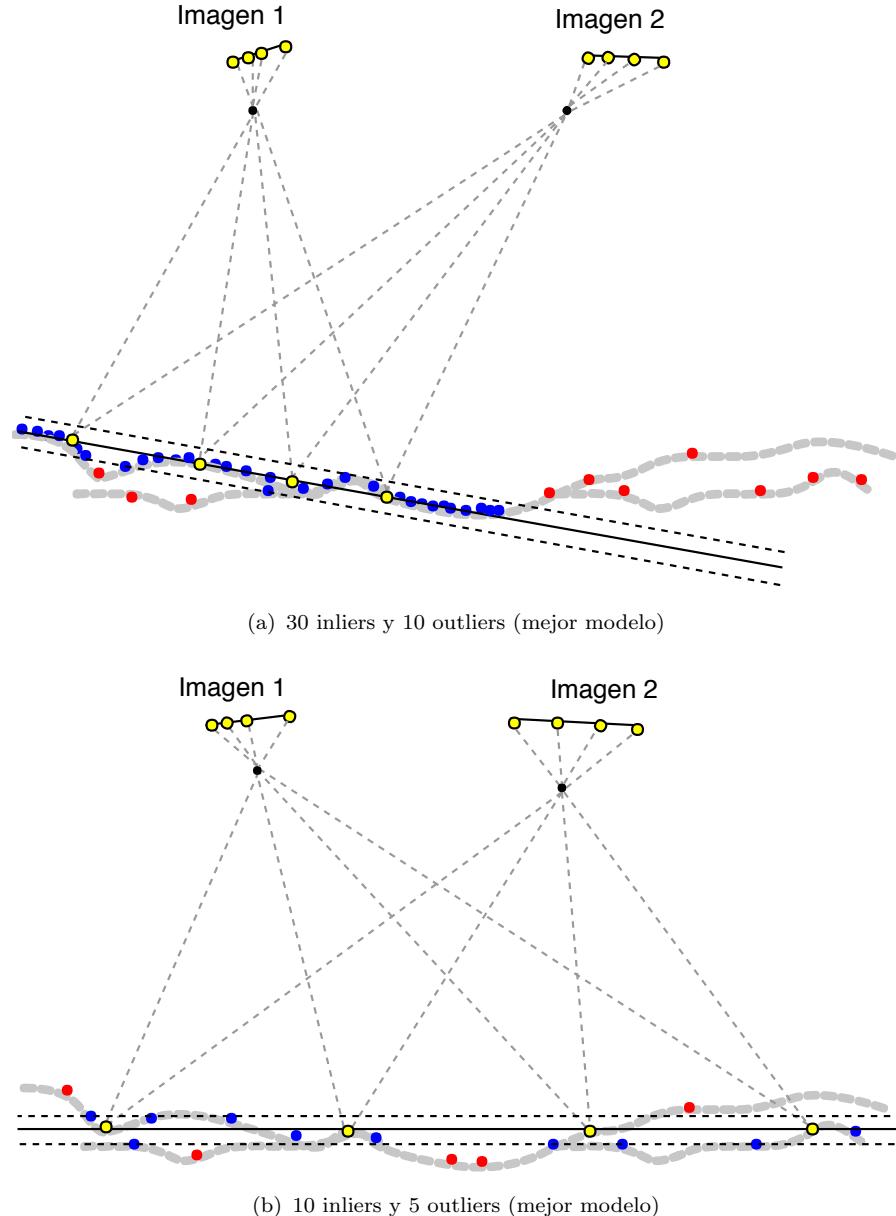


FIGURA 4.8: Ambas imágenes representan la vista lateral del plano de un suelo. En (a) se presenta el mejor modelo del plano con la cantidad original de parejas, mientras que en (b) se muestra el mismo proceso pero luego de haber seleccionado las parejas con enlace mas fuerte por cada región (según 4.7).

Refiriéndonos a la figura 4.8, se observa como en 4.8(a) una fuerte aglomeración de parejas en una sección del plano (región con muchos puntos característicos) produce que la estimación de la matriz de homografía le asigne una alta ponderación

al modelo del plano que se ajusta a esta región.

Por otro lado, en la figura 4.8(b) se observa como se puede mejorar la estimación de la transformación, al seleccionar solo las mejores parejas por cada región, de esta forma la ponderación se distribuye de mejor manera a lo largo del plano del suelo.

4.3.3. Seguimiento de puntos

Ya hemos visto el proceso simple de relacionar dos imágenes, donde luego de aplicar la transformación a la nueva imagen, ésta presentará cierto nivel de distorsión, afectando que imágenes siguientes logren establecer una relación robusta de puntos.

La primera propuesta consiste en buscar parejas de puntos correspondientes entre la nueva imagen y el mosaico (todas las imágenes añadidas previamente), pero esto trae consigo un bajo desempeño computacionalmente cuando se cuenta con un mosaico de gran tamaño. Si bien se puede restringir el área de búsqueda hacia el mínimo cuadro que encierra la última imagen transformada, el efecto de distorsión mencionado afecta la detección e identificación de puntos correspondientes.

Para solucionar este problema se propone realizar una extracción de puntos característicos para todas las imágenes en su versión original, y una vez se conozca la matriz de transformación que ubica una imagen en el mosaico, se puede de igual forma transformar la ubicación de los puntos detectados hacia la nueva ubicación. Este proceso se encuentra ilustrado en la figura 4.9.

Definiendo \mathbf{x}_i^1 como los puntos que corresponden a la nueva imagen, \mathbf{x}_i^2 como los que corresponden a la última añadida con sus dimensiones y posición original, y \mathbf{x}_i^{2*} serían aquellos de la ubicación en el mosaico. Primero se realiza la detección para encontrar la correspondencia $\mathbf{x}_i^1 \leftrightarrow \mathbf{x}_i^2$, luego se transforman los puntos x_i^2 a su ubicación correspondiente en el mosaico $x^{2*} = H^* \mathbf{x}^2$ (H^* conocida). Finalmente se encuentra la matriz de transformación para la nueva imagen con la siguiente relación: $\mathbf{x}^{2*} = H\mathbf{x}^1$.

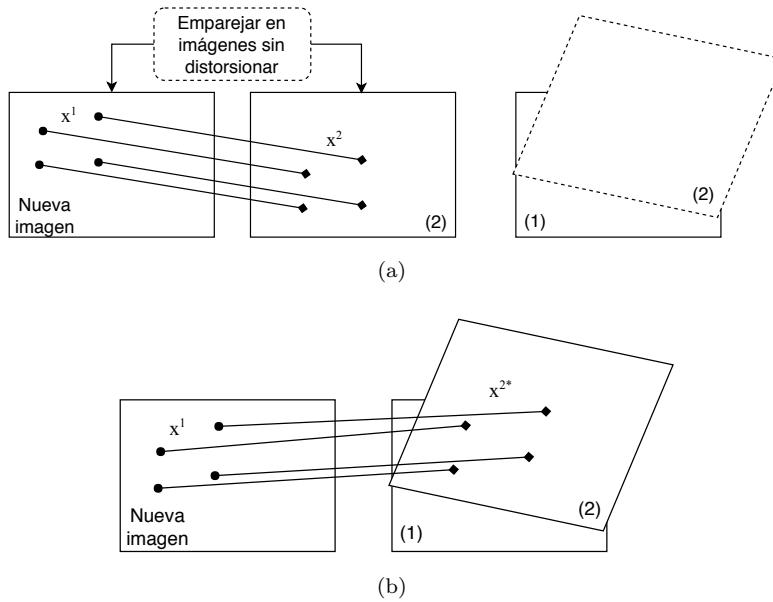


FIGURA 4.9: Seguimiento de puntos. En (a) se realiza la extracción y emparejamiento en las imágenes originales, luego los puntos detectados en la imagen (2) son transformados por la homografía que ubica esa imagen en el mosaico. finalmente en (b) se halla la matriz de transformación para la nueva imagen usando la nueva correspondencia.

4.3.4. Relación con vecinos

Tal y como se ha planteado el proceso de registro hasta este punto, consiste establecer relación entre dos imágenes para estimar la homografía que permite alinearla. Esta práctica es ideal cuando se tiene un movimiento continuo del robot al explorar el suelo, o bien cuando se tiene un bajo porcentaje de superposición (ver figura 4.10(a)), con lo cual se dificulta que una tercera imagen se relacione con la primera. Por otro lado, cuando estas condiciones no se cumplen, es necesario plantear soluciones que aprovechen toda la información posible de las imágenes para tener un sistema mucho mas robusto. En base a esto se planteó una técnica de búsqueda por vecinos para mejorar la etapa de registro, el proceso consiste en buscar relaciones de puntos entre imágenes que no sean temporalmente continuas, es decir, intentar relacionar una imagen con un grupo de todas aquellas inmediatamente anteriores, y que tengan información en común (intersección), a las cuales llamaremos imágenes vecinas (ver figura 4.10(b)).

Los casos presentados en la figura 4.10 es aquel donde el movimiento del robot es monótono en un mismo sentido, pero también se puede presentar el caso en el cual este movimiento no se mantiene, con el cual una imagen distinta a la anterior

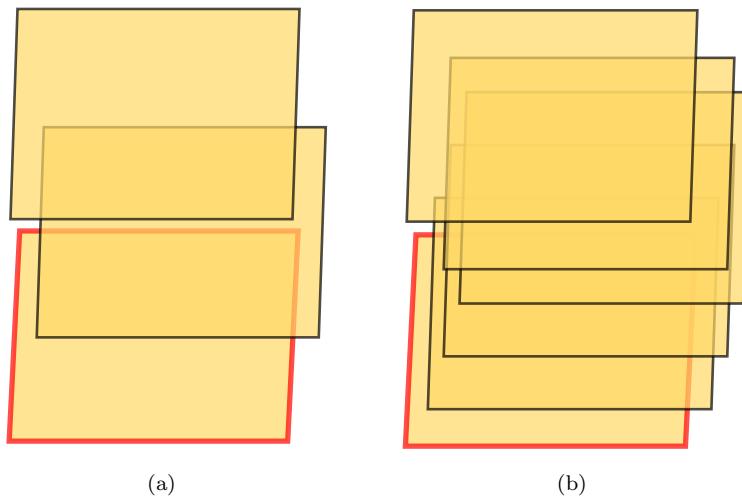


FIGURA 4.10: En función a la ultima imagen añadida (superior), todas las imágenes con marco de color negro son considerados como vecinos. En el caso de la figura (b) una búsqueda en la que solo se considere la imagen anterior no aprovecha toda la información de aquellas con las que se intersecta

puede aportar mas información para establecer la correspondencia con el mosaico. Este caso se ilustra en la figura 4.11, en 4.11(a) el movimiento es continuo hacia arriba, en 4.11(b) el movimiento cambia pero solo se considera la ultima imagen añadida, en 4.11(c) el sentido también cambia pero se consideran ambas imágenes para establecer la relación de homografía.

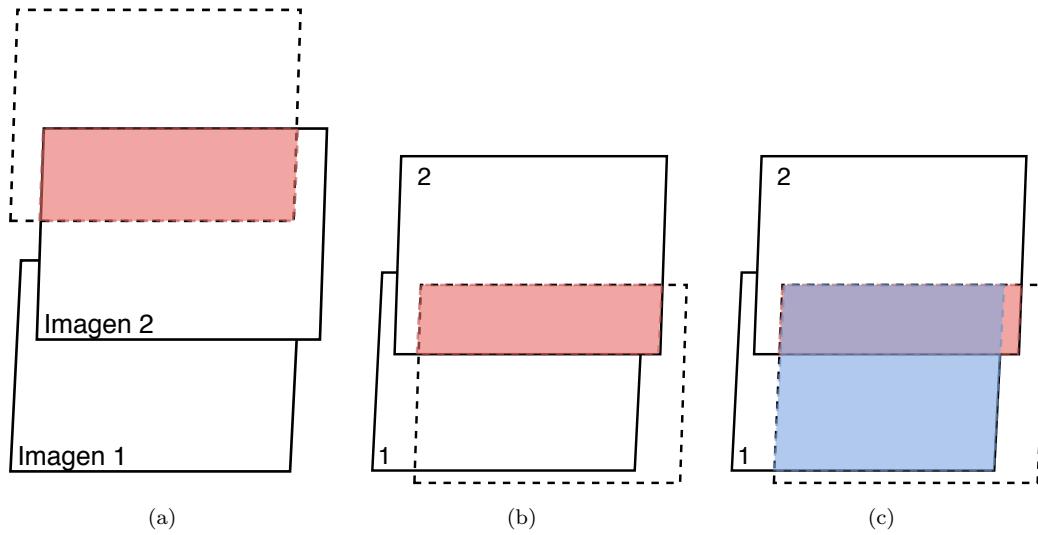


FIGURA 4.11: En rojo se muestra el área de superposición entre la nueva imagen (bordes punteados) y la última añadida (2), mientras que en azul se muestra aquella que comparte con una imagen no consecutiva (1).

Considerando toda la información posible, se tiene un sistema que se adapta a una gran variedad de circunstancias que se pueden presentar en la exploración

cuando se hace uso de un robot no tripulado. En este sentido se presenta en el algoritmo 5 el proceso final para establecer la correspondencia de una imagen con el mosaico, es decir, el proceso de registro de imágenes.

Algoritmo 5: Relacionar imágenes

```

60  $I_{i+1} \equiv$  Imagen nueva;
61  $I_i \equiv$  ultima imagen añadida al mosaico;
62  $V_i \equiv$  vecinos de  $I_i$ ;
63 mientras puntos emparejados  $\geq 4$  hacer
64   | Emparejar puntos de  $I_{i+1}$  con  $I_i$ ;
65   | si  $I_i$  tiene vecinos entonces
66     |   | para cada vecino de  $I_i$  hacer
67       |   |   | Emparejar puntos de  $I_{i+1}$  con  $V_i$ ;
68       |   |   | fin
69     |   | fin
70   |   descartar malos emparejamientos;
71   |   aplicar búsqueda sectorizada;
72   |   si puntos totales emparejados  $\leq 3$  entonces
73     |   modificar criterio para descartar;
74     |   si criterio para descartar llega al minimo entonces
75       |   | terminar ; // no es posible emparejar imagen
76       |   | fin
77     |   | fin
78 fin
```

4.4. Corrección euclíadiana

La concatenación de un gran grupo de imágenes en la formación de un mosaico, suele traer consigo un error de distorsión acumulado producto de la estimación de la mejor matriz de homografía, afectando en muchos casos el tamaño original de las imágenes. Para resolver este problema *A. Elibol y R. Garcia* en [32] proponen un método de corrección global basado en transformaciones de isometría.

Considerando que un robot de exploración (aéreo o submarino) con una cámara dirigida hacia abajo presenta pocas variaciones de rotación (ángulo de cabeceo y balanceo), además de intentar mantener una altura constante con el suelo, las transformaciones que describan su movimiento debería estar descrita por esos grados de libertad (rotación y translación, sin escala). En este sentido, se utiliza un modelo del sub mosaico que utiliza solamente transformaciones de isometría

(ecuación 4.1). Así mismo, al no contar con datos de navegación, como GPS que no está presente en aplicaciones bajo el agua, el método propuesto presenta una buena aproximación para reducir el error de tamaño en las imágenes del extremo del mosaico. Este proceso se encuentra listado a continuación, además se ilustra gráficamente en la figura 4.12.

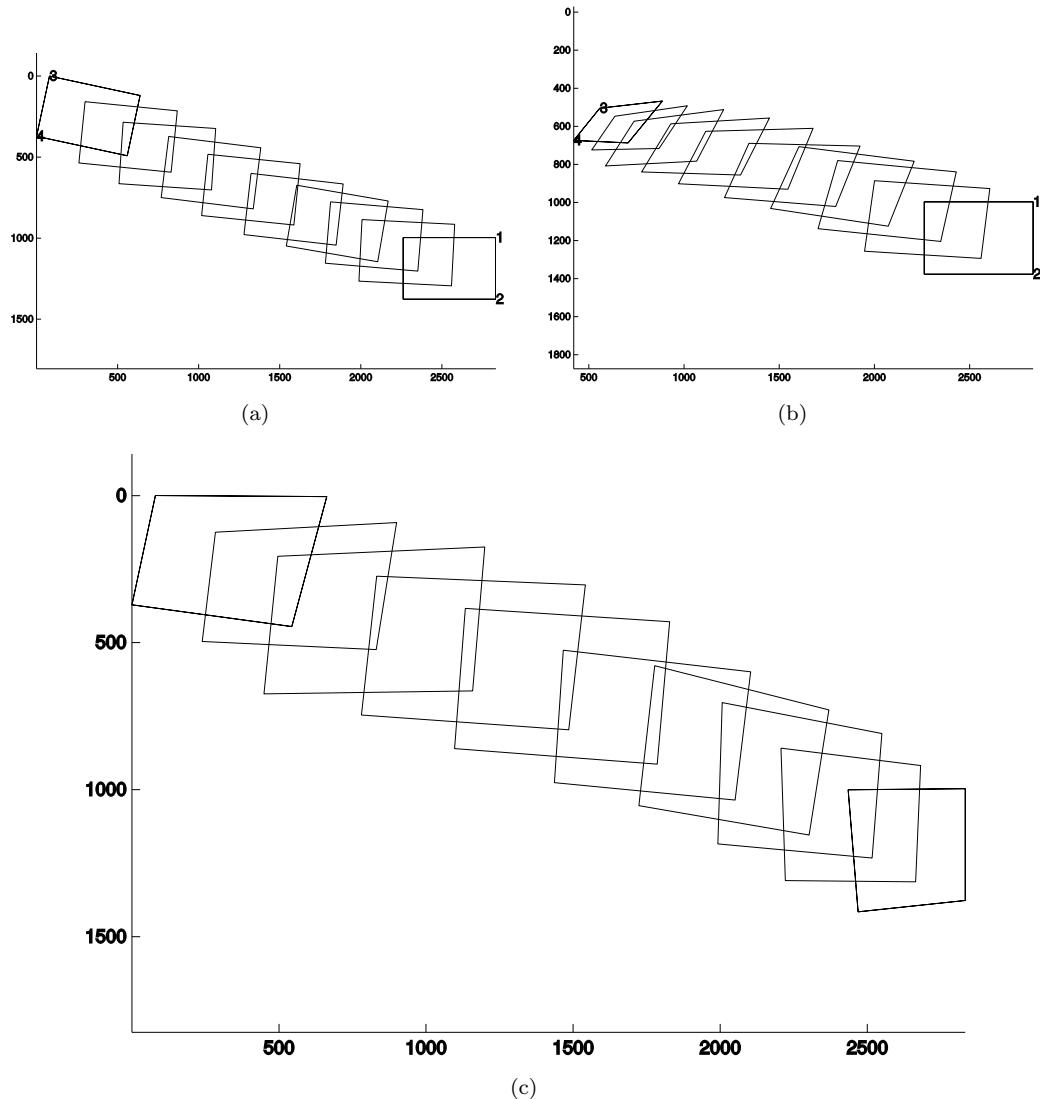


FIGURA 4.12: Corrección euclíadiana. (a) muestra el mosaico usando únicamente transformaciones de isometría, (b) muestra el mosaico usando transformaciones de perspectiva, y (c) muestra el mosaico tras aplicar la corrección. de [32]

1. Alinear imágenes usando transformaciones de Isometría
2. Seleccionar coordenadas del borde del mosaico
3. Alinear imágenes usando transformaciones de perspectiva

4. Seleccionar mismas coordenadas del paso (2)
5. Hallar homografía desde los puntos (2) hasta (4)
6. Aplicar transformación a todas las imágenes del mosaico.

Con respecto a la selección de los puntos frontera del mosaico, el primer par se selecciona considerando aquellos de la primera imagen que se encuentren mas alejados del centro de la última, y para el ultimo par se consideran aquellos de la última imagen mas alejados del centro de la primera.

4.5. Resultados

Resumen

4.6. Resumen

Resumen

Capítulo 5

Unión de imágenes

5.1. Introducción

El objetivo final de la creación de un mosaico, es lograr un mapa que represente de la mejor forma la trayectoria recorrida. Esto es, que sea visualmente congruente y que no presente ningún tipo de discontinuidades de modo que todo parezca una misma imagen. De las primeras etapas se manifiestan muchos errores producto de los problemas ya planteados, si bien a lo largo del proceso éstos se intentan reducir lo mas posible, siempre es necesaria la etapa final de fusión para lograr los objetivos propuestos.

En esta sección se describen los algoritmos utilizados para lograr fusionar las imágenes del mosaico. Como ya se especificó, fueron seleccionados aquellos que presentaron resultados importantes en diversos estudios externos, y se implementaron en conjunto para lograr resultados mucho mas robustos. Estos se detallan a continuación en el orden de aplicación sobre el mosaico: linea de corte, ajuste de color y finalmente la fusión ponderada.

Por ultimo, al final del capítulo se muestran los resultados con su respectivo análisis de aplicar cada uno de los algoritmos aquí descritos.

5.2. Línea de costura

Este es el primer algoritmo aplicado al mosaico luego de todas las correcciones geométricas, ya que el resto de métodos requieren conocer de antemano los límites de cada imagen. A diferencia de los anteriores, este tipo de algoritmos es el único que toma en cuenta la información que comparten las imágenes en el área que tienen en común, con lo cual su implementación logra corregir la mayor cantidad de imperfecciones.

5.2.1. Corte por grafo

Este es un método derivado de la teoría de grafos, donde la idea es separar un grafo con conexiones simples en dos grafos separados con un mínimo costo de separación. Se define un grafo $\mathcal{G} = \langle \mathcal{N}, \mathcal{E} \rangle$ como un conjunto de nodos \mathcal{N} , conectados por enlaces \mathcal{E} . Cada enlace conecta dos nodos y tiene asociado un costo o peso $\mathcal{W}(p, q)$ $p, q \in \mathcal{N}$ — cuando se habla de conexión simple, se refiere a que el costo en los enlaces está asociado para ambas direcciones $\mathcal{W}(p, q) = \mathcal{W}(q, p)$ —. Se dice que dos grafos están separados si no se tiene ningún enlace que conecte dos nodos entre los grafos. Definimos \mathcal{S} y \mathcal{T} como los grafos separados que se tienen luego de aplicar el corte en \mathcal{G} . El método para determinar el mejor corte, se basa en encontrar el camino entre los enlaces que logra separa un grafo en dos, con el mínimo costo de corte [33], donde el costo del corte es la suma de los pesos de todos los enlaces del camino seleccionado.

En el conjunto de nodos presentes en el grafo se cuentan con dos especiales llamados nodos terminales, el inicio (**I**) y el final (**F**), donde el resto de píxeles en la imagen corresponden con un nodo no terminal. En aplicaciones de visión por computadora, cuando se desea unir dos imágenes en una región de intersección, lo que se busca es lograr un etiquetado de píxeles que permita distinguir que nodos corresponden a cada imagen en el área de intersección.

Se presenta en la ecuación 5.1 la función de coste que etiqueta los nodos, y minimiza el costo del corte.

$$C(f) = \sum_{p \in \mathcal{N}} D_p(f_p) + \sum_{p,q \in \mathcal{N} - \{\mathbf{I}, \mathbf{F}\}} \mathcal{W}_{p,q}(f_p, f_q) \quad (5.1)$$

Donde p es un nodo que pertenece al conjunto de nodos no terminales $\mathcal{N} - \{\mathbf{I}, \mathbf{F}\}$. El término $D_p(f_p)$ es el costo de asignar una etiqueta f_p ($f_p \in \{0, 1\}$) al nodo p — en este caso una etiqueta binaria, asociando el píxel a una imagen u otra —. El término $\mathcal{W}_{p,q}(f_p, f_q)$ es el costo de asociar una etiqueta al nodo p y una distinta al nodo q .

Una gran diferencia de intensidades entre píxeles adyacentes representa un fuerte indicador de la existencia de un borde o contorno entre dos objetos, es decir, que el costo de un enlace se puede definir como el inverso de la diferencia entre la intensidad de los píxeles que conecta. Siendo $I(p)$ la intensidad de un píxel p , se defina el coso de cada enlace como:

$$\mathcal{W}_{p,q} = 255 - |I(p) - I(q)| \quad (5.2)$$

Si bien se consideran los parámetros necesarios con esa función, se obtienen resultados mucho mas robustos usando una función exponencial [34]:

$$\mathcal{W}_{p,q} = e^{\left(\frac{255 - |I(p) - I(q)|}{2\sigma}\right)} \quad (5.3)$$

Donde σ es la desviación estándar de la imagen, y siendo válido para los casos en los que $f_p \neq f_q$. Esto indica que dos nodos continuos pertenecen a distintos grafos $(\mathcal{S}, \mathcal{T})$ y a su vez que son separados por la linea de corte.

Refiriéndonos a la figura 5.1, se observa el proceso de modelar una imagen mediante un grafo con conexiones simples y usando un esquema de 4 vecindad — 4 vecindad solo considera a los píxeles superior, inferior y laterales como vecinos — , donde cada cuadro está compuesto por el inverso de la diferencia de intensidades entre dos imágenes, representado en escala de grises. Al final se tiene una imagen compuesta por dos grafos separados, donde $\mathbf{I} \in \mathcal{S}$ y $\mathbf{F} \in \mathcal{T}$ donde a cada nodo del grafo se le asigna un valor binario dependiendo de la etiqueta resultante el algoritmo de minimización del costo de corte.

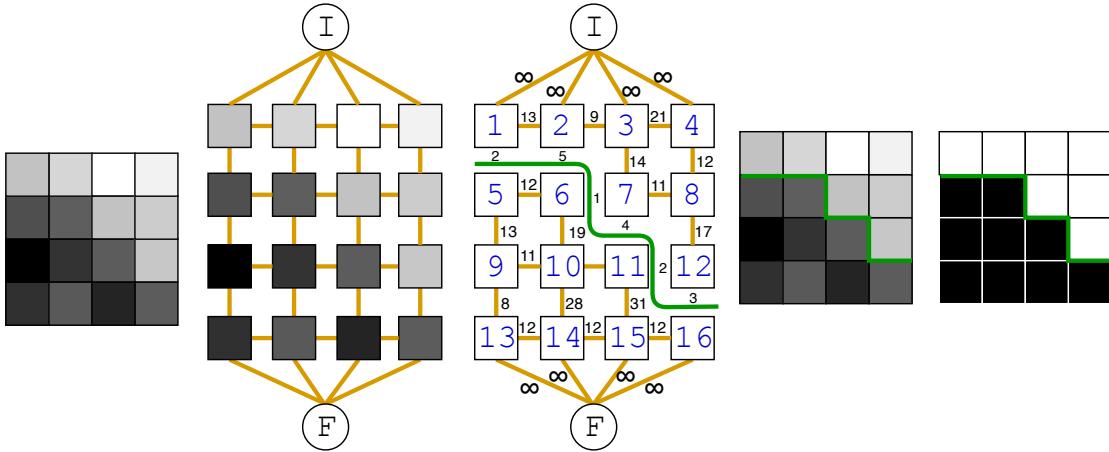


FIGURA 5.1: De izquierda a derecha: imagen original, creación de nodos y enlaces, se asignan los pesos y se halla la linea de corte, finalmente se binariza la imagen según las etiquetas para crear una mascara.

5.3. Corrección de color

Cuando se tienen imágenes capturadas desde distintos puntos de vista, se suelen tener en la composición final cambios de intensidades, debido a cambios de exposición de luz en la escena. Por ellos es necesario implementar algoritmos que logran reducir la diferencia de intensidades y color entre los bordes de las imágenes. Para esto se implementa un algoritmo de mapeo de color llamado método de *Reinhard* [35] que trabaja en el espacio de color CIELAB, descrito a continuación.

Espacio de color CIELAB

Conocido en ocasiones con el nombre de $l^*a^*b^*$, es un espacio derivado del *CIE 1931 XYZ*, creado por la comisión internacional de la iluminación CIE (del francés: Comission Internationale de l'Éclairage).

Este es un espacio de color tridimensional descrito por los ejes a , que extiende desde verde ($-a$) hasta rojo ($+a$), el eje b que se extiende desde azul ($-b$) hasta amarillo ($+b$), y la luminancia l que varía desde negro ($-l$) hasta blanco ($+l$). Esta representación se puede observar en la figura 5.2.

Debido a que la cantidad de valores para representar los colores en este espacio es menor que en RGB, es mas rápido hacer correcciones de color en LAB, ya

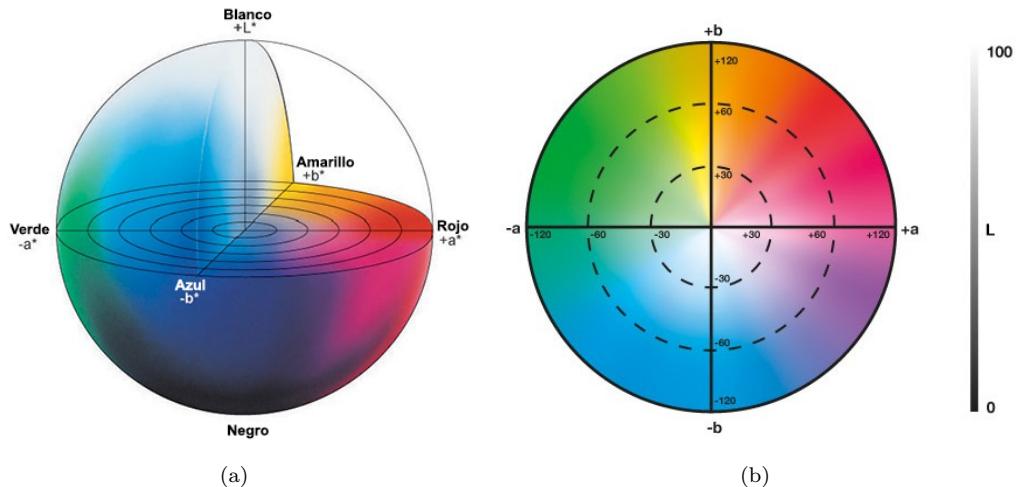


FIGURA 5.2: Espacio de color CIELAB. En (a) se aprecia el diagrama tridimensional, mientras que en (b) una vista superior del rango de colores con luminancia fija.

que un cambio en la cantidad de valor de una variable produce un cambio con importancia visual. Además de permitir trabajar la iluminación de la imagen por separado de los colores.

5.3.1. Método de Reinhard

El objetivo de esta transformación de color es lograr que la distribución de los puntos en el espacio de color LAB se transfieran entre imágenes. Para esto se utiliza la información de la media y desviación estándar sobre cada canal (L, A y B).

Para un par de imágenes I e I^* , donde I es la imagen origen e I^* es la imagen que se quiere corregir, se calcula la media y desviación estándar para cada canal.

$$\begin{array}{ll}
 \langle l \rangle & \langle l^* \rangle \\
 \langle a \rangle & \langle a^* \rangle \\
 \langle b \rangle & \langle b^* \rangle
 \end{array}
 \quad
 \begin{array}{ll}
 \sigma_l & \sigma_l^* \\
 \sigma_a & \sigma_a^* \\
 \sigma_b & \sigma_b^*
 \end{array}$$

Luego se ajusta la distribución de la imagen objetivo restandole la media obtenida en el paso anterior:

$$\begin{aligned} l^* &= l^* - \langle l^* \rangle \\ a^* &= a^* - \langle a^* \rangle \\ b^* &= b^* - \langle b^* \rangle \end{aligned}$$

Luego se escalan los valores en función a las desviaciones estándar:

$$\begin{aligned} l^* &= \frac{\sigma_l^*}{\sigma_l} l^* \\ a^* &= \frac{\sigma_a^*}{\sigma_a} a^* \\ b^* &= \frac{\sigma_b^*}{\sigma_b} b^* \end{aligned}$$

Finalmente se le suma la media, pero en este caso de la imagen de origen (imagen del color deseado):

$$\begin{aligned} l^* &= l^* + \langle l \rangle \\ a^* &= a^* + \langle a \rangle \\ b^* &= b^* + \langle b \rangle \end{aligned}$$

Para lograr un mapeo de color correcto es necesario obtener la información de desviación estándar y media únicamente en el área de intersección entre las imágenes $I \cap I^*$. De esta forma se logra igualar estas regiones de la imagen que deben corresponder a la misma escena.

5.4. Fusión de imágenes

Una vez se logre determinar la mejor linea de corte, y aplicar una corrección de color, es posible que aun se cuenten con transiciones entre imágenes con cierto nivel de discontinuidad. En este caso es conveniente aplicar algoritmos que permitan una transición suavizada entre dichos bordes.

Se tienen varias versiones de estos algoritmos que serán explicados a continuación: fusión ponderada simple o bajo un esquema piramidal.

5.4.1. Fusión ponderada

Este tipo de algoritmos realiza una fusión que consiste en realizar un suma ponderada, en la cual se modifica el peso de los píxeles en función de la distancia con el borde de su imagen. Es decir, se realiza una suma de los valores de las imágenes (I_1, I_2) en el área de intersección ($I_1 \cap I_2$) para cada canal, dándole menor peso a los píxeles que se encuentren mas cercanos del extremo de la imagen. Cabe destacar que esta suma se normaliza para asegurar que no se supere el rango 0 – 255 con el cual se representan los valores de la imagen final. La ecuación para la suma ponderada se muestra a continuación:

$$F = (1 - \alpha) \cdot I_1 + \alpha \cdot I_2$$

Donde el parámetro α varia en el rango $0 \rightarrow 1$, en este caso en función a la distancia, tal y como se ilustra en la figura 5.3.

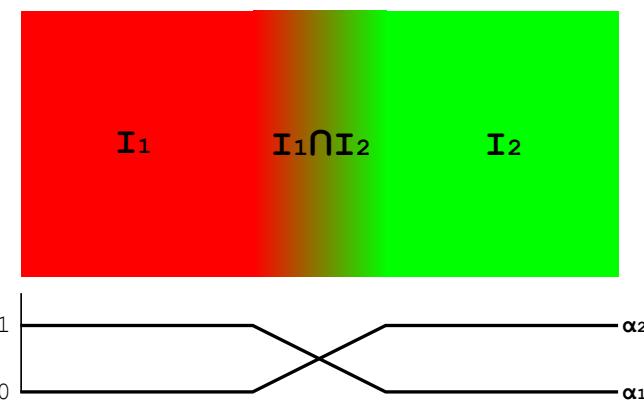


FIGURA 5.3: Fusión ponderada simple, donde α_1 y α_2 corresponden con el factor multiplicativo de las imágenes 1 y 2 respectivamente ($\alpha_1 = 1 - \alpha_2$).

El algoritmo de fusión ponderada bajo el esquema de transición presenta ciertas desventajas, entre estas el efecto fantasma, en el que se observan duplicados de los objetos en la escena pero con cierto grado de desvanecimiento. Si bien no será implementado, el estudio de su funcionamiento es necesario para comprender el algoritmo de fusión que trabaja bajo un esquema piramidal.

5.4.2. Fusión ponderada piramidal

El método planteado anteriormente propone una transición suavizada para imágenes que tengan una región en común. Si bien es una primera aproximación al problema aquí planteado, los algoritmos que se pretenden utilizar en etapas previas (línea de corte) eliminan ésta área de intersección, quedando don imágenes con solo un borde en común.

Esta técnica es llamada fusión piramidal o fusión multibanda, y su proceso consiste en fusionar las imágenes para distintos niveles de escala. Refiriéndonos a la figura 5.4, se crea para una misma imagen distintos niveles de escala (G_i), donde cada nivel corresponde con una reducción a un cuarto del área del nivel anterior. La cantidad de niveles en la pirámide corresponde con la cantidad de bandas que se quieren fusionar. Es importante mencionar que el proceso de compresión y posterior expansión de una imagen, se aproxima al proceso de aplicar un filtro gaussiano, con lo cual a cada nivel creado G_i recibe el nombre de gaussiana, resultando lo que se denomina una pirámide de gaussianas, tal y como se ilustra en la figura 5.4 con 5 niveles.

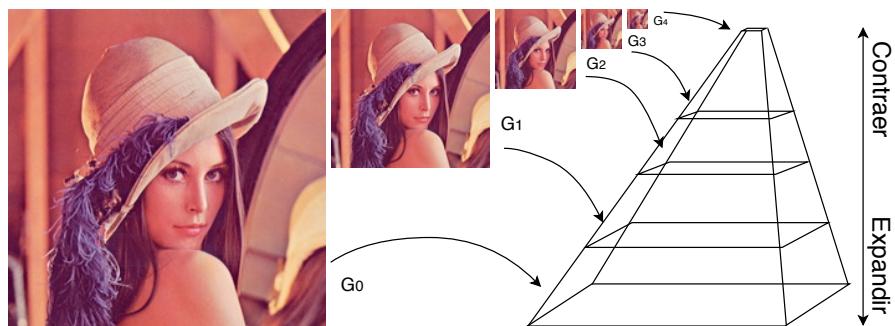


FIGURA 5.4: Construcción de pirámide de gaussianas

Una vez se tiene la pirámide de gaussianas, se elabora una pirámide de laplacianas utilizando *DoG*, es decir, cada nivel de esta pirámide se construye restando un nivel de la pirámide gaussiana con su nivel siguiente pero expandido (para lograr las mismas dimensiones). Al restar una imagen con ella misma pero aplicada un filtro gaussiano — que remueve las componentes en bajas frecuencias —, se logra una imagen que resalte las componentes en las frecuencias altas. Esta composición se puede observar en la figura 5.5, donde a cada imagen laplaciana L_i se le sumó una constante para efectos de visualización, ya que presenta valores muy bajos al aplicar únicamente la resta.

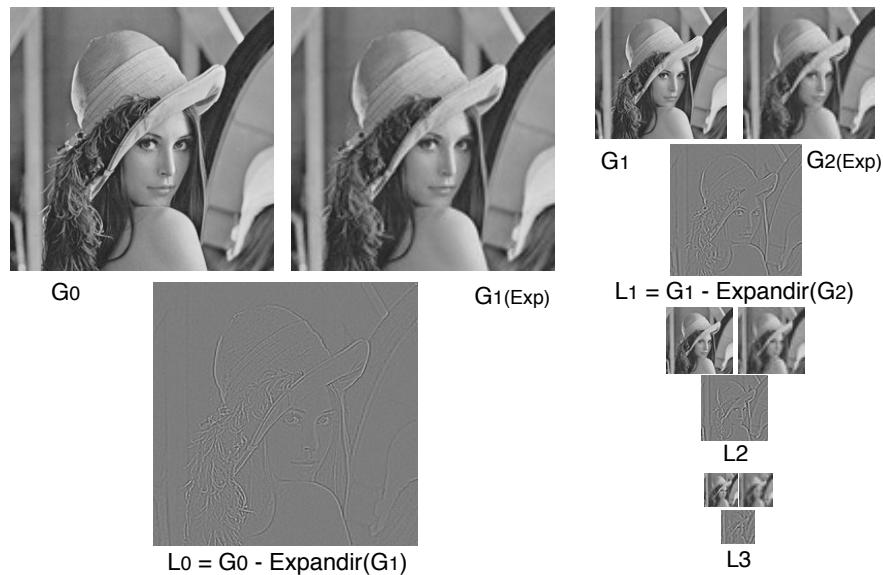


FIGURA 5.5: Construcción de pirámide de laplacianas

Como ya se mencionó, este proceso se aplica para imágenes que no comparten un área de superposición, con lo cual es necesario definir mediante una máscara binaria los límites de cada imagen. Convenientemente, el proceso de determinar la línea de corte ya nos ofrece esta máscara mediante el proceso de etiquetado, con lo cual se utiliza para definir las fronteras en esta fusión.

La pirámide gaussiana que se elaboró en un principio para cada imagen a fusionar, debe también crearse para la máscara, ya que la fusión debe realizarse en cada nivel de la pirámide. Finalmente para mostrar la imagen final en la resolución original, el proceso de fusión debe realizarse escalando cada nivel hasta las dimensiones originales.

Éste proceso consiste en expandir el último nivel de la pirámide — con su respectiva máscara—, una vez expandido se le suma el laplaciano correspondiente, devolviéndole la componente en alta frecuencia que le fue removida. En este punto se tiene el par de imágenes expandidas un nivel, además de la máscara aun afectada por el efecto del filtro gaussiano, con lo cual se efectúa una suma ponderada entre las dos imágenes similar que en la sección 5.4.1, pero en este caso se utiliza la máscara difuminada para determinar el peso de la suma, y donde el inverso de la máscara determina la ponderación de la segunda imagen.

En la figura 5.6 se puede observar el resultado de aplicar una fusión bajo el esquema piramidal.

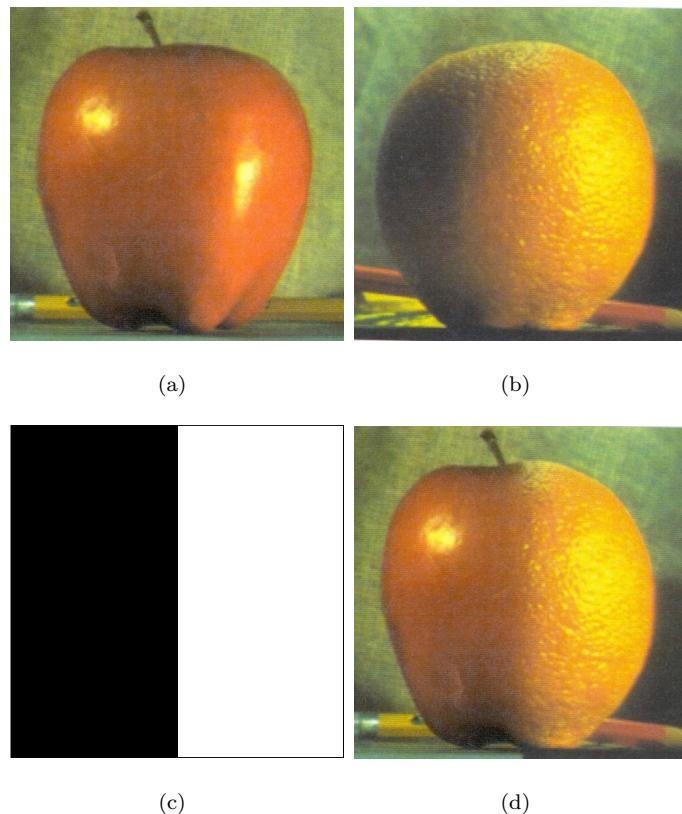


FIGURA 5.6: Fusión piramidal. Donde (a) y (b) representan las imágenes que se desean unir, (c) es la máscara para la fusión y (d) es el resultado final utilizando 9 bandas¹.

5.5. Resultados

test

5.6. Resumen

test

¹ <https://compvisionlab.wordpress.com/2013/05/13/image-blending-using-pyramid/>

Capítulo 6

Conclusiones y trabajos futuros

Conclusiones

Bibliografía

- [1] Danilo Díaz Tarascó. Desarrollo de sistema de telemetría para teleoperación de robot submarino, 2016.
- [2] Said Alexander Alvarado Marín. Caracterización, instrumentación y control de robot cuadricóptero volador, 2017.
- [3] Paul R. Wolf. *Elements of Photogrammetry (Second Edition)*. McGraw-Hill Higher Education, 1983.
- [4] D.P. Capel. *Image Mosaicing and Superresolution*. Springer Science and Business Media, 2004.
- [5] Sherin Ghannam and A. Lynn Abbott. Cross correlation versus mutual information for image mosaicing. *International Journal of Advanced Computer Science and Applications*, 2013.
- [6] Kostas Berberidis y Irene Karybali. A new efficient cross-correlation based image registration technique with improved performance. *European Signal Processing Conference*, 2002.
- [7] Amaury Dame y Eric Marchand. Video mosaicing using a mutual information-based motion estimation proces. *IEEE International Conference on Image Processing (ICIP)*, 2011.
- [8] Hans P. Morevec. Towards automatic visual obstacle avoidance. 1977.
- [9] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Computer Science Department, University of British Columbia*, 2004.
- [10] Tuytelaars T. y Van Gool L Bay, H. Surf: Speeded up robust features. *European Conference on Computer Vision*, 2006.

- [11] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. *Proceedings of the 9th European Conference on Computer Vision - Volume Part I*, 2006.
- [12] Vincent y Strecha Christoph y Fua Pascal Calonder, Michael y Lepetit. Brief: Binary robust independent elementary features. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*. Springer-Verlag, 2010.
- [13] Kurt Konolige y Gary Bradski Ethan Rublee, Vincent Rabaud. Orb: an efficient alternative to sift or surf. *International Conference on Computer Vision*, 2011.
- [14] Andrew J. Davison Pablo Fernandez Alcantarilla, Adrien Bartoli. Kaze features. *European Conference on Computer Vision*, 2012.
- [15] Pablo Fernández Alcantarilla. Fast explicit diffusion for accelerated features in nonlinear scale spaces. *British Machine Vision Conference (BMVC)*, 2013.
- [16] G. Wiet T. Pan y K. Huang J. Prescott, M. Clary. Automatic registration of large set of microscopic images using high-level features. *IEEE International Symposium on Biomedical Imaging*, 2006.
- [17] G.R. Keller H. Huang y V. Kreinovich H. Xie, N. Hicks. An idl/envi implementation of the fft-based algorithm for automatic image registration. *Computers and Geosciences*, 2003.
- [18] Z. Zhang y H. Tang F. Yang, L. Wei. Image mosaic based on phase correlation and harris operator. *Journal of Computational Information Systems*, 2012.
- [19] U.C. Pati A. Pandey. A novel technique for non-overlapping image mosaicing based on pyramid method. *IEEE India Conference*, 2013.
- [20] Fan Zhang and Feng Liu. Parallax-tolerant image stitching. *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.
- [21] Shahriar Negahdaripour y Mohammad Mahoor Nuno Gracias, Art Gleason. Fast image blending using watersheds and graph cuts. *Image and Vision Computing* 27(5):597-607, 2009.
- [22] Y. Boykov y V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001.

- [23] Fabio Pazzaglia y Carlo Colombo Fabio Bellavia, Marco Fanfani. Piecewise planar underwater mosaicing. *OCEANS'15 MTS/IEEE Genova*, 2015.
- [24] Fabio Bellavia and Carlo Colombo. Estimating the best reference homography for planar mosaics from videos. *VISAPP - 10th International Conference on Computer Vision Theory and Applications; VISIGRAPP, Proceedings*, 2015.
- [25] Armagan Elibol, Jinwhan Kim, Nuno Gracias, and Rafael García. Fast underwater image mosaicing through submapping. *Journal of Intelligent and Robotic Systems*, 85:167–187, 2017.
- [26] Tinne Tuytelaars y Krystian Mikolajczyk. Local invariant feature detectors: a survey. *Foundations and Trends® in Computer Graphics and Vision*, 2008.
- [27] T Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 1998.
- [28] Kwang-Ting Cheng Xin Yang. Local difference binary for ultrafast and distinctive feature description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.
- [29] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.
- [30] David Muja, Marius y Lowe. Scalable nearest neighbor algorithms for high dimensional data. 36:2227–2240, 11 2014.
- [31] Richard Hartley y Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press New York, NY, USA, 2003.
- [32] Armagan Elibol, Nuno Gracias, and Rafael Garcia. *Efficient Topology Estimation for Large Scale Optical Mapping*, volume 82. 01 2012.
- [33] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001.
- [34] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graph-cut textures: Image and video synthesis using graph cuts. *ACM Trans. Graph.*, 2003.
- [35] Bruce Gooch y Peter Shirley Erik Reinhard, Michael Ashikhmin. Color transfer between images. *IEEE Computer Graphics and Applications*, 2001.

Apéndice A

Instalación de librería OpenCV y dependencias

A.1. @sección

A.1.1. @subsección

“Saludo”.