



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 vídeo, o incluso trayectorias más 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

<b>Resumen</b>	<b>I</b>
<b>Agradecimientos</b>	<b>I</b>
<b>Índice de Figuras</b>	<b>IV</b>
<b>Lista de Tablas</b>	<b>V</b>
<b>Acrónimos y Símbolos</b>	<b>VI</b>
<b>1. Introduccion</b>	<b>1</b>
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 . . . . .	4
<b>2. Estado del arte</b>	<b>6</b>
2.1. Sistemas de generación de mosaico . . . . .	6
<b>3. Detección de puntos de interés</b>	<b>7</b>
3.1. Introducción . . . . .	7
3.2. Revisión teórica . . . . .	7
3.2.1. Detectores de características . . . . .	7
3.2.2. Descriptores de características . . . . .	16
3.2.3. Emparejadores . . . . .	16
3.3. Implementación y resultados de modulo comparativo . . . . .	16
3.4. Conclusiones . . . . .	16
<b>4. Alineación de imágenes</b>	<b>17</b>
4.1. Introducción . . . . .	17
4.2. Revisión teórica . . . . .	18

---

4.2.1. Transformaciones geométricas . . . . .	18
4.3. Generación de sub-mosaicos . . . . .	18
4.3.1. Selección de imagen de referencia . . . . .	18
4.3.2. Matriz de transformación promedio . . . . .	18
4.4. Corrección euclidiana . . . . .	18
4.5. Resultados . . . . .	18
4.6. Conclusiones . . . . .	19
4.7. Algoritmos (temporal) . . . . .	21
<b>5. Unión de imágenes</b>	<b>22</b>
5.1. Introducción . . . . .	22
5.2. Línea de costura . . . . .	22
5.2.1. Mapa más cercano . . . . .	22
5.2.2. Corte por grafo . . . . .	22
5.3. Corrección de color . . . . .	23
5.3.1. Ajuste de ganancia . . . . .	23
5.3.2. Método de Reinhard . . . . .	23
5.4. Fusión de imágenes . . . . .	23
5.4.1. Fusión ponderada . . . . .	23
5.4.2. Fusión piramidal . . . . .	23
5.5. Resultados . . . . .	23
5.6. Conclusiones . . . . .	23
<b>6. Conclusiones y trabajos futuros</b>	<b>24</b>
<b>A. Instalación de librería OpenCV y dependencias</b>	<b>27</b>
A.1. @sección . . . . .	27
A.1.1. @subsección . . . . .	27

# Índice de figuras

1.1. Robot móvil OpenROV . . . . .	3
1.2. Logo del software Hugin . . . . .	3
3.1. Puntos de interés . . . . .	8
3.2. Deteccion de esquinas . . . . .	9
3.3. Deteccion de esquinas . . . . .	10
3.4. SIFT - Espacio de escalas . . . . .	11
3.5. SIFT - Espacio de escalas . . . . .	12
3.6. SURF - Espacio de escalas . . . . .	13
3.7. ORB - Espacio de escalas . . . . .	14
3.8. KAZE - Espacio de escalas . . . . .	15
4.1. Paralaje - Efectos en el cambio de punto de vista . . . . .	17

# Índice de Tablas



# Acrónimos y Símbolos

## ***Dedicatoria***

*A @personasImportantes, por @razonesDedicatoria.*

# Capítulo 1

## Introduccion

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 mu tiempo. Donde una de las aplicaciones mas demandantes, es la tarea de reconstruir un mapa 2D de la superficie mapeada por los robots utilizados. En el presente capitulo se pretende introducir los trabajos previos y avances que se han tenido en el desarrollo de este tipo aplicaciones, tanto en el *GIDM* como a nivel mundial, y 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

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 ubicación es una constante.

La elaboración de mosaicos para la construcción de mapas del suelo, se ha desarrollado incluso antes desde la era digital de la computadoras. Desde que el

proceso de registrar fotografías ha existido, se comenzaron a usar para elaborar mapas topográficos [1], 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, y se cubrían mayores áreas en menor cantidad de tiempo, pero debido a que no se alcanzaban alturas tan altas, 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 aun mas 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.

Del mismo modo, distintos centros de investigación en el área de la física, robótica y visión por computadora, han aplicado sus esfuerzos en desarrollar algoritmos para la realización de estos mapas en ambientes mas desafiantes como lo son el fondo marino [2, 3, 4, 5]. 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) o vehículos autónomos submarinos *AUV* (del inglés: Automated Underwater Vehicles).

## 1.2. Justificación y planteamiento del problema

En el *GIDM* se han realizado grandes avances en el desarrollo de equipos y plataformas robóticas, específicamente para su uso en aplicaciones submarinas. En este sentido, se cuenta con un vehículo submarino llamado OpenROV<sup>1</sup>, el cual se puede apreciar en la figura 1.1. OpenROV es un robot operado remotamente de baja albergadura diseñado especialmente para operar bajo el agua.

---

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

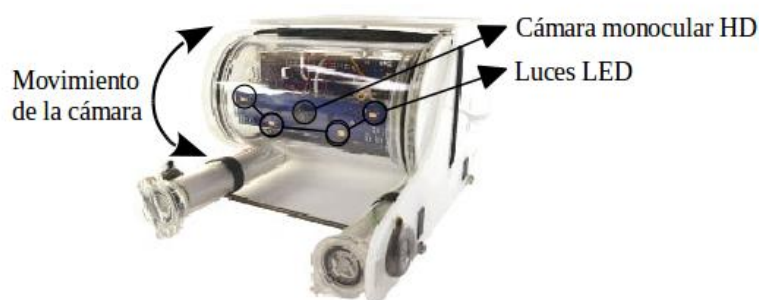


FIGURA 1.1: Robot móvil OpenROV

En el *GIDM* actualmente se utilizan mecanismos manuales para la elaboración de estos mapas, en específico, se hace uso de *softwares* como Hugin<sup>2</sup>. Este es un programa de código abierto y gratuito bajo licencia GPL<sup>3</sup>, el cual está dedicado a la generación de imágenes panorámicas, incluyendo funciones para el recorte, alineación, corrección de color; además de algoritmos para la optimización de parámetros en la cámara, y corrección de distorsión. Si bien este software está 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.



FIGURA 1.2: Logo del software Hugin

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 simultánea *SLAM*, haciendo uso de las herramientas y robots existentes en el laboratorio, es necesario 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.

---

<sup>2</sup>[hugin.sourceforge.net/](http://hugin.sourceforge.net/)

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

## 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 entorno recorrido por robot, aereo o submarino.

### 1.3.2. Objetivos Específicos

- Análisis comparativo de metodos vigentes en la reconstruccion de mosaicos 2D, a partir de imagenes y videos de entrada.
- Implementacion de modulo de preprocesamiento y correccion de entrada.
- Analisis comparativo de metodos de deteccion y description de puntos clave.
- Implementación de módulo de alineación de imagenes mediante la detección de puntos clave.
- Cuantificar el error de reproyeccion y distorsion 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. En ese mismo sentido, se describen los módulos principales que componen este tipo de sistemas.

El **Capítulo 3** comprende una revisión teórica en la cual se describe el funcionamiento de los algoritmos detectores, descriptores y emparejadores de características, así como también 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 capitulo 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 presentan 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 posibles implementaciones y recomendaciones que pueden aportar mejoras y/o permitir la continuación del proyecto aquí planteado.

# Capítulo 2

## Estado del arte

En este capítulo se presenta una revisión teórica del estado actual de las aplicaciones e investigaciones que se han desarrollado en el área de procesamiento de imágenes, aplicado a la construcción de mosaicos. Además se presenta una reseña histórica de la evolución de dichos métodos. 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.

### 2.1. Sistemas de generación de mosaico



# Capítulo 3

## Detección de puntos de interés

### 3.1. Introducción

### 3.2. Revisión teórica

#### 3.2.1. Detectores 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 [6] 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 pixel). 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: Ejemplo de caracterización de puntos en una imagen

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 encontrar la ubicación exacta. 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 de puntos de interés llamado Detector de esquinas de Harris [7] (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.

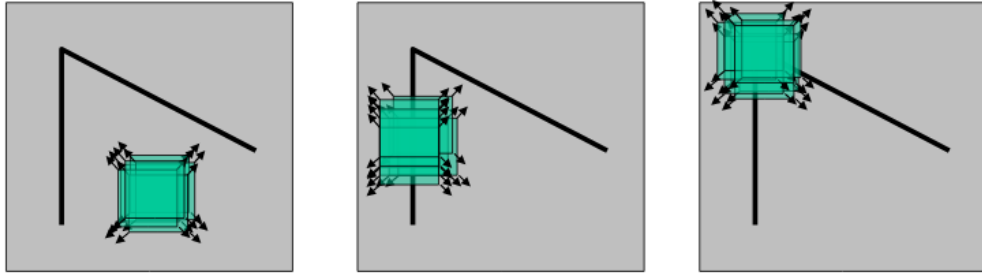


FIGURA 3.2: Desplazamiento de la ventana de búsqueda, para de detección de esquinas

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.

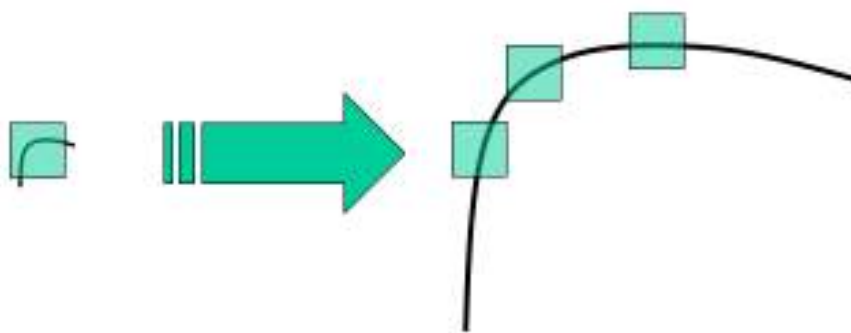


FIGURA 3.3: Esquinas escalas

Con el fin de conseguir detectar los mismos puntos ante cambios en la escala de la imagen, Lindeberg, T. [8] 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 variable 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 calculo 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 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.

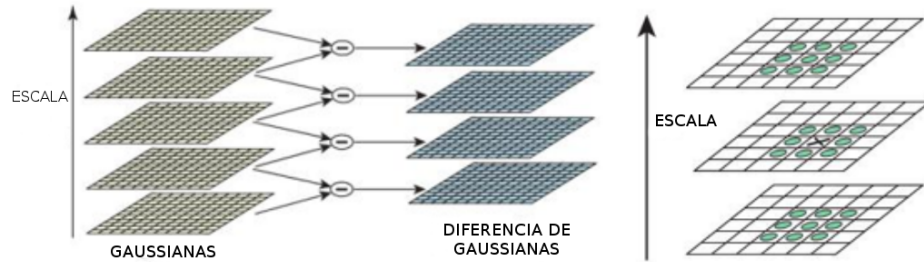


FIGURA 3.4: Detección de máximos en el espacio de la escala

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.

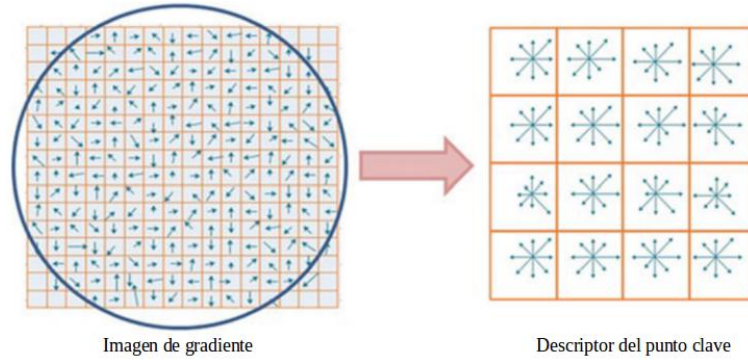


FIGURA 3.5: Detección de máximos en el espacio de la escala

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 más provechoso para aplicaciones embebidas que demanden mayor velocidad de cómputo y menor uso de recursos, como por ejemplo *SLAM* (del inglés: Simultaneous Localization and Mapping). 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.

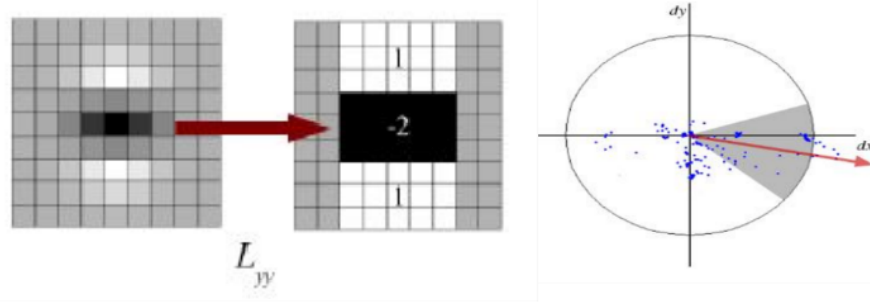


FIGURA 3.6: Detección de máximos en el espacio de la escala

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  $60^\circ$ . 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* [11] (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).

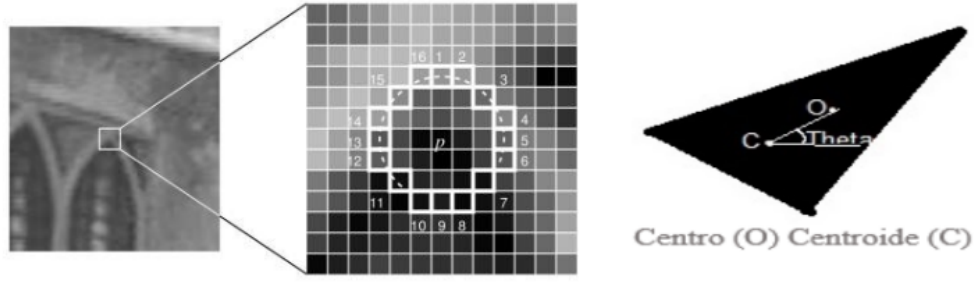


FIGURA 3.7: Detección de máximos en el espacio de la escala

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 BRIEF produce una palabra de  $n$ -bits usando el algoritmo *Local Binary 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 [12] 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.

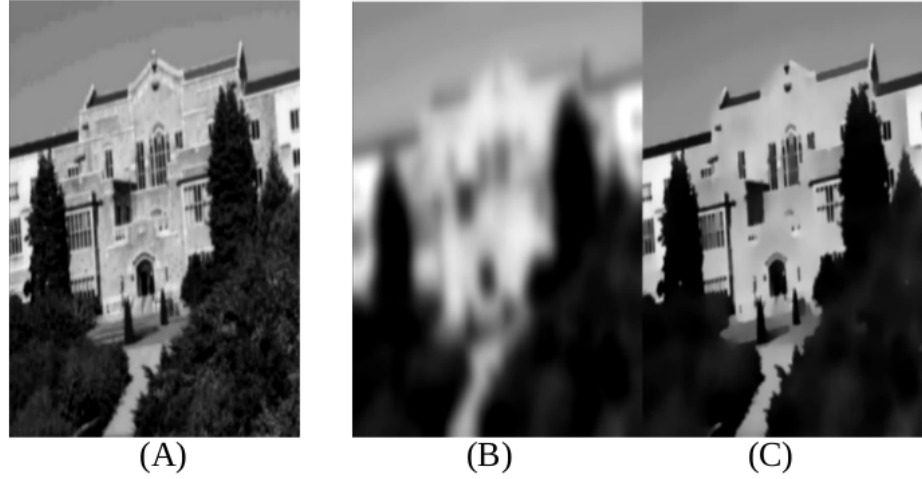


FIGURA 3.8: Detección de máximos en el espacio de la escala

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* [13] (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 cómputo 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  $60^\circ$ .

Por otro lado, la versión acelerada *A-KAZE* emplea un descriptor basado en una versión modificada del algoritmo de diferencia local binaria *LDB* [14] (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. Descriptores de características**

Descriptores

### **3.2.3. Emparejadores**

Emparejadores

## **3.3. Implementación y resultados de modulo comparativo**

## **3.4. Conclusiones**

Resumen

# Capítulo 4

## Alineación de imágenes

### 4.1. Introducción

Introducción

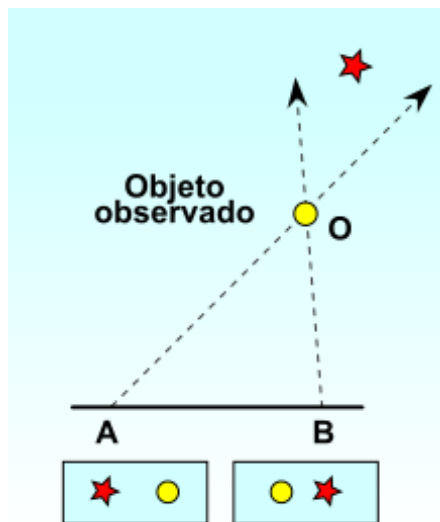


FIGURA 4.1: Efectos en el cambio del punto de vista.

Desde el punto de vista **A**, se observa la estrella a la izquierda del círculo. Mientras que desde el punto **B** la estrella es observada a la derecha del círculo.

## **4.2. Revisión teórica**

Prueba

### **4.2.1. Transformaciones geométricas**

## **4.3. Generación de sub-mosaicos**

Prueba

### **4.3.1. Selección de imagen de referencia**

Prueba

### **4.3.2. Matriz de transformación promedio**

Prueba

## **4.4. Corrección euclidiana**

Prueba

## **4.5. Resultados**

Resumen

## 4.6. Conclusiones

Resumen



## 4.7. Algoritmos (temporal)

---

### Algoritmo 1: Calculo de matriz de homografía promedio

---

```

1  mientras no se alcanza el maximo de iteraciones hacer
2      seleccionar 4 puntos aleatorios del primer sub-mosaico;
3      seleccionar los 4 puntos correspondientes en el segundo sub-mosaico;
4      calcular el punto medio para cada par de puntos correspondientes;
5      calcular la transformacion desde los puntos del primer sub-mosaico hasta los
        puntos medios;
6      aplicar transformacion en el primer sub-mosaico;
7      calcular error de distorsión en el primer sub-mosaico;
8      si el error es menor que el mas bajo obtenido entonces
9          guardar el error como el mas bajo;
10         guardar la matriz de transformación como la mejor;
11     en otro caso
12         restaurar valores del primer sub-mosaico;
13     fin
14 fin

```

---



---

### Algoritmo 2: Encontrar matriz de transformacion

---

```

15  $I_{i+1} \equiv$  Imagen nueva;
16  $I_i \equiv$  ultima imagen anadida al mosaico;
17  $V_i \equiv$  vecinos de  $I_i$ ;
18 mientras puntos emparejados  $\geq 4$  hacer
19     Emparejar puntos de  $I_{i+1}$  con  $I_i$ ;
20     si  $I_i$  tiene vecinos entonces
21         para cada vecinos de  $I_i$  hacer
22             Emparejar puntos de  $I_{i+1}$  con  $V_i$ ;
23         fin
24     fin
25     descartar malos emparejamientos; aplicar busqueda sectorizada; si puntos
        totales emparejados  $\leq 3$  entonces
26         modificar criterio para descartar;
27         si criterio para descartar llega al minimo entonces
28             terminar ; // no es posible emparejar imagen
29         fin
30     fin
31 fin

```

---

# Capítulo 5

## Unión de imágenes

### 5.1. Introducción

Prueba

### 5.2. Linea de costura

Prueba

#### 5.2.1. Mapa mas cercano

Prueba

#### 5.2.2. Corte por grafo

Prueba



## **5.3. Corrección de color**

Prueba

### **5.3.1. Ajuste de ganancia**

Prueba

### **5.3.2. Método de Reinhard**

Prueba

## **5.4. Fusión de imágenes**

### **5.4.1. Fusión ponderada**

Prueba

### **5.4.2. Fusión piramidal**

Prueba

## **5.5. Resultados**

## **5.6. Conclusiones**

## Capítulo 6

# Conclusiones y trabajos futuros

Conclusiones

# Bibliografía

- [1] Paul R. Wolf. *Elements of Photogrammetry (Second Edition)*. McGraw-Hill Higher Education, 1983.
- [2] N. Gracias y J. Santos-Victor. Underwater mosaicing and trajectory reconstruction using global alignment. *MTS/IEEE Conference and Exhibition*, 2001.
- [3] H Pizarro, O. y Singh. Seabed video mosaicking with matisse: a technical overview and cruise results. *International Offshore and Polar Engineering Conference*, 2004.
- [4] Pizarro O. Singh-H. y Howland J. Eustice, R. Underwater image toolbox for optical image processing and mosaicking in matlab. *International Symposium on Underwater Technology*, 2002.
- [5] Borgetto M.-Opderbecke J. Pessel N. y Rigaud V Allais, A. G. Toward large-area mosaicing for underwater scientific applications. *IEEE Journal of Oceanic Engineering*, 2003.
- [6] Tinne Tuytelaars y Krystian Mikolajczyk. Local invariant feature detectors: a survey. *Foundations and Trends® in Computer Graphics and Vision*, 2008.
- [7] Hans P. Morevec. Towards automatic visual obstacle avoidance. 1977.
- [8] T Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 1998.
- [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] Kurt Konolige y Gary Bradski Ethan Rublee, Vincent Rabaud. Orb: an efficient alternative to sift or surf. *International Conference on Computer Vision*, 2011.
  - [12] y Andrew J. Davison Pablo Fernandez Alcantarilla, Adrien Bartoli. Kaze features. *European Conference on Computer Vision*, 2012.
  - [13] Pablo Fernández Alcantarilla. Fast explicit diffusion for accelerated features in nonlinear scale spaces. *British Machine Vision Conference (BMVC)*, 2013.
  - [14] Kwang-Ting Cheng Xin Yang. Local difference binary for ultrafast and distinctive feature description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

# Apéndice A

## Instalación de librería OpenCV y dependencias

### A.1. @sección

#### A.1.1. @subsección

“Saludo”.