PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

EVALUACIÓN DE ALGORITMOS DE REGISTRO DE IMÁGENES PARA LA GENERACIÓN DE MOSAICOS APLICADOS A IMÁGENES AÉREAS

Tesis para optar el Título de Ingeniero Electrónico, que presenta el bachiller:

Edwin Wilfredo Martinez Auqui

ASESOR: Donato Andrés Flores Espinoza

Lima, Febrero de 2015

RESUMEN

En la presente tesis se realizó la elaboración de un programa de registro de imágenes, la cual hace uso de fotografías obtenidas desde un vehículo aéreo no tripulado (en este caso un hexacóptero), las imágenes son capturadas con dos tipos de lente que nos dan información de la imagen en espectro infrarrojo cercano y rojo visible, estas imágenes son parte de un gran área de cultivo y mediante una ráfaga continua y secuencial de capturas que poseen área en común, se intenta abarcar toda la superficie de interés, con el desarrollo de esta aplicación se busca realizar el mosaico de imágenes aéreas de campos de cultivo.

El proyecto también abarca el uso de diferentes algoritmos de registro de imágenes los cuales serán comparados para observar sus características resultantes, tales como: robustez y velocidad de procesamiento. El resultado final nos facilitará información espacial que será crucial en la toma de decisiones y posterior ejecución de acciones para el mejoramiento del campo de cultivo.

En el primer capítulo se dan alcances generales de la problemática así como una explicación de lo que supone la obtención de un mosaico de imágenes aéreas.

En el segundo capítulo se detalla acerca de las tecnologías actuales para el registro de imágenes, así como un breve repaso de algunos conceptos que nos permitan encaminarnos en la solución del problema.

En el tercer capítulo se expone una breve explicación de los algoritmos a usar en la comparativa propuesta para el registro de imágenes.

Finalmente en el capítulo cuatro se realizan pruebas de rendimiento entre los algoritmos propuestos, además de mostrar los resultados de los mosaicos obtenidos.

El presente trabajo esta dedicado a mis padres y hermanos por su cariño, apoyo y confianza en mí; gracias a ustedes por acompañarme en esta travesía.

Un agradecimiento especial al profesor Andrés Flores por ser un guía y compartir su conocimiento conmigo.

A mis amigos por el ánimo y consejos en momentos difíciles.

ÍNDICE GENERAL

INTRODUCCIÓN

El Perú es un país considerado dentro de los doce con más megadiversidad del mundo ya que se estima que posee entre el 60 y 70% de la diversidad biológica [9]. Sin embargo la situación de pobreza en la población campesina y los pequeños productores agropecuarios han tenido como consecuencia la mala administración de los campos de cultivo y una degradación de la base productiva de las cosechas, así pues se ha generado desequilibrios en los procesos de extracción y regeneración de los recursos naturales, llegando muchas veces a niveles críticos de deterioro como la desertificación, toxicidad en la vegetación y un uso no adecuado de las fuentes de agua.

El intentar promover acciones para el manejo y uso productivo de los recursos de manera óptima es uno de los fines que persigue la agricultura de precisión, que hace uso de tecnologías como el sensado y la geolocalización, además de la captura de imágenes aéreas, las cuales serán de utilidad para tomar acciones pertinentes tales como: irrigación y fertilización de sectores que no tienen el crecimiento adecuado o utilización de pesticidas, etc.

El CIP (Centro Internacional de la Papa) y la PUCP realizan un trabajo conjunto en la cual se busca utilizar tecnologías de sensado remoto para el mejoramiento y la optimización de los cultivos de tubérculos, facilitando a los productores agrónomos de estas tecnologías para que puedan obtener productos para el propio consumo y exportación.

CAPÍTULO 1

GENERACIÓN DE MOSAICO DE IMÁGENES AÉREAS PARA LA AGRICULTURA

1.1 Entorno general.

En la actualidad la agricultura sigue siendo importante como antaño, es uno de los pilares que sustentan el crecimiento de países industrializados, según la FAO (Food and Agriculture Organization of the United Nations) 2500 millones de personas dependen de actividades tales como la agricultura, caza o pesca para su subsistencia, en 2001 las exportaciones agrícolas en EEUU alcanzaron un nivel de 290 000 millones de dólares [2], por tanto el crecimiento acelerado de un país y una reducción de la pobreza esta acompañado del crecimiento agrícola.

Por ello la creciente preocupación de muchos países industrializados por aplicar la ciencia y tecnologías para la consecución de la seguridad alimentaria para todos. Por el lado de la ciencia podemos citar el uso de la genética para reducir el impacto de virus sobre la producción del cultivo o el uso de tecnologías (llamada agricultura de precisión) para mejorar los escenarios de cultivo de bajos recursos (muchas personas producen para su propio consumo), así encontramos organismos internacionales como el FAO o el CIP que procuran mejorar la producción agrícola, garantizando una mejor nutrición a la población.

1.2. Entorno específico.

En la zona andina del Perú encontramos en cuanto a agricultura una importante cantidad de población que se dedica al autoconsumo y también producción industrial, por ello es vital emprender una cultura de tecnificación, además hay importantes tratados comerciales con países del primer mundo, por ello incentivar el uso de tecnologías y capacitación al personal que labora en este rubro están ampliamente avaladas.

El Centro Internacional de la Papa (CIP) es un centro de investigación que promueve el mejoramiento de los cultivos de papa, camote y otros tubérculos, en un trabajo conjunto con la PUCP se busca implementar tecnologías que permitan convertir la papa en uno de los alimentos básicos para la civilización, los tubérculos poseen un alto contenido proteico pero debido a su volumen son muy difíciles y costosos de

transportar, por ello se convierten en alimentos locales que consume mucha población pobre.

1.3. Agricultura de precisión.

Durante años los sistemas de producción agropecuarios estaban implantadas para obtener los mayores rendimientos, en especial en el sector vegetal, ya que en esta los factores como la temperatura o el clima juegan un papel importante. Es así que la agricultura tendió a desarrollarse de diversas formas, tales como la agricultura intensiva, la ecológica, la orgánica, la familiar y últimamente la de precisión. Esta última emergió en Argentina en el año 1995, de la mano de la INTA(Instituto Nacional de Tecnología Agropecuaria). Este tipo de agricultura posibilitó al agricultor a poder medir, analizar y manejar los campos de una forma más óptima.

Las áreas con las cuales se apoya esta son la estadística, la informática, la electrónica y la mecánica como también al uso de herramientas tecnológicas como son: el posicionamiento global, los dispositivos de distribución de riesgo, los sensores climatológicos y de cultivo [6]. Todo esto nos permite obtener información que puede ser recopilada, interpretada y aplicada. Este último, después de haber convertido esta información en conocimiento dentro de un mismo lote. Para que así sea más eficiente el trabajo en grupos o las labores en el campo.

Continuando con la idea anterior, los privilegios o mejor dicho las fortalezas que esta le ofrece al productor son tres. En primer lugar, la reducción de costo, implica dejar de utilizar insumos en grandes cantidades debido a que con la información recogida permite conocer la cantidad necesaria a emplear. En segundo lugar, obtener mayores rendimientos a partir del mismo nivel de insumos, esto implica aprovechar al máximo los insumos que se posee sin desperdiciar nada. Aplicarlos en lugares en los que más se necesite debido a las características que el suelo posee en tal zona del lote. Por último, mayor calidad en la cosechas (esta es el resultado de los dos anteriores), ya que aplicando los insumos con los requerimientos del lote se llegará a una buena cosecha de vegetales [5].

En conclusión, la agricultura de precisión es una buena alternativa para el mejoramiento de las cosechas de los agricultores a través del empleo de tecnologías que permitan el desarrollo de estas. Pero que a su vez resultaría menos satisfactorio para los agricultores pequeños o medianos ya que a parte de que consideren los

costos de los equipos altos, la mayoría de estos no posee un computadora, además necesitarían de capacitación continua. Pero se espera que para un tiempo no muy lejano esto se vuelva más universal, ya que en América Latina se esta observando un progreso continuo.

En el siguiente cuadro podemos observar el ciclo que se cumple en la agricultura de precisión:

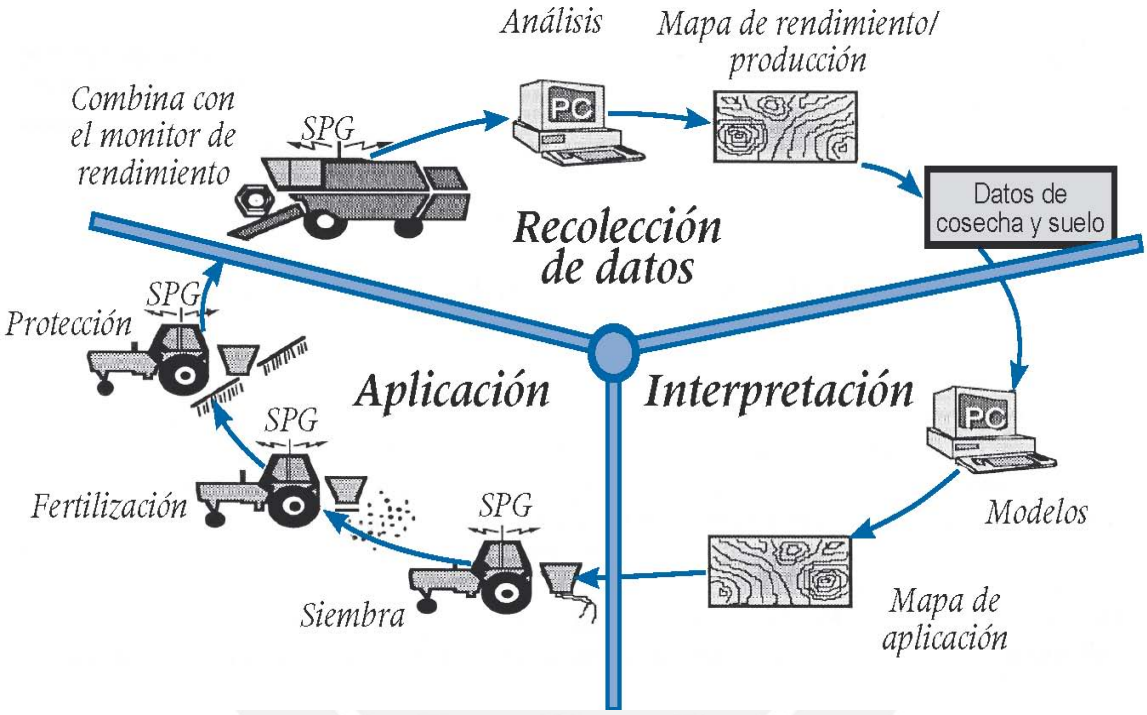

Figura 1.1 Ciclo completo de la agricultura de precisión [7].

1.4. Declaración de la problemática.

Ante la falta de tecnicidad en los procesos de la agricultura en el sector de campesinado y pequeños productores agrónomos, el CIP y la PUCP unen esfuerzos para el desarrollo de tecnologías para la mejora de la producción, así han enfocado sus esfuerzos en el proceso de registro de imágenes aéreas, con dicho proceso podremos obtener información relevante respecto al estado del campo de cultivo y su evolución a lo largo del tiempo, información que sería muy complicado recolectar y además que implicaría una gran inversión de tiempo y personal.

Sin embargo, el proceso de registro de imágenes aéreas implica algunos retos como son la precisión y sincronización en la captura de imágenes, además de pericia en el control del vehículo aéreo, se explicarán estos puntos más adelante en el capítulo.

Además de ello es de suma importancia la implementación de algoritmos eficientes, que permita la obtención de un panorama fiel del área que se requiere analizar, de esta forma obtener calidad de resultados, por ello es necesario un análisis profundo de distintos métodos (comparativas) que nos permitan obtener calidad al aplicarlas sobre imágenes aéreas de campos agrícolas, teniendo en cuenta también la carga computacional que ello requiere, generar métodos que requieran de pocos recursos es de vital importancia ya que tener un computador de altas prestaciones no es una constante en muchos poblados pobres del país.

CAPÍTULO 2

TECNOLOGÍAS ACTUALES PARA EL REGISTRO DE IMÁGENES

2.1. Generalidades.

El presente estudio tiene como fin mostrar los diferentes avances tecnológicos y algunos conceptos básicos para comprender los requerimientos de estudio para el registro de imágenes aéreas, por ejemplo se presenta el uso de los vehículos aéreos no tripulados, el proceso de registro de imágenes, así como algunos alcances de programación y los motivos por los cuales se están usando en el presente trabajo.

Sabemos que la agricultura ya no es la misma de hace algunas décadas, sin embargo tenemos productos agrícolas muy importantes como son la papa que representa un 25% del PBI agropecuario [9], el centro internacional de la papa (CIP) junto con la PUCP impulsan el desarrollo de tecnologías que beneficien a su sana producción y de esa manera aumentar la sostenibilidad ambiental y ayudar a garantizar la seguridad alimentaria en las zonas más pobres y marginadas.

2.2. Vehículos aéreos no tripulados.

Los vehículos aéreos no tripulados (UAVs por sus siglas en inglés), fueron introducidos con fines militares durante la primera guerra mundial (1917) y a través de otros conflictos bélicos, si embargo estos primeros modelos eran imprecisos y poco confiables [10], en la actualidad se le dan muchos más usos como pueden ser seguridad ciudadana, topografía, arqueología y agricultura.

Figura 2.1 Aeronave no tripulada usada en investigaciones PUCP.

Estos vehículos aéreos luego recibieron el nombre de “drones” por el cual son más conocidos coloquialmente, entre ellos podemos encontrar diversos tipos como puede ser un aeromodelo comercial, es una versión pequeña de una avioneta, o también podemos encontrar aeronaves con un determinado número de hélices, para nuestro presente trabajo utilizaremos uno de seis hélices (Figura 2.1).

Actualmente su uso en la agricultura se ha acrecentado debido a la reducción en su coste, el principal motivo por el cual se comenzó a usar las aeronaves no tripuladas, fue debido a las limitaciones que se tenían con la toma de imágenes mediante satélites, ya que los tiempos para la obtención de fotografías podía extenderse hasta en meses, además del alto coste que suponía realizar estas imágenes de manera periódica.

En estas plataformas se montaron cámaras de fotos y filmadoras de alta definición, que sirvieron para la toma de imágenes aéreas. Luego se realizaron otros avances como es la colocación de una unidad GPS que nos permitió realizar vuelos automatizados (sin presencia de un piloto en tierra), Estos avances en el área fueron muy bien usados en la agricultura de precisión, esta tecnología de toma de imágenes nos permite obtener conocimiento entre la variabilidad de los sectores de cultivo, ataque de plagas o detectar fallas de siembra o fertilización y así tomar las acciones que el agricultor considere conveniente.

Al usar estas plataformas no tripuladas debemos considerar que el aeromodelo puede realizar diferentes tipos de movimiento, ya sean controlados o debido a las condiciones del entorno en las que se realiza el vuelo, estas son el pitch, roll y yaw.

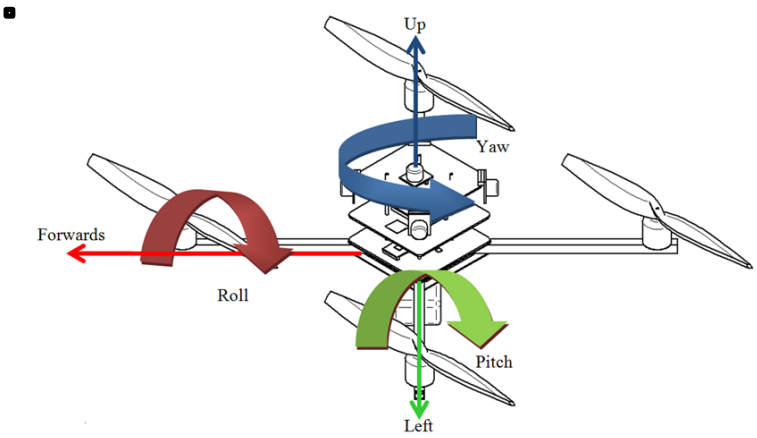

Figura 2.2 Movimientos del UAV: Pitch, Roll y Yaw.

2.3. Registro de Imágenes.

El registro de imágenes es el procedimiento por el cual se intenta encontrar correspondencia entre un par de imágenes y alinearlas espacialmente de tal forma que pueda reducirse el error de correspondencia [19]. Este proceso es muy usado en diferentes aplicaciones de visión por computadora como son localización y reconocimiento de objetos, análisis de movimiento y fusión de imágenes.

En la actualidad se tienen grandes requerimientos de calidad de imágenes, sus usos se han masificado ya sea en la cartografía digital o en el campo de las imágenes médicas, por tanto se requiere que la tecnología avance de tal manera que se pueda garantizar robustez y velocidad en este proceso, sobretodo porque con el crecimiento acelerado de la tecnología, casi todo el mundo tiene acceso a teléfonos inteligentes que tienen una mediana capacidad computacional y este es un nuevo mercado para la tecnología de registro de imágenes, de ello la importancia del presente trabajo.

2.3.1. Toma de datos e implicancias.

En este punto analizaremos los cuidados que se deben tener para obtener las fotografías (en nuestro caso imágenes en la frecuencia del rojo e infrarrojo) para minimizar en lo posible los errores al producir el mosaico, en estos estudios es común obtener errores debido a una captura de imagen no óptima.

Para nuestro caso el proceso de vuelo (figura 2.3) de la plataforma debe cumplir los siguientes requerimientos:

- Las toma de imágenes debe ser continua con un intervalo de tiempo entre tomas que dependerá de la velocidad de vuelo.
- Debe existir un traslape entre imágenes en el resultado de la captura, este área en común es vital para obtener las características invariantes entre las dos imágenes.
- El recorrido de la aeronave debe intentar abarcar toda el área de interés, manteniendo en lo posible una estabilidad en el vuelo (ante las perturbaciones del entorno actúa un sistema de control que intenta estabilizar el recorrido).

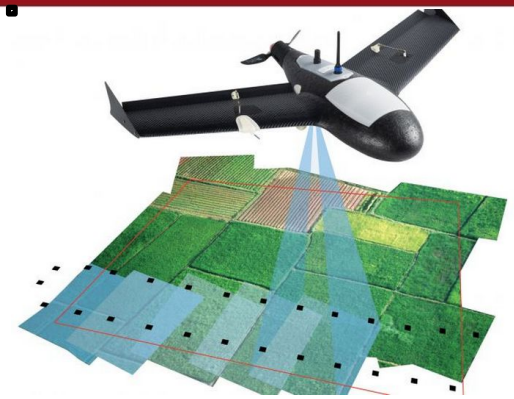

Figura 2.3 Patrón de vuelo para captura de imágenes aéreas [5].

2.3.2. Proceso de registro de imágenes

El proceso de registro de imágenes comprende de un gran número de pasos sin embargo lo podríamos resumir en el siguiente esquema:

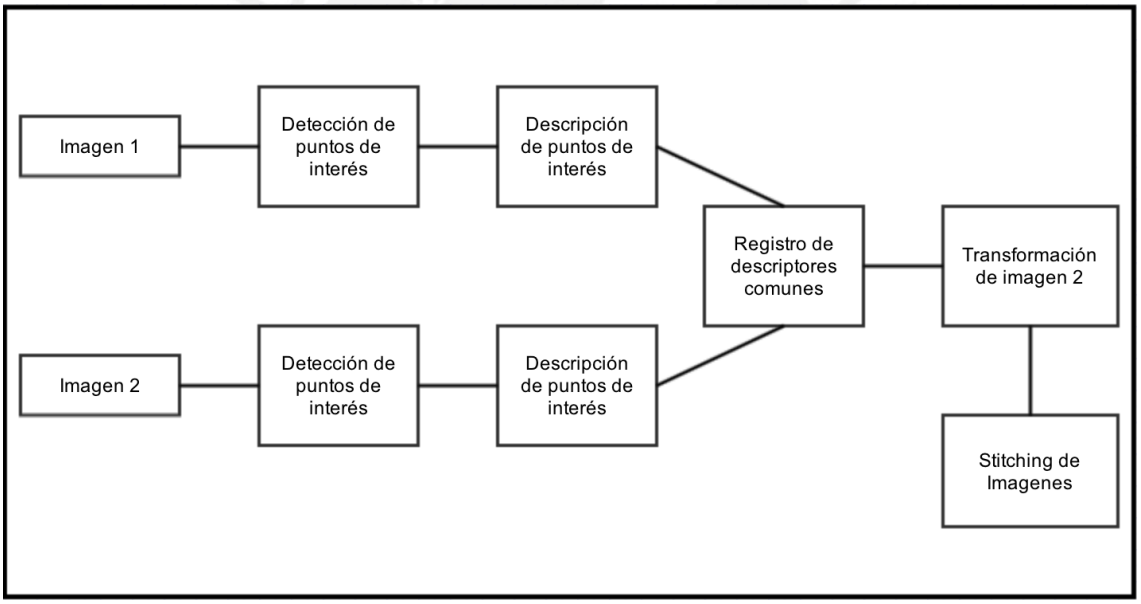

Figura 2.4 Pasos a seguir para registro de imágenes.

Cabe resaltar que las imágenes 1 y 2 poseen un área de redundancia, es decir que poseen un área en común, esta área debe corresponder de al menos un 50% para buenos resultados, esta característica que se debe tomar en cuenta cuando se toma las fotografías con el aeromodelo, de esta manera encontraremos correspondencia entre los puntos de control en ambas imágenes, podemos mostrar un pequeño ejemplo a continuación:

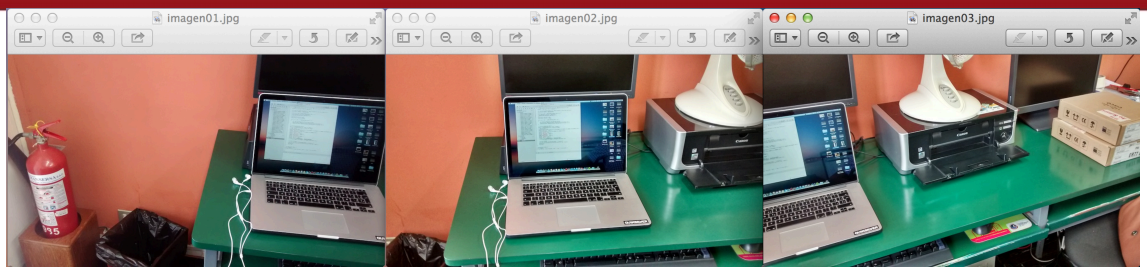

Figura 2.4 Imágenes desde diferentes perspectivas.

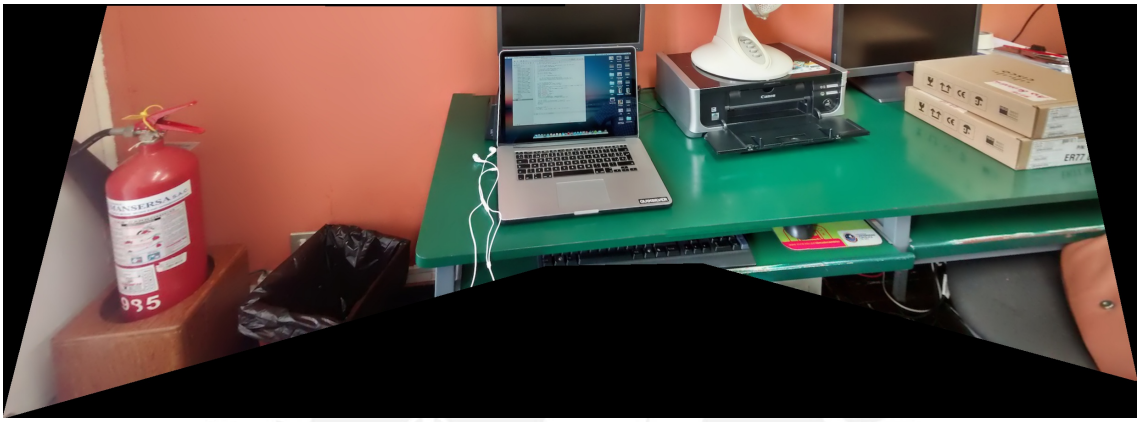

Figura 2.5 Imagen conjunta panorámica.

2.4. Lenguaje C++.

El lenguaje C++ evolucionó de C, este a su vez evoluciona de los lenguajes BCPL y B que fueron desarrollados por Martin Richards y Ken Thompson respectivamente para un uso principalmente orientado a sistemas operativos [4].

En la actualidad la mayoría de sistemas operativos están escritos en C y C++, el lenguaje C está ampliamente extendido sobretodo por la independencia que posee respecto al hardware, sin embargo tiene como contra el exhaustivo cuidado que se debe tener para que los programas diseñados sean portables.

La necesidad de portabilidad de los programas en diferentes plataformas supuso un problema, por el cual se requirió de una evolución de C, por tanto Bjarne Stroustrup desarrolló C++ que brinda más consistencia respecto a C y posee herramientas tales como la herencia y el polimorfismo, además de brindarle la capacidad de ser un lenguaje de programación orientado a objetos, los objetos son componentes reutilizables de software que modelan elementos del mundo real, los POO (programas orientado a objetos) son mas fáciles de entender, corregir y modificar.

Finalmente la decisión de usar el lenguaje C++ en este trabajo, es producto de su uso extendido junto a OpenCV (también es muy usado con Python), esto permitirá simplicidad en la programación además de orden, legibilidad y portabilidad.

2.4. OpenCV.

OpenCV fue desarrollado inicialmente por Intel y es desarrollado bajo una licencia BSD y además es libre para usos académicos y comerciales. Trabaja bajo entornos C, C++, Python y Java, también debemos mencionar que es soportado por múltiples plataformas como son Windows, Linux, Mac OSX, IOS y Android [5].

Esta optimizado para procesamiento de imagen en tiempo real así como aplicaciones de visión por computadora, además debemos mencionar que esta optimizado para procesadores Intel y usa como interfaz principal a C++ (este es un motivo por el cual el programa a sido desarrollado en lenguaje C++).

OpenCV tiene una estructura modular, colocaremos una lista de los principales módulos usados en OpenCV:

- Core: Es el módulo base de OpenCV, contiene estructura de datos básicas como Mat y algunas funciones básicas de procesamiento de imágenes que son muy usadas por otros módulos como “highgui”.
- Highgui: Este módulo posee capacidades básicas de interfaz de usuario como son codecs de imagen y video, manipulación de ventanas, barras móviles y eventos producidos por punteros de mouse, etc.
- Features2d: Este módulo contiene detectores y descriptores de puntos característicos y además de correspondencia.
- Imgproc: Este módulo incluye algoritmos de procesamiento de imágenes, tales como filtrado y transformación de imágenes, etc.
- Objdetect: Este módulo incluye detección de objetos y algoritmos de reconocimiento para objetos estándares.

Cada uno de estos módulos necesarios para una aplicación debe ser asociado en la cabecera de la siguiente manera:

```
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
```

CAPÍTULO 3

DISEÑO DE PROGRAMA EN LENGUAJE C++ PARA EL REGISTRO DE IMÁGENES

3.1. Objetivos de la investigación.

3.1.1. Objetivo general.

Análisis de la eficiencia de algoritmos para el registro de imágenes.

3.1.2. Objetivos específicos.

- Estudio de algoritmos.
- Implementación en lenguaje C++.
- Desarrollo de software para la evaluación.
- Evaluación de los algoritmos creados.

3.2 Requerimientos del sistema.

Para el presente trabajo se debe tomar en cuenta una serie de requerimientos para el correcto funcionamiento del sistema de captura de imágenes aéreas.

- Sistema UAV (para nuestro caso se usa el modelo DJI S800 EVO).
- Una cámara para captura de imagen (Point Grey firefly con lente de 8mm.).
- Software para control de vuelo en caso el vuelo sea automático (Mission Planner).
- Filtros de 800 y 1200 nm. para recoger imágenes rojas e infrarrojas.
- Un piloto en tierra con suficiente pericia para controlar el aeromodelo en caso el vuelo sea en modo manual.

Se debe considerar además algunas características para el vuelo tales como altura y espacio de sensado de imágenes, estos serán determinados por la resolución mínima requerida para poder obtener los detalles en las imágenes (pixeles por centímetro cuadrado), el interesado debe dar dicha información de acuerdo a su requerimiento, por ejemplo podría tomar en cuenta el tamaño de una hoja de viñedo como referencia de visibilidad mínima.

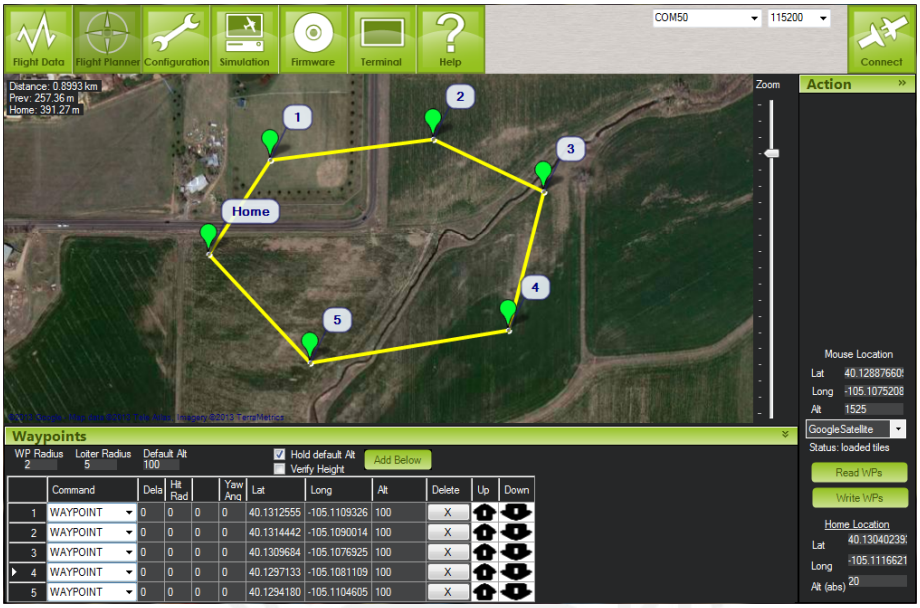

Figura 3.1 Interfaz gráfica del software Mission Planner [8].

Además se debe considerar el traslape que se desea obtener entre imágenes consecutivas, de acuerdo a este aspecto se debe encontrar la velocidad de vuelo y el tiempo entre ráfagas para la captura de imágenes, para configurar estos requerimientos de vuelo requerimos de un software de control de vuelo como el Mission Planner (Figura 3.1).

3.3. Método de registro de imágenes.

3.3.1. Algoritmos de localización y descripción de puntos de interés.

Como se indicó anteriormente se analizarán cuatro algoritmos: SIFT, SURF, BRISK, FREAK; estos algoritmos analizan la localización de puntos con características que los hacen diferentes y notables respecto a su entorno, una vez obtenidos los puntos clave, se requiere de una representación vectorial de estas características notables en las imágenes, es por ello que el siguiente paso es la descripción en forma de vectores para luego evaluarlos de la manera más conveniente. Estos métodos a su vez se valen de técnicas para filtrar estos puntos y hacerlos más robustos ante diferentes cambios tales como: cambios de iluminación, ruido, rotación, escalamiento, etc.

3.3.1.1. SIFT.

A continuación se presenta los pasos principales para obtener las características de una imagen:

1. Detección de extremos en espacio-escala: En esta primera etapa se realiza una búsqueda de los posibles puntos de interés, mediante la representación en espacio-escala de la imagen original, luego de ello se usa la función diferencia de gaussianos para dicho objetivo.

2. Localización de puntos de interés: Cada punto candidato es estudiado y comparado con sus píxeles vecinos, estos puntos serán seleccionados en base a la medida de su estabilidad.

3. Asignación de Orientación: A cada punto de interés se le asigna una o más direcciones en base a las direcciones del gradiente de su vecindad, estas operaciones como la orientación, escala y localización serán usadas para futuras operaciones, todas estas características le proveerán de invariancia ante estas transformaciones.

4. Descriptor de puntos clave: Los gradientes de la imagen local son medidos en una determinada escala en la vecindad del punto de interés, de tal forma que sea identificable y a su vez invariable ante los efectos como son la distorsión y el cambio en la iluminación.

En el siguiente apartado profundizaremos en el estudio de los pasos mencionados anteriormente:

Detección de extremos en espacio escala

En esta etapa debemos detectar los puntos de interés usando múltiples escalas, para ello debemos trabajar con la imagen original filtrada, por la contribución de Koenderink (1984) y Lindeberg (1994) se sabe que el filtro a usar debe se un filtro gaussiano pasa-bajos, de esta forma podemos aminorar el ruido, entonces la imagen $I(x, y)$ se convoluciona con el filtro gaussiano $G(x, y, \sigma)$ de la siguiente manera:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (3.1)$$

donde:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (3.2)$$

Lowe (1999) propuso un método para usar una función de diferencia de gaussianos (DoG) y convolucionarlo con la imagen original, estos filtros están separados en escala por un factor constante de valor k.

$$F(x, y, \sigma) = G(x, y, k\sigma) - G(x, y, \sigma) \quad (3.3)$$

$$D(x, y, \sigma) = F(x, y, \sigma) * I(x, y) \quad (3.4)$$

En este caso trabajaremos con número de tres escalas por octava y un sigma de 1.6 (dado los resultados de la investigación de Lowe). Entonces con tres escalas tomamos los gaussianos y luego para obtener la DoG se resta las imágenes adyacentes, continuamos la misma operación para la siguiente octava, la cual fue obtenida muestreando la imagen por un factor de 2, aquí se obtienen muchas imágenes filtradas con valores extremos donde el tamaño y el lugar de la DoG es similar a la estructura dentro de la imagen.

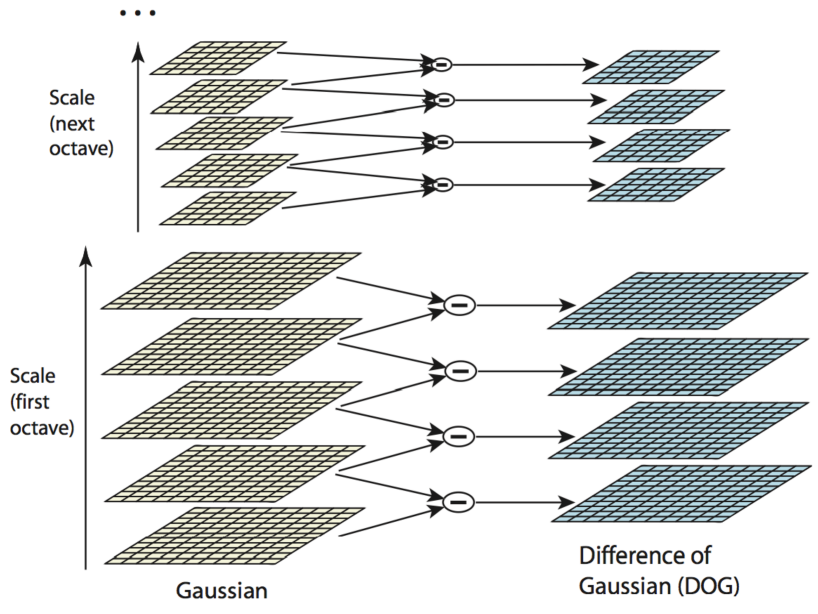

Figura 3.2 Esquema espacio-escala de diferencia de gaussianos [12].

Luego para la búsqueda de extremos cada pixel debe ser comparado con sus veintiséis vecinos (nueve en la escala superior, nueve en la escala inferior y ocho en su propia escala local), en la siguiente figura se muestra de manera gráfica lo anterior.

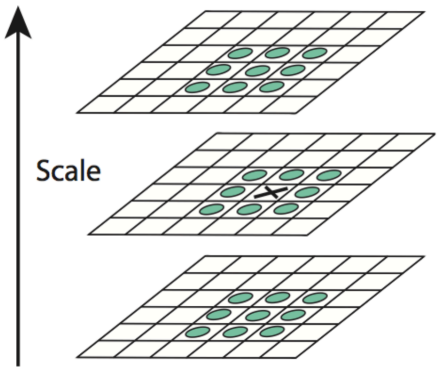

Figura 3.3 Comparativa en escala local de un punto con sus 28 vecinos [12].

Localización de puntos clave.

Entre los puntos que pasaron la búsqueda de máximos debemos de eliminar los puntos que poseen poco contraste ya que no son estables ante alteraciones como son el ruido producido por un cambio de iluminación, para ello debemos estimar una función D con una serie de Taylor:

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x \qquad (3.5)$$

Luego derivamos esta función e igualándola a cero, dado:

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2} * \frac{\partial D}{\partial x} \qquad (3.6)$$

$$\Rightarrow D(\hat{x}) = D + \frac{1}{2} * \frac{\partial D^T}{\partial x} \hat{x} \qquad (3.7)$$

Si el valor obtenido en la ecuación 3.7 es menor a 0.03 (en un rango de valores entre 0 y 1) el valor es eliminado [12], además de encontrar los puntos con poco contraste

debemos eliminar puntos que provengan de líneas rectas, la curvatura de D (representada por una matriz Hessiana de 2x2) en estos puntos corresponderá a un valor α grande y un β pequeño:

$$Tr(H) = \frac{\partial^2 D}{\partial x^2} + \frac{\partial^2 D}{\partial y^2} = \alpha + \beta \quad (3.8)$$

$$Det(H) = \frac{\partial^2 D}{\partial x^2} * \frac{\partial^2 D}{\partial y^2} = \alpha * \beta, \alpha = r\beta \quad (3.9)$$

Para un valor de r definido se analiza la inecuación, Lowe propone un valor r = 10:

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r} \quad (3.10)$$

Asignación de orientación.

Aquí debemos hallar la gradiente y orientación de los puntos característicos hallados, esta operación es realizada para formar los histogramas de orientación que serán usados para crear los vectores descriptores, la magnitud y la orientación son calculadas de la manera siguiente:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (3.11)$$

$$\theta(x,y) = \tan^{-1}(L(x,y+1) - L(x,y-1)))/(L(x+1,y) - L(x-1,y))) \quad (3.12)$$

Descriptor de puntos clave.

Las operaciones previas fueron usadas para asignar localización, escala y orientación de cada punto de interés, y por tanto se logró invariancia ante estos parámetros, ahora mediante un vector descriptor se busca invariancia ante otros tipos de variaciones como son cambios de iluminación y puntos de vista 3D.

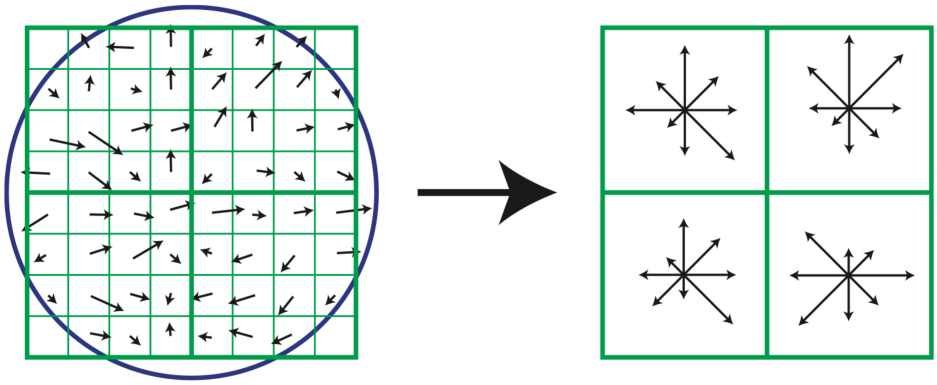

Figura 3.4 Histograma de orientación y magnitud de una ventana gaussiana [12].

Se realiza un tratamiento inspirado en la visión biológica, de esta forma se usa una ventana gaussiana circular y se usa los valores m y θ (figura 3.4), donde tendrán una mayor consideración los valores de puntos cercanos al centro, podemos observar que el histograma posee 8 valores distintos, donde dichos histogramas poseen valores iguales a pesar que la ventana gaussiana pueda moverse hasta 4 pixeles, para el ejemplo de la figura 3.4 podemos ver un ejemplo de 2x2 histogramas, sin embargo en la realidad esta será de 4x4 que multiplicado por los 8 valores distintos de orientación obtendremos el vector descriptor de SIFT de 128 dimensiones por cada punto característico.

3.3.1.2. SURF.

Detección de puntos de interés.

En este algoritmo se usa una aproximación básica de la matriz Hessiana debido a su gran precisión, es usado para hallar puntos donde la determinante es máxima. Dado un punto $\mathbf{x} = (x, y)$ de una imagen $\mathbf{I}$, la matriz hessiana esta determinada por:

$$H(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \quad (3.13)$$

Donde $L_{xx}(\mathbf{x}, \sigma)$ es la convolución de segundo orden de la Gaussiana con la imagen $\mathbf{I}$ en el punto $\mathbf{x}$, se maneja de manera similar para $L_{xy}(\mathbf{x}, \sigma)$ y $L_{yy}(\mathbf{x}, \sigma)$, debido al éxito obtenido por Lowe [12], se usará una aproximación de la matriz Hessiana con los filtros de caja (L_{xx}, L_{xy}, L_{yy}) con un valor $\sigma = 1.2$ los cuales también serán discretizados y serán representados por D_{xx}, D_{xy}, D_{yy} (derivadas parciales) evaluándolos con un costo computacional muy bajo gracias al uso de imágenes integrales, se muestran en la figura 3.5 lo descrito.

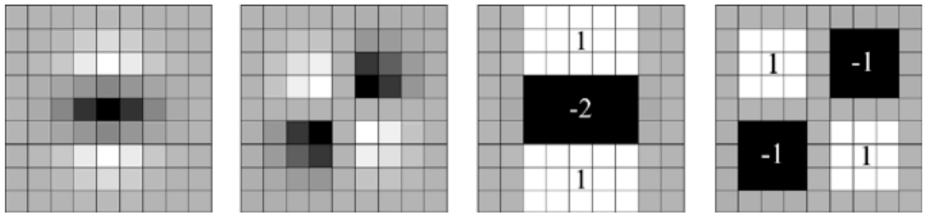

Figura 3.5 Derecha a izquierda: Derivada parcial de segundo orden L_{yy} y L_{xy}, aproximaciones de las anteriores D_{xx}, D_{xy} [13].

La determinante de las aproximaciones de las derivadas parciales D_{xx}, D_{xy}, D_{yy} se calcula en la ecuación 3.14, donde el valor de 0.9 se debe a la aproximación del filtro Gaussiano:

$$\det(H_{aprox}) = D_{xx}\, D_{yy} - (0.9 D_{xy})^2 \quad (3.14)$$

En el detector SURF la representación espacio-escala es similar a SIFT, es decir que esta dividido en octavas, sin embargo hay un cambio de paradigma ya que en SIFT se

realizaba un re-muestreo de la imagen original, en el caso SURF el filtro crecerá a medida que la escala se incrementa, como se muestra a continuación:

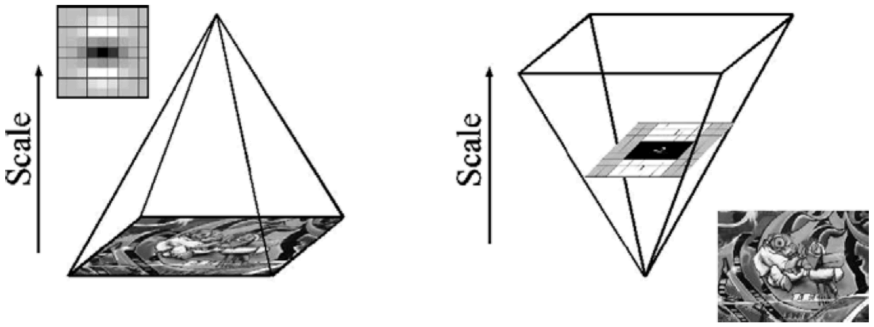

Figura 3.6 Izquierda: Enfoque clásico usado en SIFT, derecha: enfoque usado en SURF [13].

En el algoritmo de detección de SURF las octavas están compuestas por un número fijo de imágenes que son resultado de la convolución de la imagen original con una serie de filtros que incrementan su tamaño (figura 3.7).

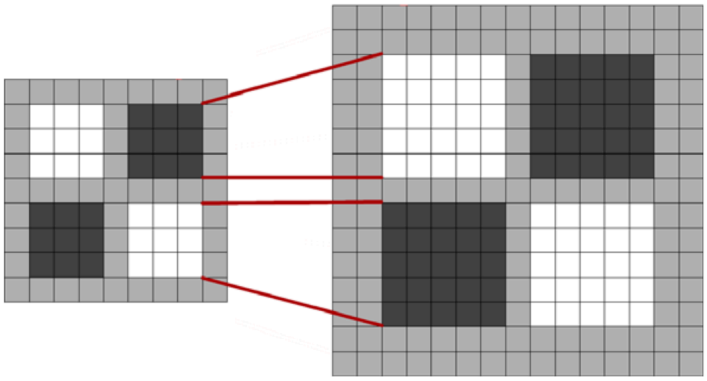

Figura 3.7 Incrementos de tamaño sucesivos en los filtros [13].

La imagen de salida obtenida tras la convolución de la imagen original con un filtro de dimensiones 9x9, que corresponde a la derivada parcial de segundo orden de una gaussiana con $\sigma = 1.2$ [13], es considerada como la escala inicial, las capas siguientes serán incrementos graduales de los filtros (figura 3.8). Finalmente para calcular la localización de todos los puntos de interés en todas las escalas, se procede mediante la eliminación de los puntos que no cumplan la condición de máximo en un vecindario de 3x3x3. De esta manera, el máximo determinante de la matriz Hessiana es interpolado en la escala y posición de la imagen.

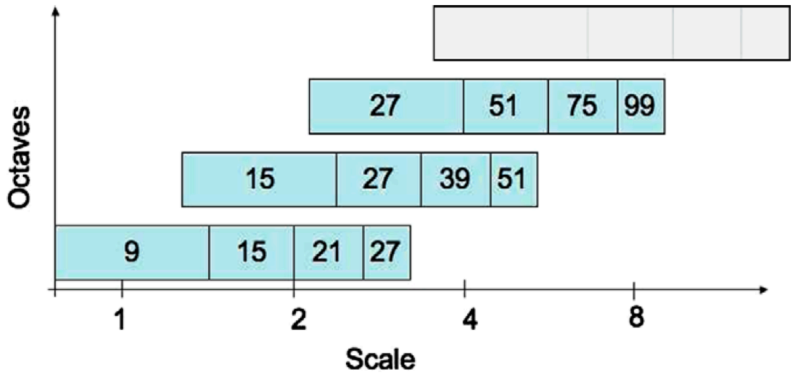

Figura 3.8 Representación numérica del tamaño de los filtros a través de diferentes octavas [13].

Extracción de descriptores.

Ahora debemos obtener la orientación de cada punto clave para que nuestro descriptor sea invariante ante la rotación, por ello debemos hallar la respuesta wavelet Haar en las direcciones x e y en un radio de 6σ para cada punto de la imagen I. El tamaño del wavelet es de 4σ, los filtros usados se muestran en figura 3.9, donde podemos usar nuevamente imágenes integrales para filtrar rápidamente.

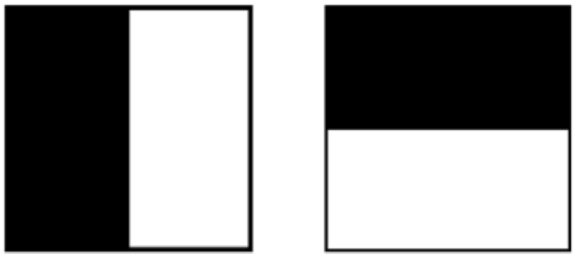

Figura 3.9 Filtros wavelet Haar en ejes X e Y [13].

Una vez halladas las respuestas wavelet con un valor de gaussiano $\sigma = 2s$ (donde s es la escala en que el punto fue detectado) centrado en el punto de interés, estas respuestas son representadas como puntos en el espacio, la respuesta horizontal a lo largo del eje de abscisas y el vertical a lo largo de la ordenada, luego se obtiene la orientación dominante mediante la suma de todas las respuestas de cada sector dentro de una ventana móvil de valor $\pi/3$, el mayor vector obtenido en todas las ventanas será la orientación del punto de interés.

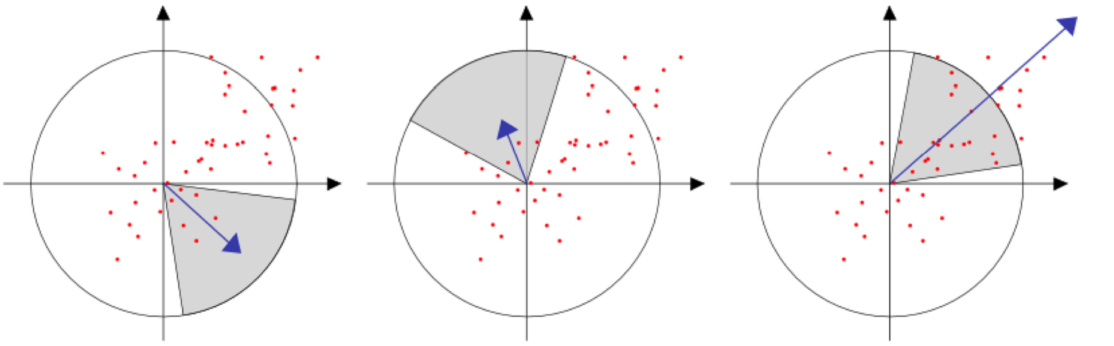

Figura 3.10 Asignación de orientación en una ventana de tamaño $\pi/3$ [19].

Para la extracción del descriptor, primero debemos construir una región cuadrada de tamaño $20s$ que estará centrada en el punto de interés y orientada con el valor obtenido en el paso anterior, dentro de esta ventana se formará 4x4 sub-regiones, para cada subregión se calculará la respuesta wavelet Haar de 25 puntos de muestra regularmente distribuidos, sumando las respuestas paralelas a los ejes x e y se obtiene lo siguiente:

$$v_{subregión} = \left[\sum dx \quad \sum dy \quad \sum |dx| \quad \sum |dy|\right] \qquad (3.15)$$

De esta forma por cada una de las 16 sub-regiones tenemos 4 valores, se esta forma tenemos un vector descriptor de longitud 64.

3.3.1.3. BRISK.

A continuación se detalla el algoritmo BRISK, cabe resaltar la modularidad de este método, es decir el detector BRISK puede ser usado en combinación con cualquier descriptor de puntos clave y viceversa.

Detección espacio-escala de puntos de interés.

Este método está inspirado en el algoritmo AGAST, el cual es una extensión del algoritmo FAST, que es eficiente para extracción de características, con el fin de obtener invariancia a cambios de escala, se realizará una búsqueda de máximos no solo en la imagen sino también en espacio-escala usando el valor s como medida de prominencia, además de estimar cada punto de control en un espacio-escala continuo.

Primero se crea una pirámide de capas espacio-escala que consiste de n octavas c_i y n intra-octavas d_i, con n = 4 como valor típico, las octavas son formadas muestreando entre 2 la imagen original c_0, cada intra-octavas d_i estará ubicada entre las capas c_i y c_{i+1} de tal forma que se cumple que $t(c_i) = 2^i$ y $t(d_i) = 2^i * 1.5$.

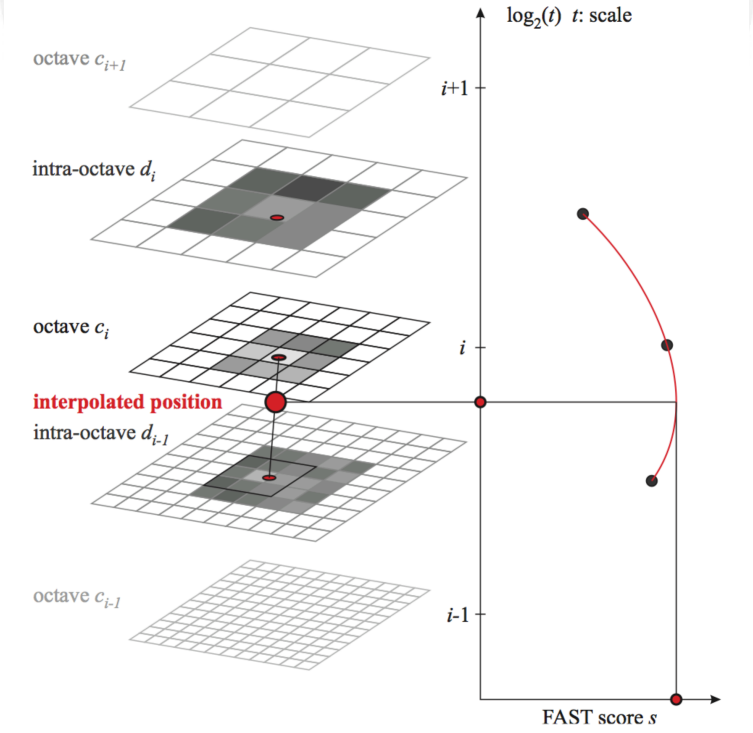

Figura 3.11 Detección de punto de interés en espacio-escala [14].

En BRISK se hace uso de una máscara 9-16 (detector FAST 9-16), lo cual significa que al menos 9 pixeles consecutivos de los formados por un radio de 16 pixeles son lo suficientemente más brillantes o más oscuros que el pixel central (Figura 3.12), tal que esa distancia debe superar un umbral T y así identificar potenciales zonas de interés. Luego se realiza una eliminación de los puntos no máximos, para ello primero se debe cumplir que el punto a evaluar debe tener el máximo valor s (s es el máximo umbral considerando el punto como una esquina) respecto a sus 8 vecinos, luego dicho valor también debe ser máximo respecto a la capa superior e inferior.

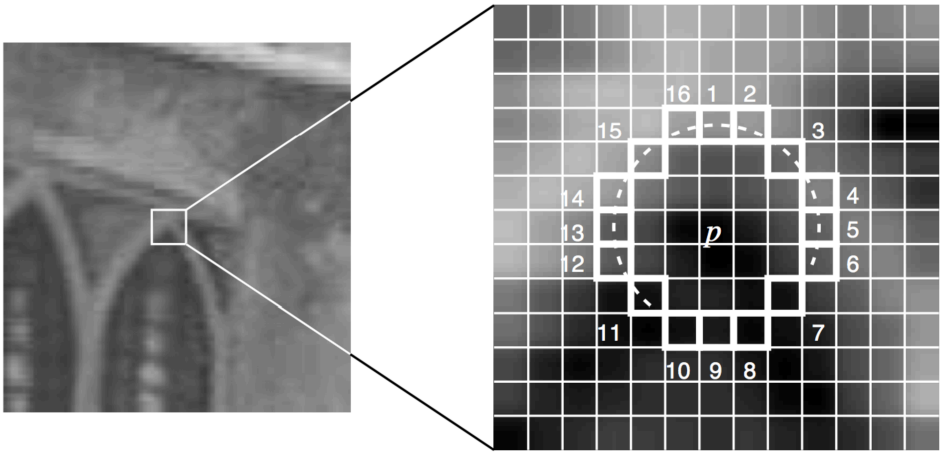

Figura 3.12 Test de detección de esquinas en una porción de imagen [17].

Se obtiene luego de 3 en 3 el valor de prominencia s (en la capa del punto de interés, la capa superior e inferior), con estos puntos obtenidos se ajusta una función 2D cuadrática, la proyección de la función obtenida la ajustamos a una parábola, donde estimemos el máximo "score" s de la función, realizaremos una interpolación para obtener la escala en la cual se obtiene el valor máximo s, este valor que denominaremos t será la escala verdadera del máximo, es decir obtenemos el valor en una escala continua, este valor t será usado para calcular el patrón de muestreo en la sección de descripción del punto de interés.

Descripción de puntos de interés.

El descriptor BRISK hace uso de una cadena binaria de valores, de esta manera podremos hacer uso de un menor tiempo computacional, además de requerir de menos recursos para la evaluación del mismo, para ello utilizaremos un patrón de muestreo (Figura 3.13) que nos permitirá con simples comparaciones de brillo e

identificando la dirección de cada punto clave obtener invariancia ante la rotación y en general robustez.

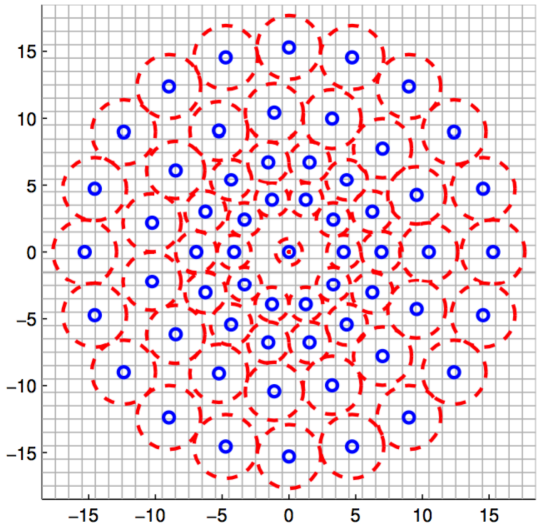

Figura 3.13 Patrón de muestreo usado en BRISK [14].

Inspirado en el descriptor DAISY [16], aquí se aplica un kernel gaussiano con una desviación estándar σ_i (proporcional a la distancia entre puntos en el círculo) a cada punto $\mathbf{p}_i$, dado un número N de puntos en el patrón de muestreo tendremos $N.(N-1)/2$ pares de puntos. Con los valores de intensidad de dos puntos $I(\mathbf{p}_i, \sigma_i)$ e $I(\mathbf{p}_j, \sigma_j)$ estimaremos el gradiente local de la siguiente manera:

$$\mathbf{g}(\mathbf{p}_i, \mathbf{p}_j) = (\mathbf{p}_i - \mathbf{p}_j).\frac{I(\mathbf{p}_j, \sigma_j) - I(\mathbf{p}_i, \sigma_i)}{\|\mathbf{p}_i - \mathbf{p}_j\|^2} \qquad (3.16)$$

Luego definiremos un par de subconjuntos, pares distancias cortas S y pares distancias largas L, donde definiremos estos subconjuntos mediante umbrales, los puntos pertenecen a distancias cortas si las distancias entre los puntos no superan el umbral $\delta_{max} = 9.75t$ y las distancia largas si son mayores al umbral $\delta_{min} = 13.67t$ [14], el valor de la gradiente (ecuación 3.17) se usará para obtener un ángulo con el cual se rotará al patrón de muestreo.

$$\mathbf{g} = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \frac{1}{L}. \sum_{(\mathbf{p}_i, \mathbf{p}_j) \in L} \mathbf{g}(\mathbf{p}_i, \mathbf{p}_j). \qquad (3.17)$$

Con el valor hallado anteriormente obtenemos el ángulo $\alpha = \arctan 2(g_y, g_x)$, con él rotaremos el patrón de muestreo alrededor del punto de interés, luego con el subconjunto de distancias cortas y mediante una comparación de las intensidades de los puntos en el patrón se forma el vector descriptor b.

$$b = \begin{cases} 1, & I(\mathbf{p}_j^\alpha, \sigma_j) > I(\mathbf{p}_i^\alpha, \sigma_i) \\ 0, & \text{en otros casos} \end{cases} \quad \forall(\mathbf{p}_i^\alpha, \mathbf{p}_j^\alpha) \in S \qquad (3.18)$$

Finalmente después de pasar por el patrón de muestreo, aplicar los umbrales y realizar las comparaciones de intensidad entre los puntos del patrón obtenemos una cadena de 512 bits, al realizar una aplicación como la comparación entre dos imágenes estos vectores formados serán fácilmente evaluados, ya que requerirá de una simple operación XOR bit a bit, de esta forma bajaremos los tiempos computacionales además de reducir espacio en memoria.

3.3.1.4. FREAK.

El algoritmo FREAK tiene la particularidad de no tener una etapa de detección, además tiene la propiedad de modularidad, es decir que se requiere de alguna etapa de detección, para el presente trabajo haremos uso de la detección del algoritmo BRISK.

Este método al igual que BRISK utiliza un patrón de muestreo, dicho patrón esta inspirado en la distribución de las células ganglionares presentes en la retina humana, podemos notar en la figura 3.14 que al igual que en la retina se observa una mayor concentración densidad de células ganglionares en el centro, además podemos notar cuatro áreas marcadas: (a) foveola, (b) fóvea, (c) parafóvea y (d) perifóvea.

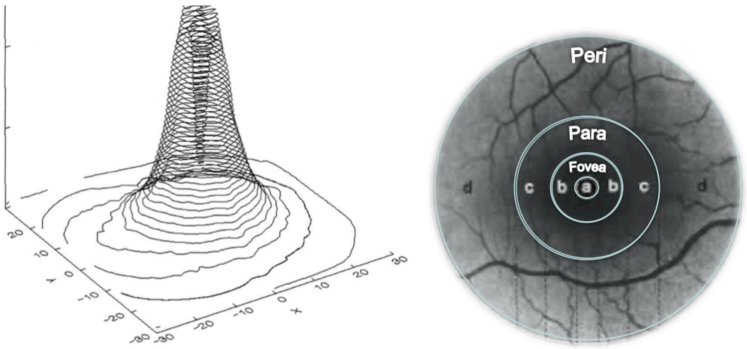

Figura 3.14 Izquierda: Distribución de células ganglionares en la retina. Derecha: Áreas en la retina humana [15].

El patrón de muestreo usado en FREAK (figura 3.15) presenta las áreas mostradas en la figura anterior, se puede notar también la redundancia entre las áreas que permite que más información sea capturada, cada campo receptivo en el patrón cumple además un crecimiento logarítmico para la desviación estándar de los kernels gaussianos similar al comportamiento de la retina humana, los puntos mencionados ayudan a un uso menor de campos receptivos.

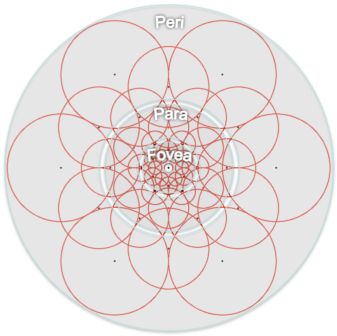

Figura 3.15 Patrón de muestreo FREAK inspirado en la retina humana [15].

Para construir el descriptor F (ecuación 3.19) debemos hacer uso de comparaciones par a par de pixeles y sus campos receptivos con su respectivo kernel gaussiano, su construcción se detalla de la siguiente forma:

$$F = \sum_{0 \le a < N} 2^a T(P_a) \qquad (3.19)$$

$$T(P_a) = \begin{cases} 1, & \text{if } (I(P_a^{r_1}) - I(P_a^{r_2})) > 0 \\ 0, & \text{en otros casos} \end{cases} \qquad (3.20)$$

Donde P_a representa a una par de campos receptivos, N es el tamaño del descriptor y $I(P_a^{r_1})$ es la intensidad de un campo receptivo con filtro gaussiano.

Dado que con solo unos cuantos campos receptivos podemos obtener miles de comparaciones de pares y con ello tener un descriptor de gran tamaño, se utiliza el siguiente método que es similar a ORB que nos permita escoger de manera selectiva los pares.

Primero debemos crear una matriz D, cada columna representa a un punto clave y su vector descriptor con todos sus posibles comparaciones de pares en el patrón de muestreo, usamos 43 campos receptivos, los cuales proporcionan más de mil pares.

Luego hallamos la media de cada columna, y ordenamos las columnas de mayor a menor varianza, donde una media de 0.5 indicará la máxima varianza.

Del método anterior experimentalmente se observó que con solo los primeros 512 pares daban información relevante, agregar más información no incrementaban el rendimiento del algoritmo, además se forma una estructura con una cierta preferencia de orden descendente, notamos que la vista humana necesita primero tener una orientación general de su entorno y así localizar su objetivo, esto quiere decir que la retina hace uso primero de los campos receptivos perifoveales, al realizar el ordenamiento de la matriz D las primeras comparaciones de pares de cada fila corresponden a los campos receptivos perifoveales y luego los demás campos.

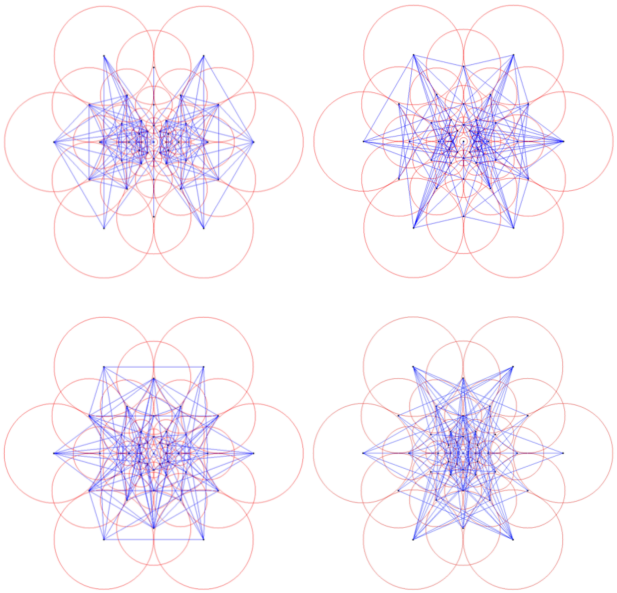

Figura 3.16 Grupos de 128 pares de comparaciones, Arriba-Izquierda: Las comparaciones se dan mayormente entre puntos de la zona perifoveal [15].

Al dividir estos 512 pares en cuatro grupos de 128 pares cada uno, el primer grupo posee los campos receptivos perifoveales, a esta característica se le conoce en inglés como un análisis “coarse-to-fine”, en la figura 3.16 se muestra esta característica.

Como mencionamos anteriormente el algoritmo emula el comportamiento de la retina humana, de esta forma el ojo realiza movimientos independientes llamados sacádicos mientras analiza el entorno de su objetivo con los campos perifoveales, de esta forma el algoritmo realiza una búsqueda en los primeros 16 bytes, esto porque se comprobó experimentalmente que el 90% de los candidatos son descartados al analizar los primeros 16 bytes (cuando se intenta realizar búsquedas de objetos en imágenes) y además los procesadores Intel mediante operaciones paralelas (SIMD) ofrecen comparaciones de 1 byte o 16 bytes en una sola instrucción [15].

Finalmente para obtener invariancia ante la rotación debemos girar el patrón de muestreo tal como BRISK, aquí usamos comparaciones de campos receptivos simétricos con respecto al centro, aquí la diferencia con BRISK.

$$O = \frac{1}{M} \sum_{P_o \in G} \left(I\left(P_o^{r_1}\right) - I\left(P_o^{r_2}\right) \right) \frac{P_o^{r_1} - P_o^{r_2}}{\left\| P_o^{r_1} - P_o^{r_2} \right\|} \quad (3.21)$$

Donde M es el número de pares usados para el cálculo de la orientación, $P_o^{r_i}$ es un vector 2D de coordenadas espaciales del centro del campo receptivo, para la orientación del descriptor usamos 45 pares, de esta manera se hace uso de menos cantidad de memoria.

3.3.2. Correspondencia entre descriptores.

3.3.2.1. Nearest neighbor.

Se define al algoritmo “Nearest neighbor” como la menor distancia euclidiana que existe entre dos puntos para hallar de esta manera su correspondencia.

$$d(\mathbf{u},\mathbf{v}) = (\sum_{i}(u_i - v_i)^2)^{1/2} \quad (3.22)$$

Uno de los métodos para purificar posible errores es el NNDR (nearest neighbor distance ratio), donde la relación entre las distancias de un punto de control al vecino más cercano y del punto de control al segundo vecino más cercano debe ser menor igual que dicho umbral NNDR.

$$\frac{d_1}{d_2} \leq NNDR \quad (3.23)$$

Estas función se encuentra dentro de la librería FLANN de OpenCV, y será usada para los algoritmos que poseen vectores descriptores multidimensionales, tales como SIFT y SURF.

3.3.2.2. Hamming distance.

Dadas dos imágenes x e y, donde se denota el descriptores binarios $b(x)\ y\ b(y) \in \{0,1\}^n$ entonces el valor de Hamming distance se calcula [18]:

$$\mathrm{Ham}(x,y) = \sum_{i=1}^{n}(b_i(x) \otimes b_i(y)) \quad (3.24)$$

Donde n es la dimensión del descriptor binario y ⊗ representa a la operación binaria XOR.

Esta función se encuentra dentro de la librería BFMatcher y será usado para los algoritmos que poseen un descriptor de tipo binario, es decir BRISK y FREAK.

3.3.3. Función de transformación.

Para ello debemos definir el concepto de homografía la cual es una transformación proyectiva que determina la correspondencia entre dos figuras geométricas planas, de esta manera una de las imágenes es proyectada en el sistema de referencia del otro, para obtener una buena transformación es necesario contar con puntos correctamente relacionados.

Entre las transformaciones homográficas podemos encontrar: la traslación, la simetría y la afinidad.

RANSAC.

El algoritmo RANSAC (Random Sample Consensus) busca utilizar la menor cantidad de puntos posibles y de esta manera estimar el modelo y luego ver cuantos datos se ajustan al modelo estimado, así hallamos la matriz de homografía con cuatro correspondencias.

Podemos decir que RANSAC sigue los siguientes pasos:

- Seleccionar un subconjunto aleatorio de puntos.
- Agregar estos puntos a una lista de inliers.
- Asignar los puntos a una línea estimada.
- Si los puntos del subconjunto no se aproximan a la línea, agregar más a la lista.
- Generar una línea con la lista de inliers.

El algoritmo RANSAC será usada de las librerías de OpenCV mediante la función “findHomography”.

Capítulo 4

Análisis de resultados.

En este capítulo realizaremos un análisis cuantitativo de los algoritmos de registro de imágenes usados para el reconocimiento de puntos clave, estos algoritmos son: SIFT, SURF, BRISK y FREAK, en este caso se analizan imágenes que se encuentran en la frecuencia del infrarrojo cercano y del rojo visible, este tipo de imágenes al juntarse y escalar los colores nos permite apreciar la salud de los sectores de cultivo.

Los algoritmos fueron implementados bajo un entorno de programación C++ (se usó el entorno Xcode versión 5.1.1), también se hizo uso de las librerías OpenCV versión 2.4.5. El programa fue elaborado en un computador Core i7 a 2.4 GHz., 8GB. de memoria RAM y sistema operativo Mac OSX 10.8.5.

4.1 Pruebas cuantitativas en imágenes

En este primer análisis encontramos un par de imágenes de campos de caña de azúcar en el espectro del rojo visible que poseen un 82% de área en común.

Imagen 1.

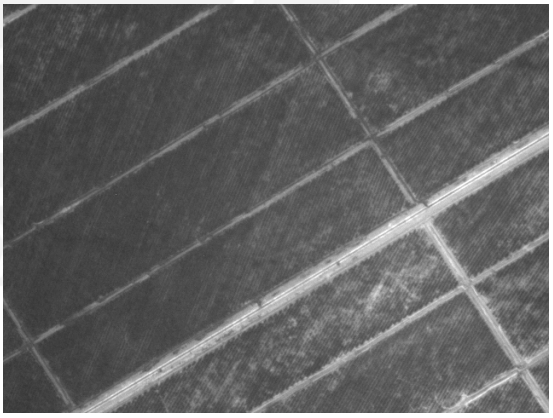

Imagen 2.

	SIFT	SURF	BRISK	FREAK
Número de puntos de control	465	1761	144	120
Tiempo de detección [ms]	30.0047	127.108	7.9297	4.9288
Tiempo de descripción [ms]	47.3145	209.742	1.0198	32.8507
Tiempo total [ms]	77.3192	336.85	8.9495	37.7795
Tiempo por punto [ms]	0.1663	0.1913	0.0621	0.3148

	SIFT	SURF	BRISK	FREAK
Puntos en primera imagen	465	1761	144	120
Puntos en segunda imagen	398	1399	110	91
Tiempo total [ms]	8.7852	34.036	0.9315	0.6699
Tiempo por comparaciíon [ms]	0.0189	0.0193	0.0065	0.0056

A continuación se muestra un ejemplo de la correspondencia entre los puntos de control, donde en azul se muestras las buenas correspondencias y en amarillo se nota los puntos que no encontraron su descriptor correspondiente para el algoritmo BRISK.

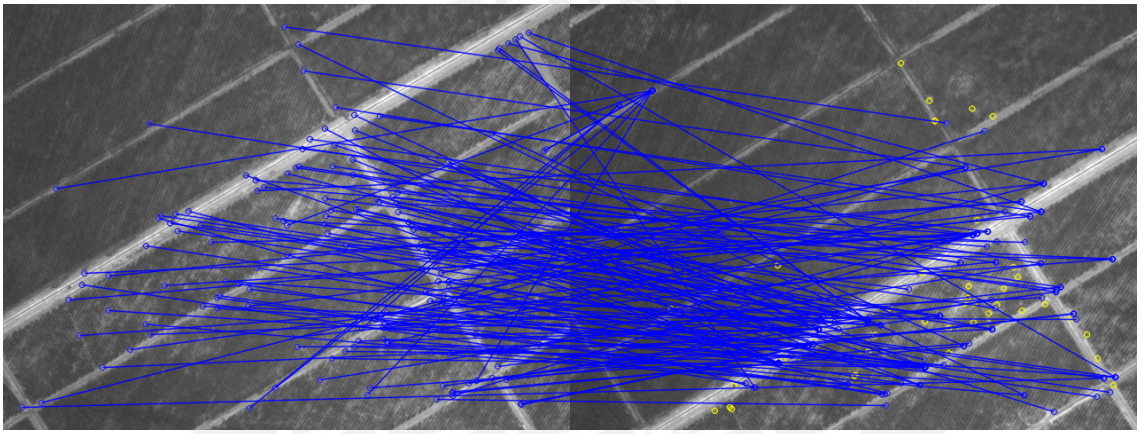

Figura 4.1 Correspondencia de puntos para algoritmo BRISK.

En este segundo análisis tenemos dos imágenes de campos de caña de azúcar en el espectro infrarrojo cercano con un área en común de 80%.

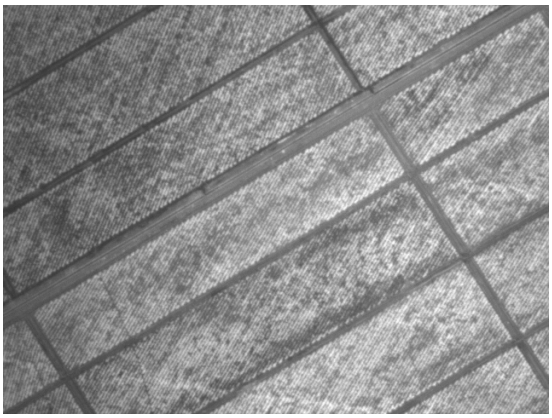

Imagen 1.

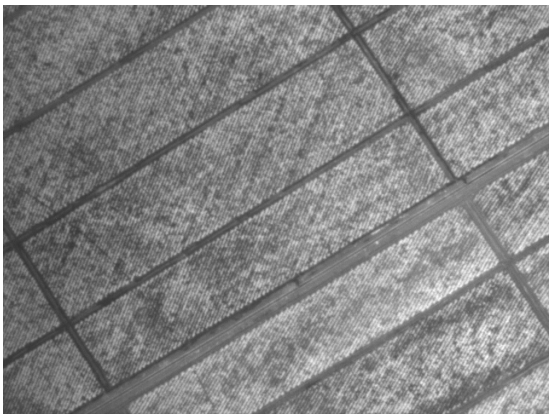

Imagen 2.

	SIFT	SURF	BRISK	FREAK
Número de puntos de control	1293	2812	728	625
Tiempo de detección [ms]	47.1579	193.567	19.2529	14.4415
Tiempo de descripción [ms]	88.9055	336.368	3.7272	34.4782
Tiempo total [ms]	136.0634	529.935	22.9801	48.9197
Tiempo por punto [ms]	0.1052	0.1885	0.0316	0.0783

	SIFT	SURF	BRISK	FREAK
Puntos en primera imagen	1293	2812	728	625
Puntos en segunda imagen	1295	2894	768	634
Tiempo total [ms]	35.0975	65.6805	30.0995	21.2106
Tiempo por comparaciíon [ms]	0.0271	0.0234	0.0413	0.0339

En este ejemplo se evalúan dos imágenes de campos de caña de azúcar que posen un 72% de área en común.

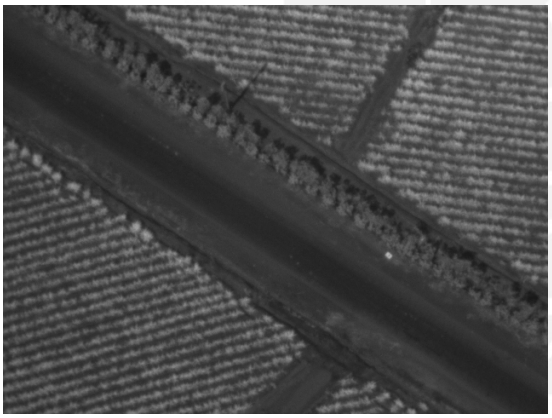

Imagen 1.

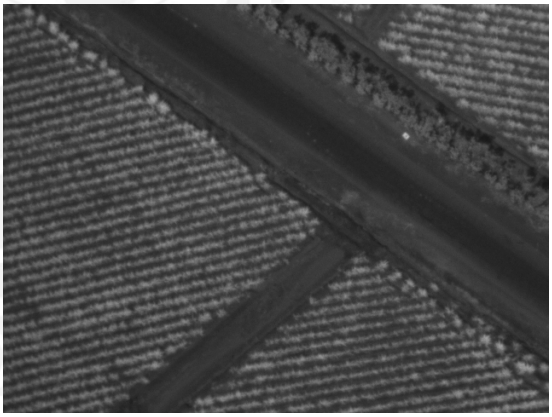

Imagen 2.

	SIFT	SURF	BRISK	FREAK
Número de puntos de control	271	1216	206	159
Tiempo de detección [ms]	39.2857	101.841	9.9033	4.9486
Tiempo de descripción [ms]	32.0076	155.899	1.3156	32.6775
Tiempo total [ms]	71.2933	257.74	11.2189	37.6261
Tiempo por punto [ms]	0.2631	0.2120	0.0545	0.2366

	SIFT	SURF	BRISK	FREAK
Puntos en primera imagen	271	1216	206	159
Puntos en segunda imagen	275	1190	195	149
Tiempo total [ms]	10.2863	26.349	2.8086	1.3823
Tiempo por comparaciíon [ms]	0.0380	0.0217	0.0136	0.0087

En esta cuarta prueba se hace uso de un par de imágenes del jardín central de la PUCP, donde existe un 78% de área en común.

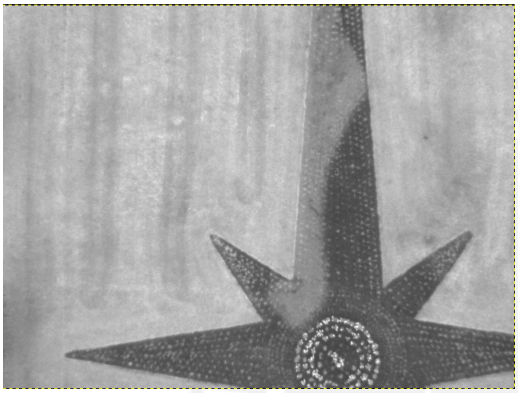

Imagen 1.

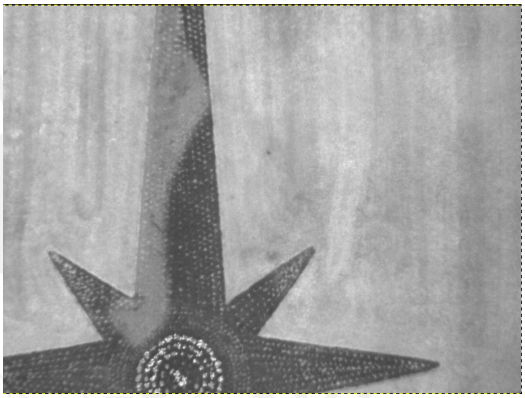

Imagen 2.

	SIFT	SURF	BRISK	FREAK
Número de puntos de control	2008	1991	134	87
Tiempo de detección [ms]	40.8562	145.04	14.7597	5.1207
Tiempo de descripción [ms]	129.025	200.631	1.9569	33.0786
Tiempo total [ms]	169.8812	345.671	16.7166	38.1993
Tiempo por punto	0.0846	0.1736	0.1248	0.4391

	SIFT	SURF	BRISK	FREAK
Puntos en primera imagen	2008	1991	134	87
Puntos en segunda imagen	2379	2173	180	142
Tiempo total [ms]	47.6805	47.2193	2.9435	0.7265
Tiempo por comparaciíon [ms]	0.0237	0.0237	0.0220	0.0084

En este caso se muestra un ejemplo de los puntos de correspondencia para el algoritmo FREAK, donde los puntos azules corresponden a corresponden a correspondencia exitosas y los puntos amarillos representan los puntos descriptores no correspondidos.

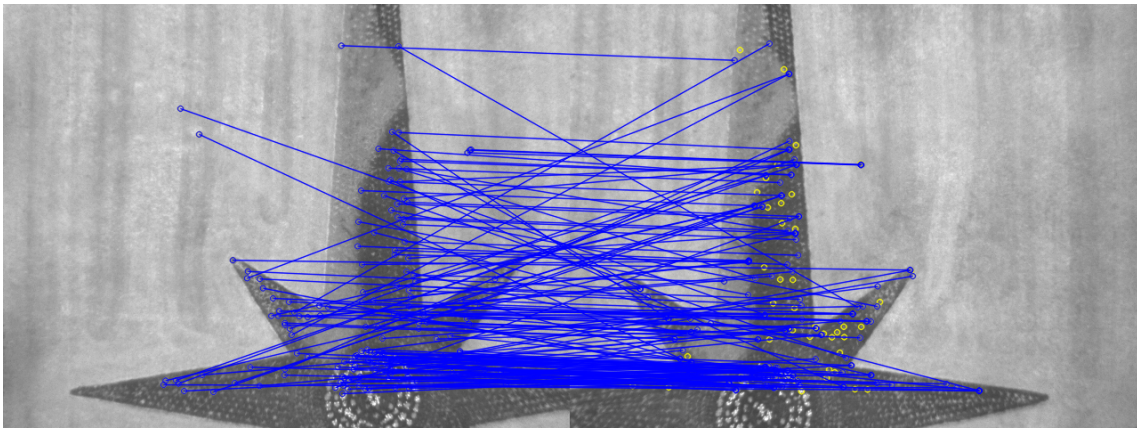

Figura 4.2 Correspondencia de puntos para el algoritmo FREAK.

4.2 Resultados finales de mosaico de imágenes aéreas.

Para estas pruebas se hizo uso de un algoritmo que usa a dos imágenes consecutivas y así encuentra la figura panorámica de las dos imágenes, luego esta nueva imagen formada se une con otra consecutiva así hasta formar el mosaico requerido.

La siguiente imagen fue realizada con un total de 25 imágenes en formato .bmp y corresponde a un campo de caña de azúcar.

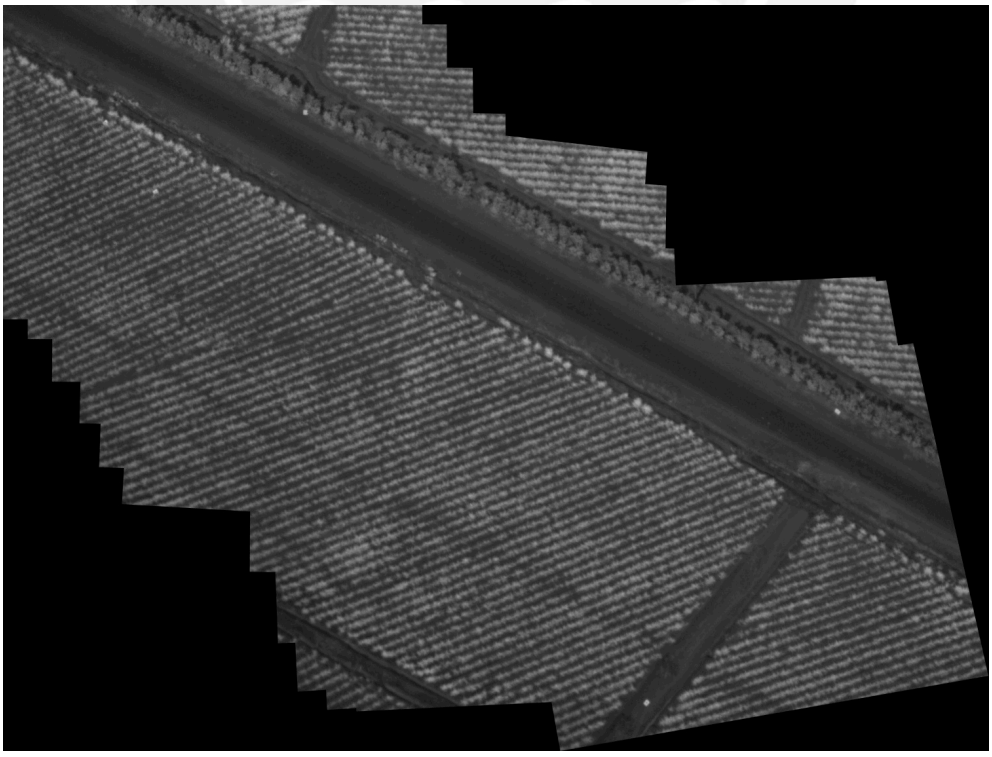

Figura 4.3 Mosaico de 25 imágenes de un campo de caña de azúcar.

Cabe señalar que cuando un imagen es transformada no produce errores de detección en general a menos que la transformación no sea correcta, en dicho caso el resultado final será erróneo ya que se arrastrará el error.

En el siguiente cuadro se muestran los tiempos obtenidos luego de ejecutar los cuatro algoritmos para la figura anterior.

	SIFT	SURF	BRISK	FREAK
Cantidad de imágenes	25	25	25	25
Tiempo consumido	87.0897	42.7965	18.6683	17.8534

La siguiente imagen mosaico corresponde al jardín central de la PUCP y posee 36 imágenes en formato .pgm y fueron conseguidas mediante un vuelo automatizado.

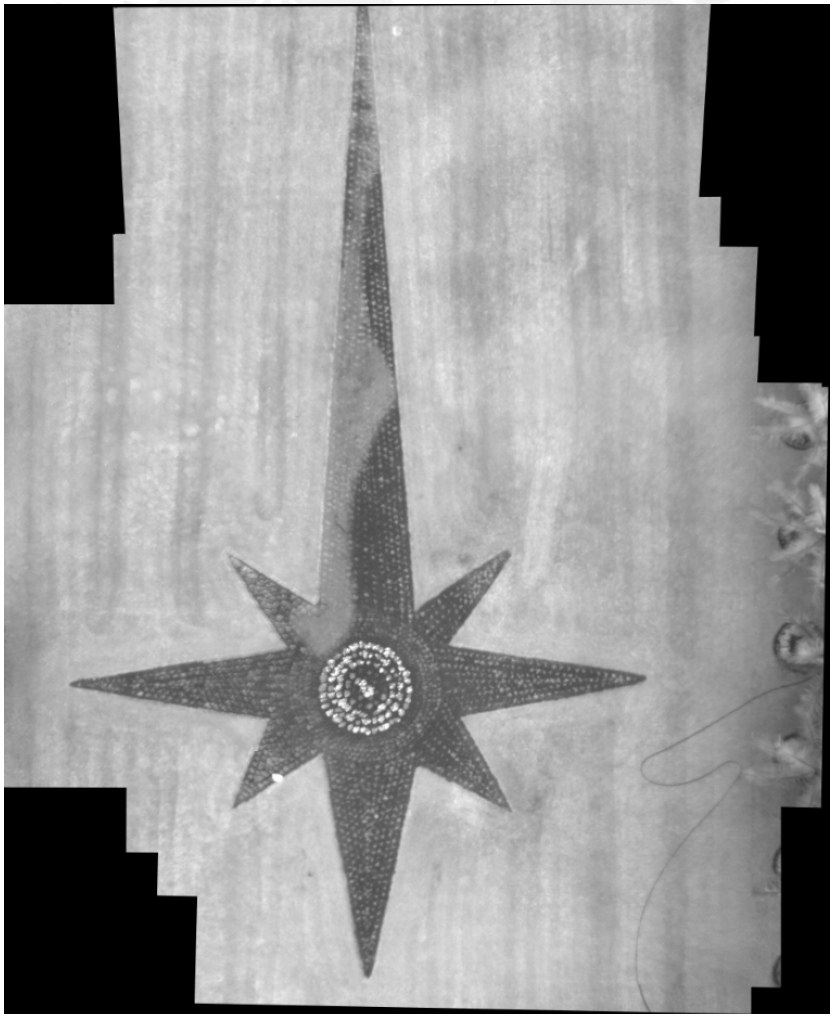

Figura 4.4 Mosaico de 36 imágenes del jardín central PUCP.

4.3 Discusión de resultados

En esta sección se analizan distintos puntos de vista después de haber realizado una gran cantidad de pruebas.

En los cuadros mostrados se observa una gran cantidad de puntos de interés seleccionados, sin embargo muchos de esos puntos en una imagen tienen una correspondencia errada en la otra, por ello se realiza un filtro donde solo las mejores correspondencias sobreviven, ello con el fin de obtener una matriz de transformación fiel a la realidad, debido a la naturaleza del algoritmo de registro de imagen, si tenemos un error, este será arrastrado en las siguientes imágenes, obteniendo distorsiones apenas apreciables hasta una deformidad total (ininteligible) de la imagen resultante.

Cabe resaltar que los algoritmos BRISK y FREAK al poseer descriptores binarios poseen un proceso de formación de dicho vector de una manera muy acelerada, además de poseer la ventaja de una rápida evaluación de correspondencias, son simples comparaciones binarias (XOR), lo cual explica la rapidez de citados algoritmos al realizar un mosaico de muchas imágenes.

En los cuadros mostrados anteriormente se nota una predisposición a obtener menor cantidad de puntos de control cuando se usa algoritmos de vector descriptor binario, en muchas ocasiones este ha sido un problema ya que al ejecutar el algoritmo la cantidad de puntos de interés obtenidos y que superan el filtro de mejores correspondencias quedan cortos, es decir muchas veces la cantidad de puntos restante no alcanza el mínimo requerido por el algoritmo RANSAC para formar la matriz de homografía, obteniendo un error en tiempo de ejecución.

Cabe resaltar que en las pruebas que se realizaron con el algoritmo FREAK, ya que solo tiene proceso de descripción, se hizo uso del algoritmo BRISK para detectar puntos de interés, esto ayudó sobremanera para que FREAK sea el algoritmo más rápido y además que logre invariancia ante la escala y cambios de iluminación.

CONCLUSIONES

1. Podemos concluir que el sistema de registro de imágenes realizado con diferentes algoritmos (SIFT, SURF, BRISK y FREAK) nos sirvieron para encontrar las relaciones de transformación y así lograr una correcta proyección de las imágenes con su imagen referencia, al tener una mayor área en común se observa mejores resultados ya que obtenemos un mayor número de coincidencias.
2. Podemos afirmar que en términos de tiempo y costo computacional el algoritmo FREAK es el que mejor responde a las pruebas realizadas debido a la facilidad de procesamiento de su vector descriptor binario, sin embargo se debe remarcar que a veces tiende a fallar dado la menor cantidad de descriptores que se obtienen de las imágenes de prueba.
3. En términos de calidad y cantidad de descriptores podemos citar al algoritmo SIFT como el más robusto, además de remarcar que casi no obtuvimos errores de registro, pero debido a la gran dimensionalidad de su vector descriptor requiere de grandes recursos computacionales además de procesar las imágenes en un mayor tiempo, a su vez SURF también tiene respuesta parecida a SIFT pero debemos indicar que se cometen ciertos errores cuando la rotación de las imágenes es mayor de 40 grados.
4. Siempre existe la posibilidad de encontrar errores de transformación, dado que estas son de tipo proyectivas, en determinados casos esta no es suficiente para calzar con el mosaico de imágenes anteriores, cuando este fenómeno se produce se arrastra un error que hace que el resultado no sea reconocible, en estos casos es mejor retirar la imagen problema de la batería de imágenes a procesar para lograr los resultados esperados.

RECOMENDACIONES

Para el uso de los algoritmos binarios se debe tomar en cuenta que no se obtendrá una gran cantidad de puntos de control debido a la poca diferenciación de zonas y bajo contraste que propiciará errores de correspondencias, por tal motivo sería recomendable agregar algunas marcas en la zona de captura de imágenes, de esta manera tendremos puntos “artificiales” de alta diferenciación.

Dado la superioridad aparente del algoritmo FREAK en términos de tiempo y costo computacional, podemos aprovechar su modularidad y usar un algoritmo robusto como SURF para la detección de puntos de interés para luego crear el descriptor binario con el algoritmo FREAK, lo cual proporcionará un performance equilibrado.

Para obtener un menor error de registro en el mosaico sería altamente recomendable el uso de imágenes orto-rectificadas con anterioridad, este proceso es posible realizando un sensado continuo del aeromodelo tal que se obtengan datos de su posición angular en cada momento que se realice una captura aérea.

El presente trabajo se puede usar como referencia para usar sistemas de procesamiento embebido que puedan realizar el registro de imágenes mientras el aeromodelo se encuentra en vuelo, obteniendo en tiempo real el área barrida por la aeronave y así asegurar una captura completa de toda el área de interés.

BIBLIOGRAFÍA

[1] Centro Internacional de la papa (CIP).

[2] FAO. “Importancia de la agricultura en la actualidad” [en línea]. Disponible en: http://www.fao.org/docrep/008/a0015s/a0015s04.htm Consultado el 19/11/14.

[3] R. Laganière, *OpenCV 2 Computer Vision Application Programming Cookbook*. Birmingham, UK: Packt Publishing, 2011.

[4] P. Deitel, H. Deitel, *C++ How to Program*, 8va Edición. Boston, USA: Pearson Education Inc, 2012.

[5] A. Mendez, “La agricultura de precisión toma vuelo”, Universo MAQ [en línea]. Disponible en: http://www.universomaq.com/noticias2/158-la-agricultura-de-precision-toma-vuelo Consultado el 06/10/14.

[6] Ovalles, V., F.A. 2006. “Introducción a la agricultura de presición”. Revista Digital CENIAP HOY No 12 septiembre-diciembre 2006, Maracay, Aragua, Venezuela. Disponible en: http://sian.inia.gob.ve

[7] M. Canamero,. “Aplicaciones de la agricultura de Precisión” [en línea]. Disponible en: http://www.kriego.net/ap3.htm Consultado el 07/10/14.

[8] APM Multiplatform Autopilot. “Mission Planner | Ground Station” [en línea]. Disponible en: http://planner.ardupilot.com Consultado el 03/10/14.

[9] Ministerio de Agricultura. “Papa, Indicadores básicos de cultivo” [en línea]. Disponible en: http://www.minag.gob.pe/portal/sector-agrario/agricola/cultivos-de-importancia-nacional/papa Consultado el 06/11/2014.

[10] K. Valavanis, *Advances in Unmanned Aerial Vehicles. State of the Art and the Road to Autonomy*, vol. 33. Florida, USA: Springer, 2007

[11] Ministerio de Agricultura. “Problemas en la agricultura en el peruana” [en línea]. Disponible en: http://www.minag.gob.pe/portal/sector-agrario/agricola/vision-general/problemas-en-la-agricultura-peruana
Consultado el 05/11/2014.

[12] D. Lowe, “Distinctive image features from scale-invariant keypoints,” International Journal of Computer Vision (2004).

[13] H. Bay, T. Tuytelaars, and L. Van Gool. “SURF: Speeded up robust features,” Computer Vision and Image Understanding (CVIU), 110(3):346–359, 2008.

[14] S. Leutenegger, M. Chli, and R. Siegwart. “Brisk: Binary robust invariant scalable keypoints,” 2011.

[15] A. Alahi, R. Ortiz, and P. Vandergheynst. “Freak: Fast Retina Keypoint.” 2012.

[16] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger. “Adaptive and generic corner detection based on the accelerated segment test,” In Proceedings of the European Conference on Computer Vision (ECCV), 2010.

[17] E. Rosten and T. Drummond. “Machine learning for high-speed corner detection,” In Proceedings of the European Conference on Computer Vision (ECCV), 2006.

[18] B. Fan, Q. Kong, X. Yuan, Z. Wang, C. Pan. “Learning weighted hamming distance for bynary descriptors”.

[19] C. Evans. “Notes on the OpenSURF Library”. 2009.

[20] R. Rojas. “Diseño de un sistema de registro de imágenes orientado a la agricultura de precisión”. PUCP, 2009.

[21] J. Barba. “Registro de imágenes para agricultura de precisión usando lenguaje C”. PUCP, 2012.

[22] OpenCV documentation. Features2d. 2D Features Framework [en línea]. Disponible en: http://docs.opencv.org/modules/features2d/doc/features2d.html, Consultado el 13/08/14.

[23] Stackoverflow [en línea]. Disponible en: http://stackoverflow.com/questions/tagged/opencv Consultado el 20/08/14.

[24] K. Mikolajczyk “Affine covariante feartures,” [en línea]. Disponible en: http://www.robots.ox.ac.uk/~vgg/research/affine/