CONSTRUCCIÓN DE UN MAPA DE SUPERFICIE CON INFORMACIÓN CAPTURADA DESDE UN UAV

INGENIERA ELECTRÓNICA
JOHANA MARÍA FLÓREZ LOZANO

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE ELECTRÓNICA
MAESTRÍA EN INGENIERÍA ELECTRÓNICA
BOGOTÁ DC
2014

CONSTRUCCIÓN DE UN MAPA DE SUPERFICIE CON INFORMACIÓN CAPTURADA DESDE UN UAV

ING. JOHANA MARÍA FLÓREZ LOZANO

INFORME FINAL DE PROYECTO DE INVESTIGACIÓN PARA OPTAR POR EL TITULO DE MAGISTER EN INGENIERÍA ELECTRÓNICA

DIRECTOR: ING. CARLOS ALBERTO PARRA Ph.D
CODIRECTOR: ING. JULIÁN DAVID COLORADO Ph.D

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE ELECTRÓNICA
MAESTRÍA EN INGENIERÍA ELECTRÓNICA
BOGOTÁ DC
2014

Índice general

Índice de figuras

Índice de cuadros

Agradecimientos

Este libro es el capitulo final de una de las fases academica de mi vida, además es el fruto del trabajo de varios meses, horas, días y minutos de dedicación permanente. En los cuales, siempre tuve el apoyo de las personas más cercanas a mi, entre ellos mis padres, mi querido novio, mis amigos y mis queridos directores.

Quiero agradecer a cada uno de ellos por estar siempre a mi lado y confiar en mis capacidades.

A mis padres un especial saludo y agradecimiento de todo corazón, por siempre apoyarme en mis proyectos, por estar presentes en cada día de mi vida y estar pendientes de mi. Espero siempre llenarlos de orgullo y poderles retornar todo el cariño y amor que me han brindado.

A mi novio Francisco por ser esa persona especial, a la que amo con todo mi corazón, fuente de inspiración, que me brinda la fuerza para superarme cada día, para llegar a ser mejor persona y una excelente profesional.

A mis amigos y compañeros, les agradezco por incluir esos minutos al día para darme sus buenos deseos y aportar con ideas frescas en este proceso.

Y a mis queridos directores, les agradezco por brindarme el apoyo sin condición, en cada una de las etapas del proyecto, por ser más que un par de guías, ser un par de amigos en los cuales puedo confiar. Muchas gracias por escuchar, corregir y aportar en cada una de las líneas del presente libro.

Abreviaturas y siglas

UAV	*Unmanned Aerial Vehicle*
ROS	*Robot Operating System*
IMU	*Inercial Measurement Unit*
QVGA	*Quarter Video Graphics Array*
OpenCV	*Open Source Computer Vision Library*
VTOL	*Vertical Take-Off and Landing*
HTA	*Heavier Than Air*
θ	*Pitch* - rotación en eje y
ϕ	*Roll* - rotación en eje x
ψ	*Yaw* - rotación en eje z
DOF	*Degrees Of Freedom*
FEK	Filtro Extendido de Kalman
SIFT	*Scale Invariant Feature Trasform*
SURF	*Speeded Up Robust Features*
MSER	*Maximally Stable Extremal Regions*
FAST	*Features from Accelerated Segment Test*
ORB	*Oriented FAST and Rotated BRIEF*
BRISK	*Binary Robust Invariant Scalable Keypoints*
FREAK	*Fast Retina Keypoint*
BRIEF	*Binary Robust Independent Elementary Features*
LoG	*Laplacian of Gaussians*
DoG	*Difference of Gaussians*
RANSAC	*Random Sample Consensus*
SLAM	*Simultaneous localisation and mapping*
SAD	*Sum of Absolute Difference*
SSD	*Sum of Square Difference*
GPS	*Global Positioning System*
SDF	*Simulation Description Format*
KNN	*K-Nearest Neighbor*
Flann	*Fast Approximate Nearest Neighbor Search Library*
MAP	Mean Average Precision

Capítulo 1

Introducción.

Este documento recopila la información teórica y práctica, requerida y obtenida durante el desarrollo del proyecto de investigación titulado: "Construcción de un mapa de superficie con la información capturada desde un *UAV*", el cual tiene como objetivo obtener una sola imagen a partir de una serie de imágenes a lo cual se le denomina mosaico de imágenes. Las imágenes son capturadas con una de las cámaras embarcadas en un vehículo aéreo no tripulado *UAV* comercial e integradas en la imagen final (mosaico) de forma secuencial. Este tipo de aplicación puede llegar a ser útil en diferentes campos de acción, entre ellos las relacionadas con la agricultura [2], la vigilancia [3] u otras aplicaciones enfocadas al control y localización visual de una plataforma aérea [4, 5].

El *UAV* comercial del presente trabajo es el *ARDrone 2.0*, que dispone de sensores de orientación y cámaras vertical y horizontal, de los cuales se empleó, la información de orientación para calcular la matriz de homografía (detalle en la sección 4.3.1) y las imágenes tomadas con la cámara vertical, para crear un mapa[1] de una zona de interés en tiempo real de desplazamiento. Adicionalmente se emplearon herramientas software *Open Source* para robótica, entre ellas *ROS* y *Gazebo* las cuales fueron usadas para establecer las comunicaciones con la plataforma, ejecutar el algoritmo de generación de mapa y comprobar el funcionamiento del algoritmo implementado, con la plataforma en escenarios de simulación diseñados (ver sección 4.1.2.2).

La fusión o generación de mosaicos de imágenes es una herramienta clásica de la visión por computador, que se compone de dos fases, la más importante por su gran utilidad en diferentes campos, es la fase de registro en la cual tiene como etapa fundamental el uso de detectores y descriptores de puntos en la imagen. Etapa en el cual, el presente proyecto aporta una metodología de evaluación asociada con el escenario, la cual permite tener argumentos de selección de un conjunto detector-descriptor. Para posteriormente realizar los ajustes correspondientes para la aplicación específica. Solo se seleccionó, un conjunto detector-descriptor de entre las técnicas más recientes del estado del arte [6, 7, 8, 9, 10] para ser empleada dentro del proceso de registro y mejorar la captura de información.

Por otra parte, los mapas han sido una herramienta valiosa a través del tiempo para los

[1] Para el caso de este proyecto se define mapa como la representación bidimensional de una zona geográfica de la tierra mediante una imagen.

seres humanos, entre sus objetivos están la captura y agrupación de información perteneciente a características de un lugar o un área en particular. Además permiten tener un conocimiento previo del lugar lo que facilita la futura localización y generación de una posible ruta de desplazamiento.

Han surgido propuestas de mapas que se construyen mediante la fusión de información que ha sido adquirida con cámaras embarcadas en satélites [11], también denominadas observaciones globales, las cuales han permitido tener niveles de información geográfica desde continentes hasta el punto de ciudades, de tal forma que permite incluir información adicional de localización global y direccionamiento local.

En general, la información de imágenes satelitales ha permitido aumentar el conocimiento previo de una zona. Aunque, esta información puede no ser confiable debido a que las imágenes no son actualizadas en tiempo real. Esto no permite garantizar que la información presentada en estos mapas sea correcta, incluyendo que no todos los lugares tienen la misma resolución, lo que implica información incompleta de la zona. Lo cual, hace que el uso de esta información sea útil para aplicaciones de ubicación geográfica, más sin embargo no son útiles para aplicaciones en las cuales se requiere información reciente como en los campos de agricultura, vigilancia o mantenimiento, entre otras.

Entre las ramas económicas más importantes en Colombia se tienen el sector energético y de transporte o transmisión de fluidos mediante tubería, en los cuales el mantenimiento preventivo es una herramienta para mejorar la eficiencia del sistema (calidad del servicio, confiabilidad, cobertura). Estos sectores son ejemplos en los cuales la recolección de información de interés, la cual se puede capturar en mapas de la zona, contribuiría en la reducción de accidentes y la ejecución de un mantenimiento preventivo.

Para finalizar, se contempló establecer un protocolo de pruebas en un ambiente controlado y evaluar los resultados obtenidos. El ambiente seleccionado para las pruebas, fue construido en el laboratorio de robótica. Haciendo referencia a la transmisión eléctrica mediante la elaboración de una maqueta de líneas de transmisión con el apoyo de un proyecto de la Universidad financiado por Colciencias, cuyos detalles técnicos de diseño y construcción se especifican en la sección 4.1.1.3.

En general, en el siguiente documento se presentan las siguientes secciones: el marco teórico en el cual se presentan las temáticas principales: la plataforma robótica elegida y sus características, la generación de trayectorias, la fusión sensorial mediante el uso del Filtro de Kalman y se detalla el funcionamiento de cada uno de los detectores y descriptores que fueron incluidos en la evaluación, selección y adaptación para la fase de registro en generación de mosaico; también se presentan las secciones de las especificaciones genéricas del sistema en la cual se encuentra la descripción detallada del sistema con un diagrama en bloques que permite comprender el funcionamiento del sistema, la sección del desarrollo tanto de hardware, como de software del proyecto de investigación, en la cual se pueden encontrar el protocolo de las pruebas finales que se especifican para garantizar el correcto desempeño del sistema; la sección de resultados y conclusiones que presentan los análisis e información obtenida en cada una de las fases del proyecto.

Capítulo 2

Marco Teórico

2.1. Plataforma robótica

La plataforma robótica elegida para este proyecto es un vehículo aéreo, cuya configuración consta de cuatro rotores, denominado quadrotor o cuatrirotor. El cual hace parte del conjunto de vehículos aéreos no tripulados *UAV*s que tienen la capacidad de desplazarse en forma *VTOL*, lo cual implica que el avión vuela sin piloto a bordo y que es totalmente controlado de forma remota (tierra, otro avión, el espacio) o programado para ser completamente autónomo.

Usualmente, estos vehículos aéreos tienen embarcados diferentes elementos como cámaras, magnetómetros, acelerómetros, altímetros [1], procesadores que pueden ser empleados para ejecución de tareas o servir como complemento del control aéreo, ya sea ejecutado de forma remota o de manera autónoma [4, 12, 13, 14, 15], debido a esto, estas plataformas aéreas no tripuladas se han convertido en un área de investigación activa y cuenta con un gran potencial para el desarrollo de aplicaciones hacia la sociedad civil. Algunas de las tareas en las que son empleadas los *UAV* son: el control de fronteras, la lucha contra incendios, aplicaciones de vigilancia [3, 16, 17], tareas de rescate, control de contaminación, tareas en el sector agrícola [2], entre otros.

En el presente proyecto se empleó el sistema de referencia del quadrotor, el sistema de referencia global y la numeración de los motores embarcados que se presentan la figura 2.1, además se tiene que los ángulos de rotación de la plataforma son:

1. Roll (ϕ) - Rotación en el eje X.
2. Pitch (θ)- Rotación en el eje Y.
3. Yaw (ψ)- Rotación en el eje Z.

Es usual que en las plataformas de quadrotor, las cargas útiles estén localizadas en el centro del cuerpo y los rotores sean impulsados con motores de corriente continua, con o sin caja de cambios. Además los rotores se dividen en dos pares, un par gira en la dirección opuesta a la otra pareja con el fin de equilibrar el momento angular del sistema tal y como se ilustra en la figura 2.1.

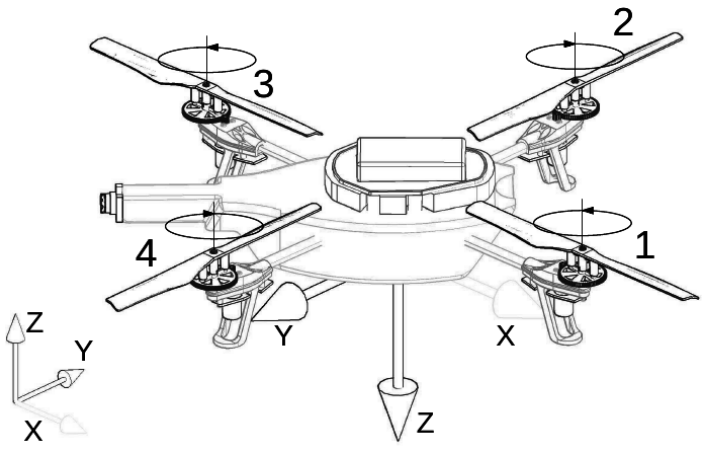


Figura 2.1: Sistema de referencia global y numeración de motores del *UAV* comercial *ARDrone 2.0*, para el presente proyecto

2.1.1. Plataforma Comercial

Para este proyecto, se seleccionó el *UAV* comercial denominado como ARDrone 2.0[1] cuyas especificaciones son:

- En la figura 2.2se muestra la estructura y partes generales de la plataforma comercial.
- En el Cuadro 2.1 se presentan las características numeradas en la figura 2.2.
- Características de la plataforma, no incluidas en la figura 2.1 son:

  1. Giróscopo de 3 ejes +/- 2.000°/s.
  2. Acelerómetro de 3 ejes +/- 50mg.
  3. Magnetómetde la siguiente formaro de 3 ejes, precisión 6°.
  4. Sensor de presión +/- 10 PA (80cm a nivel del mar).
  5. Sensor de ultrasonido con alcance de 6 m.
  6. Cámara QVGA[2] vertical, 60 fps .
  7. Peso con carcasa para exterior de 320 gr.
  8. Peso con carcasa para interior de 420 gr.

[1]Imagen e información tomada de: http://ardrone2.parrot.com/ardrone-2/especificaciones/
[2]Resolución gráfica de 320x240 píxeles.

Figura 2.2: Esquema del *ARDrone 2.0*, complementario al Cuadro 2.1. En el cual, se indican cada una de las partes de la plataforma robótica

Con respecto al método de navegación de la plataforma, se tiene el diagrama de bloques de la arquitectura interna dada por el fabricante en la Figura 3.2a y explicado en detalle en [1]. El ARDrone es una estructura que ha sido previamente elaborada, evaluada y de la cual ya se tiene un conocimiento previo de operación, esto con el fin de enfatizar los esfuerzos de la investigación en el uso de la plataforma. Además, entre las ventajas de manejar una plataforma comercial, se tiene la posibilidad de duplicar los resultados finales en múltiples plataformas o en otros campos de aplicación, a esto se añade la facilidad de mantenimiento de la plataforma con el reemplazo de partes o en casos de no ser suficiente el reemplazo de las piezas, el cambio total de la plataforma sin mayor afectación en el avance del proyecto o para el usuario final.

2.2. Generación de trayectorias

Esta trata el problema de diseñar el camino que un vehículo debe seguir con el fin de alcanzar una meta. En general, hay dos enfoques diferentes para construir este camino: la planificación de la trayectoria abordo y la planificación de trayectoria fuera de línea, en el caso de este proyecto se busca aprovechar una trayectoria continua fuera de línea. En la literatura actual existen métodos de generación de trayectorias [18], entre ellos los que se pueden considerar determinísticos tales como: descomposición en celdas, campos de potenciales, diagramas de Voronoi, grafos de visibilidad o los métodos considerados probabilísticos y aleatorios como: planificador aleatorio de trayectorias *RPP*, mapas probabilísticos *PRM*, métodos basados en optimización (Algoritmos Genéticos *GA*, enjambre de partículas *PSO*) [19] estos últimos pueden llegar a plantear la trayectoria en

No.	Descripción
1	Cámara HD 720p, 30fps[a]
2	Objetivo gran angular: 92°, diagonal
3	Procesador ARM Cortex A8 de 32 bits a 1Ghz[b]
4	Tubos de fibra de carbono
5	Piezas de plástico nylon cargado 30 % de fibra de vidrio de alta calidad
6	Espuma para aislar el centro de inercia de las vibraciones de los motores
7	Casco de protección de polipropileno expandido (PPE)
8	Nano revestimiento repelente a los líquidos en los sensores de ultrasonido
9	4 motores de rotor interno sin escobillas.[c]
10	Rodamiento de esferas miniatura
11	Engranajes de Nylatron de bajo ruido para reductor de hélice de 1/8', 75
12	Eje de las hélices de acero templado
13	Cojinete de bronce auto-lubricante
14	Alta fuerza de propulsión de las hélices para mayor maniobrabilidad
15	Microcontrolador AVR de 8 MIPS por controlador de motor
16	Batería Recargable Li-Po de 3 celdas, 1000 mA/h
17	Puerto para memoria USB o dispositivo remoto
18	*Flight Recorder, GPS* y memoria interna flash de 4GB

[a]Perfil de codificación básica de H264 (H264 o MPEG-4 es una norma que define un códec de video de alta compresión).

[b]Con DSP de video TMS320DMC64x a 800Mhz, SO: Linux 2.6.32, RAM DDR2 de1GB, USB 2.0, Wifi b.g.n

[c]Con un consumo de 14,5W y 28,5W al quedar suspendido en el aire

Cuadro 2.1: Listado de características de la plataforma comercial ARDrone 2.0, complementaria a la Figura 2.2.

búsqueda de minimizar el trayecto entre dos puntos o minimizar el gasto energético. Sin embargo, en múltiples aplicaciones la duración de la misión es dada, y el objetivo principal es maximizar la recolección de información en un tiempo fijo como se presenta en [20], en el cual se emplean algoritmos genéticos como método de optimización.

2.3. Fusión sensorial

Para el presente proyecto se realiza filtrado de la información sensorial, para los cual, se emplea un filtro extendido de Kalman, previamente implementado como un nodo de *ROS*. La desarrollo teórico de este tipo de filtros es presentada en el anexo 7.2.

2.4. Generación de mosaico

Se tiene como propósito del proyecto la elaboración del mapa de una zona, para el cual se plantea aplicar técnicas de fusión de imágenes con la información sensorial de orientación adicional a las cámaras; la fusión de imágenes es una herramienta clásica de la visión por computador entre sus aplicaciones más usuales se encuentra la creación de las observaciones satelitales [11] y es empleada en conjunto con estabilización de imagen en las cámaras digitales para lograr imágenes panorámicas [21, 22].

Los métodos para la alineación y fusión de imágenes automáticas normalmente se dividen en dos categorías: método directo y basado en características. Para el caso directo, se intenta estimar iterativamente los parámetros de la cámara, reduciendo al mínimo una función de error con base a la diferencia de intensidad en la zona de solapamiento. Tienen la ventaja de usar todos los datos de imagen disponibles, sin embargo se requiere inicialización. En cambio, para los esquemas basados en características no se requiere inicialización, son muy robustos, e incluso se pueden utilizar para el reconocimiento de objetos en imágenes muy separadas, estas características no sólo responden a las regiones con alta concentración de bordes, sino también a las áreas uniformes [21, 22]. El presente proyecto se enfoca en el esquema basado en características y se toma como estructura base el algoritmo 2.1 en el cual se presenta la ejecución para la construcción de un mosaico de imágenes, en el cual los descriptores son adquiridos empleando *SIFT* [21].

Por otra parte, la creación de mosaicos de imágenes ha tenido un continuo mejoramiento del hardware y ha incursionado en diferentes campos de acción. Entre los trabajos más recientes se denota el interés en la reducción de tiempos o eficiencia requeridos para fusionar las imágenes [23] enfocados para la ejecución en tiempo real, dando ajustes a las técnicas existentes y ejecutando el sistema de fusión en sistemas hardware más elaborados [16]. Entre otras aplicaciones se tiene la fusión de imágenes como herramienta en el campo de la vigilancia [3, 24], incluyendo las que han sido evaluadas desde un *UAV* [17, 25, 26].

2.4.1. Funcionamiento de detectores y descriptores de características.

Entre las técnicas de obtención de características que se han usado para fusión de imágenes tomadas desde un *UAV* se tienen: el detector de esquinas de Harris [27] el cual no se evaluará en el presente proyecto por ser considerado ya clásico en la literatura, también se presenta la transformación de características invariantes de escala *SIFT* [28, 21] la más usual debido a que los descriptores se operan en escala-espacio y utilizan una orientación dominante, pueden coincidir entre las imágenes que difieren en escala y orientación, o finalmente las características robustas aceleradas *SURF* [3, 29] que son el resultado de algunas mejoras realizadas a la técnica de *SIFT*.

Sin embargo, han surgido en el estado del arte otras técnicas de obtención de características, las cuales se van a estudiar, adecuar y evaluar en el transcurso del presente proyecto y estas son: *MSER* [6], *FAST* [7], *BRIEF* [30], *ORB* [8], *BRISK* [9] y *FREAK* [10].

Algoritmo 2.1 Descripción del algoritmo de mosaico automático descrito en la referencia [21], tomado como base de desarrollo del algoritmo del presente proyecto.

Entrada: n imágenes desordenadas

1. Extraer las características *SIFT* de todas las n imágenes.
2. Encontrar los k vecinos más cercanos para cada una de las características, empleando $k-dtree$.
3. Para cada imagen:
   a) Seleccionar m imágenes candidatas que coincidan, en otras palabras que tengan el mayor número de características coincidentes con la imagen base.
   b) Encontrar geométricamente que coincidan las características usando *RANSAC* para solucionar la homografía entre cada par de imágenes.
   c) Verificar la coincidencia entre imágenes usando un modelo probabilístico.
4. Encontrar los componentes conectados de la imagen coincidente.
5. Por cada componente:
   a) Ejecutar un agrupamiento para resolver las rotaciones y las distancias focales de las cámaras.
   b) Hacer el panorama usando una mezcla multi-banda.

Salida: Imagen total.

Para poder adecuar cada una de estas técnicas es necesario conocer su enfoque y funcionamiento, para lo cual se presentan a continuación la información básica para conocer la ejecución de cada una de estas técnicas.

2.4.1.1. Transformación a características invariantes a escala (*SIFT*)

El reconocimiento de los objetos en el mundo real requiere características que no se afecten por desorden u oclusiones parciales. Adicionalmente las características deben por lo menos ser parcialmente invariantes a la iluminación, las proyecciones de perspectiva y variaciones comunes del objeto. Por otro lado, las características deben ser suficientemente distintivas para identificar un objeto entre varias alternativas.

SIFT es un método que transforma una imagen en una colección de vectores de características locales, cada una de ellas invariantes a la imagen, traslación, escalado y rotación, incluyendo la no varianza parcial a los cambios de iluminación y transformaciones a fin o de perspectiva. Las características obtenidas con esta transformación comparte propiedades con la respuesta de las neuronas en la corteza inferior temporal de la visión de los primates.

Las características invariantes de escala son eficientemente identificadas usando una aproximación de filtros.

La primera etapa identifica las claves de localización, para ello se parte de que Lindeberg [31] demostró que bajo algunos supuestos generales en la escala invariante, el kernel Gaussiano y sus derivadas son el único kernel de suavizado para el análisis en el espacio escala.

Para lograr la no varianza en la rotación y un alto nivel de eficiencia, se seleccionan las claves de localización mediante los máximos y mínimos de una diferencia de la función Gaussiana. Esta puede ser calculada eficientemente mediante la construcción de una imagen piramidal con sub-muestreo entre niveles, esta técnica ubica las claves de localización en regiones y escalas de alta variación, haciendo de estas localizaciones características particularmente estables dentro de la imagen. Esta diferencia Gaussiana *DoG* usada en el espacio escala fue usada previamente por Crowley & Parker [32] y por Linderberg [33], sin embargo con propósitos diferentes.

Para las claves de localización, todas las operaciones de suavizado consisten en ejecutar la convolución en cada orientación con una función Gaussiana unidimensional con $\sigma = \sqrt{2}$. En la Figura 2.3 se presenta en resumen el proceso para obtener la diferencia Gausiana y los niveles de la pirámide. La interpolación bilineal empleada para sub-muestrear, de espaciado de 1,5 implica que cada nueva muestra será una combinación lineal constante de los 4 píxeles adyacentes.

La máxima y mínima en la función de espacio escala es determinada por la comparación de cada píxel en la pirámide con sus vecinos. Primero es píxel es comparado con sus 8 vecinos en el mismo nivel de la pirámide y si es máximo o mínimo en este nivel, se localiza el píxel más cercano en el siguiente nivel inferior de la pirámide, teniendo en cuenta el sub-muestreo de 1.5. Si el píxel permanece como el mínimo o máximo entre los píxeles cercanos y sus 8 vecinos, luego la prueba es repetida en el nivel inferior. Dado que los píxeles van a ser eliminados con pocas comparaciones, el costo de esta detección es pequeño y no mucho más bajo que la construcción de la pirámide.

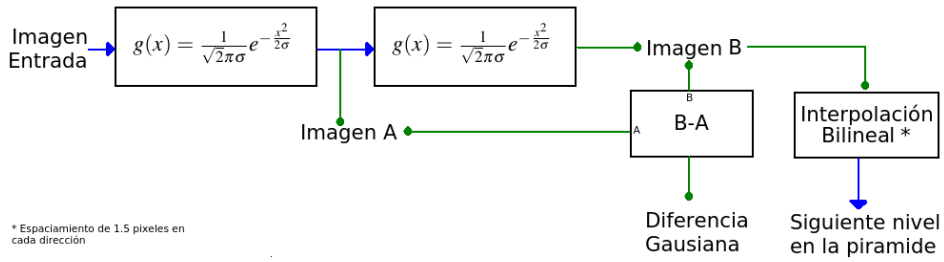


Figura 2.3: Diagrama explicativo de la obtención de las claves de localización con el descriptor SIFT

Para la segunda etapa se tiene que para caracterizar la imagen, se toma la imagen suavizada A de cada uno de los niveles de la pirámide y es procesada para extraer la

magnitud de los gradientes M_{ij} y orientación R_{ij} de la imagen, en cada uno de los píxeles A_{ij}. Los cuales se calculan mediante las ecuaciones 2.1 y 4.5.

$$M_{ij} = \sqrt{\left(A_{ij} - A_{i+1,j}\right)^2 + \left(A_{ij} - A_{i,j+1}\right)^2} \tag{2.1}$$

$$R_{ij} = atan2\left(A_{ij} - A_{i+1,j}, A_{i,j+1} - A_{ij}\right) \tag{2.2}$$

La diferencias de píxeles son eficientes al calcular y proveer suficiente precisión debido al nivel sustancial de suavizados previos. La robustez a la iluminación es lograda por un umbral en la magnitud del gradiente de un 10 % del valor máximo calculado. La orientación es determinada por el pico en el histograma de las orientaciones de los gradientes locales de la imagen. El histograma de orientación es creado usando una ventana de pesos Gaussiana con $\sigma = (3 * k)$, k : Escala de suavizado actual. Estos pesos son multiplicados por los valores del umbral del gradiente.

Dada una localización estable, la escala, y la orientación por cada clave. Ahora es posible describir una región en la imagen de tal manera que sea no sea variante a este tipo de transformaciones. Para indexar se necesita almacenar las claves *SIFT* de imágenes de ejemplo y luego identificar las claves que coincidan de las nuevas imágenes, el problema para identificar la mayoría de las claves similares es la alta dimensión de los vectores de características. Sin embargo, una modificación al algoritmo *k-d tree* llamado método de búsqueda *best-bin-first* puede identificar los vecinos cercanos con alta probabilidad solamente usando una cantidad limitada de recursos computacionales.

Después de este proceso, el grupo de claves *SIFT* que coinciden con el modelo planteado son identificados mediante la transformada *Hough* y luego se predice la localización, orientación y escalamiento entrando datos a una tabla *hash*.

2.4.1.2. Características robustas aceleradas (*SURF*)

El problema de encontrar la correspondencia entre dos imágenes consta de varias fases. La primera fase es encontrar puntos de interés dentro de la imagen tales como esquinas, uniones en T, bordes o estructuras, teniendo en cuenta que la propiedad más valiosa de un detector es la repetibilidad, la cual consiste en garantizar la confiabilidad de detectar los mismos puntos físicos de interés bajo diferentes condiciones de orientación. La segunda fase consiste en establecer una vecindad alrededor del punto de interés y representarlo por un vector de características denominado descriptor, finalmente los vectores de descriptores son emparejados entre imágenes basados en separación por distancia, los cuales son calculados usualmente mediante distancia Euclidiana o Mahalanobis.

El tamaño del descriptor es un elemento que impacta el tiempo de ejecución del algoritmo, entre menor tamaño pueda este tener, es posible realizar el emparejamiento entre puntos en menor tiempo, sin embargo entre menor tamaño tengan los descriptores de los puntos son menos distintivos.

El propósito principal de *SURF* es hacer que el cálculo de los descriptores sea más rápido, en comparación a las técnicas previas en el estado del arte tales como *LoG* y *DoG*, mientras

no se sacrifique el desempeño del descriptor.

Detector de puntos de interés.

Se usa una aproximación básica de la matriz Hessiana. Esto en si mismo permite emplear imágenes integrales. Las imágenes integrales permiten hacer operaciones con filtros de convolución más rápidas a nivel computacional. La entrada de una imagen integral $I_{\Sigma}(x)$ en el punto $\mathbf{x} = (\mathrm{x}, \mathrm{y})^{\mathbf{T}}$ representa la suma de todos los píxeles en la entrada de imagen I alrededor de un área rectangular alrededor del punto $\mathbf{x}$.

$$I_{\Sigma} = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j) \tag{2.3}$$

Una vez se haya calculado la imagen integral , toma tres sumas para calcular la suma de intensidades sobre cualquier, área rectangular como se muestra en la Figura 2.4[3]. Por lo tanto, el tiempo de cálculo es independiente del tamaño de la ventana.

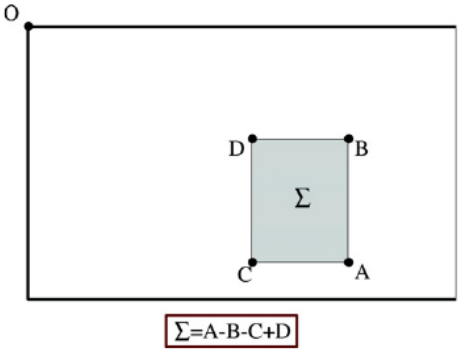


Figura 2.4: Esquema explicativo, del cálculo de la intensidad en una región de una imagen integral aplicando la Ecuación 2.3, para *SURF*.

La base del detector *SURF* se basa en matrices Hessianas por su buen comportamiento para la precisión. Dado un punto $\mathbf{x} = (\mathrm{x}, \mathrm{y})$ en una imagen I la matriz Hessiana $\mathscr{H}(\mathbf{x},\sigma)$, en $\mathbf{x}$ en una escala σ es definida como aparece en la Ecuación 2.4.

$$\mathscr{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \tag{2.4}$$

$L_{xx}(\mathbf{x}, \sigma)$ es la convolución de la segunda derivada de una Gaussiana $\frac{\partial^2}{\partial x^2} g(\sigma)$ con una imagen I en el punto $\mathbf{x}$, de forma similar para $L_{yy}(\mathbf{x}, \sigma)$ y $L_{xy}(\mathbf{x}, \sigma)$.

Como se había mencionado en *SIFT* las funciones Gaussianas son óptimas para el análisis de espacio escala, pero en la práctica estas tienen que ser discretizadas y recortadas tal como aparece en la figura 2.5.

[3]Imagen tomada de [29]

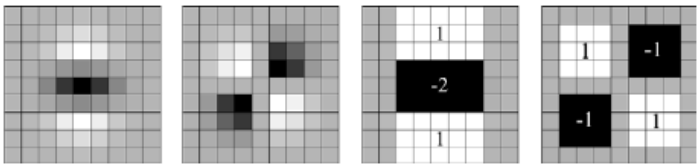


Figura 2.5: Visualización de los filtros Gaussianos de segundo orden L_{yy} y L_{xy} originales y discretizados.

Lo que hace perder repetibilidad bajo imágenes con rotaciones alrededor de múltiplos impares de $\frac{\pi}{4}$. Sin embargo, mediante el uso de las matrices Hessianas e imágenes integrales se puede realizar la siguiente aproximación al determinante alrededor de **x**.

$$det\left(\mathscr{H}_{aprox}\right) = D_{xx}D_{yy} - \left(wD_{xy}\right)^2 \tag{2.5}$$

En donde w es un factor de escala de 0.9 para darle balance al determinante Hessiano y las D son los filtros presentados en la figura 2.5 para xx, xy y yy.

Los espacios de escala son usualmente implementados como una imagen piramidal tal y como se mencionó previamente en la sección de *SIFT* las imágenes son pasadas por un filtro de suavizado Gaussiano y luego sub-muestreadas. En cambio para el caso de *SUFT* tenemos que al usar filtros de caja e imágenes integrales, no es necesario aplicar de forma iterativa el mismo filtro a la salida de la capa de filtrado previa, pero en cambio se puede aplicar un filtro de caja de cualquier tamaño con la misma velocidad directamente sobre la imagen original e incluso de forma paralela.

Denotemos que la mayor motivación para este tipo de muestreo es eficiencia computacional. Además como no se realiza un sub-muestreo en la imagen, no hay efecto de *aliasing* sobre la imagen en la pirámide, a diferencia de lo que sucede en *SIFT*.

El espacio de escala está dividido en octavas, una octava representa una serie de mapas de respuestas de filtros obtenidos por hacer la convolución a la misma imagen de entrada con un filtro con incremento de tamaño. En total, una octava abarca un factor de escala de 2 (lo cual implica que una necesita más que el doble del tamaño del filtro, filtro 1: 9x9 y filtro 2: 15x15). En la Figura 2.6[4] se presenta los tamaños laterales de los filtros para 4 diferentes octavas.

No se requiere muchas octavas ya que los puntos de interés en el espacio escala decaen rápidamente por octava. Para localizar los puntos de interés en la imagen y sobre escalas, una supresión no máxima es aplicada en el vecindario de un tamaño de 3x3x3.

Descripción de los puntos de interés y emparejamiento.

Para darle orientación a los puntos de interés se calcula la respuesta a las *wavelet* Haar en dirección x y y en un vecindario circular de radio $6s$ donde s es la escala donde el punto de interés es detectado. Se puede usar nuevamente las imágenes integrales para filtrado rápido,

[4]Imagen tomada y modificada de [29]

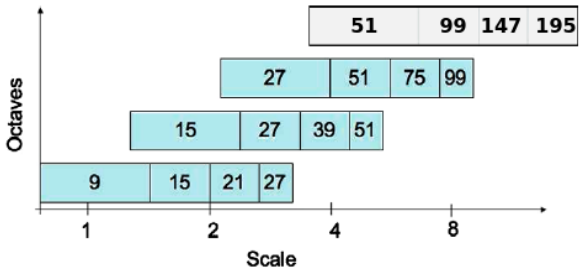


Figura 2.6: Representación gráfica de 4 niveles de octavas para el algoritmo de *SURF*

se emplean los filtros presentados en la Figura 2.7[5] y solo se requiere calcular 6 operaciones para obtener las respuestas en x o en y en cualquier escala.

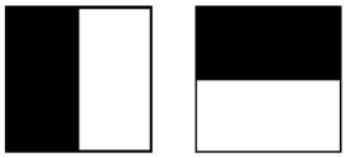


Figura 2.7: Ventanas de los filtros de Haar, como ejemplo de operadores de filtrado rápido.

Por razones de simplicidad, se denominará d_x como la respuesta de la *wavelet* Haar en sentido horizontal y d_y en sentido vertical. Para incrementar la robustez ante deformaciones geométricas y errores de localización, las repuestas d_x y d_y son primero evaluadas con una Gaussiana ($\sigma = 3{,}3s$) centrada en el punto de interés.

Luego las respuestas a *wavelet* d_x y d_y se suman sobre cada sub-región y forman el primer conjunto de entradas en el vector de características. En orden de entregar información acerca de los cambios de intensidad, también se extraen la suma absoluta de los valores de la respuesta , $|d_x|$ y $|d_y|$. Así se tiene un vector de descripción v de 4 dimensiones $v = \left(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|\right)$. La respuesta a las *wavelet* invariantes a cambios de iluminación sobre espacio, no variante al contraste se consigue ajustar el descriptor en un vector de unidad.

Los puntos de interés se encuentran en estructuras tipo gota, el signo del Laplaciano distingue gotas brillantes en fondos oscuros y en forma inversa. En la fase de emparejamiento, solo se comparan las características si estas tienen el mismo contraste, ver la Figura 2.8[6].

Por lo tanto esta información mínima permite un emparejamiento más rápido, sin reducir el desempeño del descriptor. Hay que denotar que esto es una ventaja frente a los métodos más avanzados de indexación.

[5]Imagen tomada de [29]

[6]Imagen tomada de [29]

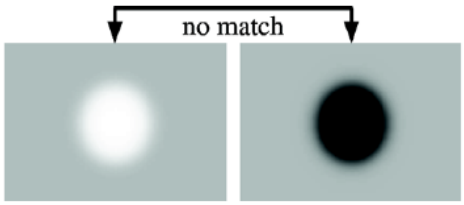


Figura 2.8: Contraste entre dos puntos de interés obtenidos con *MSER*, ejemplo de candidatos que no son correspondientes entre si.

2.4.1.3. Regiones extremas máximamente estables (*MSER*)

Encontrar correspondencia entre dos imágenes tomadas desde puntos arbitrarios, posiblemente con diferentes cámaras y con diferentes condiciones de iluminación es una dificultad y un paso crítico para realizar reconstrucción de una escena 3D, o en el caso del presente proyecto es un paso importante para construir un mosaico de imágenes.

En la mayoría de imágenes hay regiones que pueden ser detectadas con alta repetibilidad desde que sus posiciones sean distinguibles, invariantes y con características estables. En general, los datos dependientes de la forma, son llamados regiones distinguibles ***DRs*** denominado así por sus siglas en ingles (*distinguished regions*).

Se introduce un nuevo tipo de elementos en la imagen útil para la correspondencia, estas son las regiones extremas máximamente estables. Estas regiones se definen únicamente por una propiedad extrema de la función de intensidad en la región y en su límite exterior. El concepto puede explicarse de la siguiente manera informal: Imagine todo posible umbral de una imagen en escala de grises I. A continuación se hará referencia a los píxeles por debajo de un umbral como “negro” y para los mayores o iguales como “blanco”. Si se mostrará una película de imágenes umbralizadas I_t, con el cuadro t correspondientes al umbral t se vería primero una imagen en blanco. Posteriormente puntos negros que corresponden a mínimos locales de la intensidad que van a aparecer y crecer. En algún punto las regiones correspondientes a dos mínimos locales se fusionarán. Finalmente, la última imagen será de color negro. El conjunto de todos los componentes conectados de todos los cuadros de la película es el conjunto de todas las regiones máximas; las regiones mínimas podrían obtenerse a través de la inversión de la intensidad de I y ejecutar el mismo proceso. La definición formal del concepto *MSER* y las definiciones auxiliares necesarias se indican en el Cuadro 2.2.

En varias imágenes, la binarización local es estable sobre un amplio rango de umbrales en ciertas regiones. Estas regiones son de interés desde que sus posiciones sigan las siguientes propiedades:

- Invarianza a la transformación afín en la imagen de intensidad.
- Covarianza a la preservación de adyacencia.
- Estabilidad.

Imagen I es un mapeo si $I : D \subset \mathbb{Z}^2 \rightarrow \mathscr{S}$. Las regiones extremas están bien definidas si:

1. $\mathscr{S}$ es totalmente ordenada, esto es que sea reflexiva, anti-simétrica y que la relación transitiva binaria $\leq$ exista. En el caso de una imagen de grises el conjunto considerado es $\mathscr{S} = \{0,1,2\ldots,255\}$, pero las regiones extremas pueden ser definidas por ejemplo en imágenes de valor real ($\mathscr{S} = \Re$).

2. Una relación adyacente (vecindad) $A \subset D \times D$ es definida. En el caso de *MSER* una vecindad de 4 es usada. Esto significa que $p,q \in D$ son adjacentes (pAq) si y si solo si $\sum_{i=1}^{d} |p_i - q_i| \leq 1$.

Región $\mathbb{Q}$ es un subconjunto contiguo a D, esto es que por cada $p,q \in \mathbb{Q}$ hay una secuencia $p,a_1,a_2,\ldots,a_n,q$ y $pAa_1,\ldots,a_iAa_{i+1},\ldots a_nAq$.

(Externa)Región límite $\partial\mathbb{Q} = \{q \in D \setminus \mathbb{Q} : \exists p \in \mathbb{Q} : qAp\}$ esto es que el límite $\partial\mathbb{Q}$ de $\mathbb{Q}$ es el conjunto de píxeles que siendo adyacentes al menos por un píxel de $\mathbb{Q}$ pero que no pertenecen a $\mathbb{Q}$.

Región Extrema $\mathbb{Q} \subset D$ es una región tal que para todos $p \in \mathbb{Q}, q \in \partial\mathbb{Q} : I(p) > I(q)$ (Región de máxima intensidad) o $I(p) < I(q)$(Región de mínima intensidad).

Región Extrema máximamente estable (*MSER*). Sean $\mathbb{Q}_1,\ldots,\mathbb{Q}_{i-1},\mathbb{Q}_i,\ldots$una secuencia de regiones extremas anidadas, esto es que $\mathbb{Q}_i \subset \mathbb{Q}_{i+1}$. Una región extrema $\mathbb{Q}_{i*}$ es máximamente estable si y solo si $q(i) = \frac{|\mathbb{Q}_{i+\triangle} \setminus \mathbb{Q}_{i-\triangle}|}{|\mathbb{Q}_i|}$ tiene un mínimo local en $i*$(|.|denota cardinalidad). $\triangle \in \mathscr{S}$ es un parámetro del método.

Cuadro 2.2: Definiciones para *MSER*

- Detección en múltiple-escala.
- El conjunto de todas las regiones extremas puede ser enumerado en $O(nloglogn)$, donde n es el número de píxeles en la imagen.

Después del proceso de detección de *DRs* mediante el cálculo de las imágenes de intensidad (*MSER+*) y en la invertida (*MSER-*), se asocia una región de medida de un tamaño arbitrario a cada una de las *DRs,* se obtienen descriptores invariantes basados en momentos en cada una, se realiza un proceso de coincidencia robusta y se usa la medida de correlación para dar una correspondencia tentativa entre las imágenes. Finalmente, se estima la geometría epipolar (*EG*) al aplicar *RANSAC* a los centros de gravedad de las *DRs*. Este es el algoritmo sugerido en [6]. Una de las aplicaciones de este tipo de descriptores es la de obtener la linea base de estéreo.

2.4.1.4. Características desde evaluación de segmentos acelerado (*FAST*)

La detección de esquinas o de características es usado como primer paso en varias tareas de visión tales como seguimiento, *SLAM,* localización, construcción de mosaicos y reconocimiento. Es por ello que existen varios detectores de características. Sin embargo, no todos son desarrollados para aplicaciones en tiempo real, en los cuales los recursos computacionales son uno de los elementos más importantes. Además la ejecución a mayor velocidad de la extracción, se puede obtener sin necesidad de sacrificar la calidad o la detección de características.

El criterio de la evaluación de segmentos opera sobre un círculo de 16 píxeles alrededor de la esquina candidata p. El detector original clasifica p como una esquina si existe un conjunto de n píxeles contiguos en el círculo los cuales sean todos más brillantes que la intensidad del píxel candidato I_p más un umbral t, o todos mas oscuros que $I_p - t$, como se ilustra en la Figura 2.9[7].

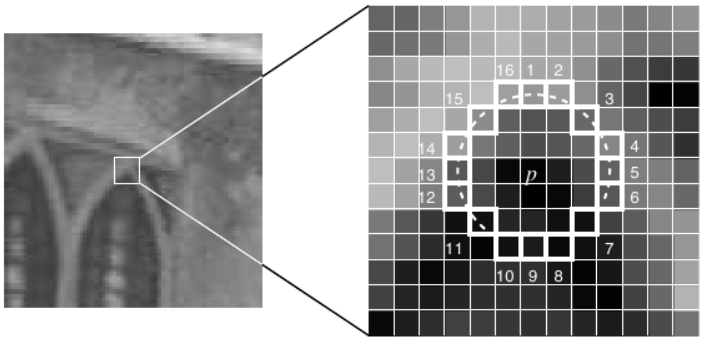


Figura 2.9: Imagen de ejemplo de detección de esquina según la iluminación de los puntos, primera fase de segmentación de *FAST*.

El valor de n es 12 porque esto admite una evaluación a alta velocidad en el cual pueden ser excluidos un gran número de no esquinas: la evaluación examina solo los cuatro píxeles el 1, 5, 9 y 13 (las cuatro esquinas cardinales). Si p es una esquina luego al menos 3 deben ser más brillantes $I_p + t$ o más oscuros $I_p - t$, si ninguna de estas condiciones se cumple entonces p no puede ser esquina. La totalidad del criterio para la evaluación del segmento, puede ser aplicada a los candidatos remanentes al examinar los píxeles en el círculo. Este detector en si exhibe alto rendimiento, pero hay varias debilidades:

1. La evaluación de alta velocidad no se generaliza bien para $n < 12$.

2. La elección y ordenamiento de la evaluación rápida de píxeles implica supuestos acerca de la distribución de las características de apariencia.

[7]Imagen tomada de [7]

3. Conocimiento de las primeras 4 evaluaciones es descartada.

4. Múltiples características son detectadas adyacentes el uno al otro.

A continuación, se presenta un método que utiliza una máquina de aprendizaje para hacer frente a los tres primeros puntos. El proceso opera en dos fases, con el objetivo de construir un detector de esquinas con un n dado.

La primera fase, las esquinas son detectadas en un conjunto de imágenes usando el criterio de evaluación de segmento para n y con un umbral conveniente. Este usa un algoritmo lento en el cual por cada píxel se evalúa todas las 16 posiciones en el círculo alrededor de este. Por cada posición en el círculo $x \in \{1 \dots 16\}$, el píxel en esa posición relativo a p se denota por $p \to x$, puede tener uno de tres estados:

$$S_{p\to x} = \begin{cases} d, & & I_{p\to x} & \leq & I_p - t & Oscuro \\ s, & I_p - t & < & I_{p\to x} & < & I_p + t & Similar \\ b, & I_p + t & \leq & I_{p\to x} & & Brillante \end{cases} \tag{2.6}$$

Eligiendo un x y calculando $S_{p\to x}$ para todos los $p \in P$ (conjunto de todos los píxeles en las imágenes de entrenamiento) en tres particiones P_d,P_s y P_b, donde cada p es asignado a $P_{S_{p\to x}}$. Sea K_p una variable que toma dos estados, verdadero si p es un borde y falso en otro caso.

La segunda fase emplea un algoritmo que empieza seleccionando el x que da la mayoría de la información acerca de si el píxel candidato es una esquina, medida por la entropía de K_p. La entropía está dada por:

$$H(P) = (c + \overline{c})\, log_2 (c + \overline{c}) - c log_2 c - \overline{c} log_2 \overline{c} \tag{2.7}$$

donde $c = \left|\left\{p \mid K_p : true\right\}\right|$, lo que implica el número de bordes y $\overline{c} = \left|\left\{p \mid K_p : false\right\}\right|$, lo que implica los que no son bordes. La elección de x se obtiene de la ganancia de información:

$$H(P) - H(P_d) - H(P_s) - H(P_b) \tag{2.8}$$

Habiendo seleccionado la x el cual se obtiene se mayor información, el proceso es aplicado recursivamente en todos los 3 subconjuntos. Y solo termina cuando la entropía de cada subconjunto se hace cero. Esto permite crear un árbol de decisión que permite clasificar todas las esquinas correctamente, visto en el conjunto de entrenamiento y por lo tanto encarna las reglas del detector de esquinas *FAST*.

Para la debilidad enunciada previamente en el 4 punto se tiene la técnica de supresión no máxima. Para ello, se tiene una función de costo, V que debe ser calculada para cada esquina detectada, y remover las esquinas que tienen una esquina adyacente con un mayor V. Hay varias definiciones intuitivas de V:

1. El máximo valor de n para el cual p todavía es una esquina.

2. El máximo valor de t para el cual p todavía es una esquina.

3. La suma absoluta de diferencias *SAD* entre los píxeles que están cercanos y el píxel central.

Sin embargo, en los casos 1 y 2 se puede tener varios píxeles que comparten el mismo valor. Por ello se elige, la opción 3 con algunas variaciones, para lograr mayor velocidad computacional.

$$V = max\left(\sum_{x\in S_{Brillante}} \left|I_{p\to x} - I_p\right| - t, \sum_{x\in S_{Oscuro}} \left|I_p - I_{p\to x}\right| - t\right) \tag{2.9}$$

Con:
$S_{Brillante} = \left\{x \mid I_{p\to x} \geq I_p + t\right\}$ y $S_{Oscuro} = \left\{x \mid I_{p\to x} \geq I_p - t\right\}$.

2.4.1.5. Características binarias básicas independientes robustas (*BRIEF*)

Los descriptores son la base de varias aplicaciones en visión por computador tales como reconocimiento de objetos, reconstrucción en 3D, localización de cámara entre otros. Dado que este tipo de aplicaciones requieren de un gran manejo de datos se requiere que los descriptores sean rápidos para calcular, para emparejar y sean eficientes en el manejo de memoria, es por ello que surge la necesidad de trabajar con descriptores cortos y binarizados para ejecutar los procesos computacionales de manera eficiente. *BRIEF* es un descriptor de palabra binaria que logra un alto grado de discriminación aun con pocos bits y puede ser calculado usando un simple test de diferencia de intensidades. Adicionalmente puede ser evaluado usando la distancia Hamming, el cual es usualmente más eficiente en comparación con el cálculo de la norma L2.

Este descriptor está inspirado en trabajos previos, que muestran que las regiones de imagen pueden ser clasificadas eficientemente con base a un número relativamente pequeño de comparaciones de intensidad. Los resultados de estos exámenes son usados para entrenar dos tipos de métodos: Árboles de clasificación o un clasificador Bayesiano Naive para reconocer las regiones vistas desde diferentes puntos de vista. En el caso de *BRIEF* se emplean ambos clasificadores y los árboles. Simplemente se crea un vector de bits de los resultados de la prueba, el cual es calculado después de tener la región de la imagen suavizada con un filtro Gaussiano cuyo tamaño es de 9x9. El filtro de mejores resultados presentado en el artículo [30] es $(X,Y) \sim$i.i.d Gaussiana $\left(0, \frac{1}{25}S^2\right)$.

Por otra parte, se define el test τ en la región **p**, del tamaño SxS como aparece en la Ecuación 2.10.

$$\tau(p;x,y) = \begin{cases} 1 & si\, p(x) < p(y) \\ 0 & otro\, caso \end{cases} \tag{2.10}$$

donde $p(x)$ es la intensidad del píxel en una versión suavizada de **p** en $x = (u,v)^T$. Eligiendo un conjunto de η_d (x,y)- pares de localización únicos, definen el conjunto de test binarios. Se toma como el descriptor *BRIEF* con un tamaño de bits de palabra η_d-dimensional (ver Ecuación 2.11).

$$f_{\eta_d} = \sum_{1 \leq i \leq \eta_d} 2^{i-1} \tau(p; x_i, y_i) \tag{2.11}$$

En el artículo [30], se presentan las pruebas correspondientes a η_d=128, 256 y 512, además se menciona que este descriptor no es apto para aplicar en imágenes con rotación. Por ello, en la ejecución de la evaluación en el artículo de presentación del descriptor, se realiza la comparación con respecto a descriptores que no tienen en cuenta la rotación como *USURF*[8].

2.4.1.6. *FAST* orientado y *BRIEF* rotado (*ORB*)

El detector de puntos clave y descriptor *SIFT* ha sido usado exitosamente por más de una década en muchas aplicaciones de descriptores visuales incluyendo reconocimiento de objetos, mosaico de imágenes, mapeo visual, etc. Sin embargo, esta impone una gran carga computacional, especialmente para aplicaciones en tiempo real tales como odometría visual, o dispositivos con baja carga como celulares.

ORB es una propuesta para reemplazar *SIFT* o *SURF* para que sea un proceso computacional más eficiente teniendo un desempeño en el emparejamiento similar a *SIFT*. Se emplea el detector de puntos *FAST* y el recientemente desarrollado descriptor *BRIEF* mencionado previamente, por ser ambas técnicas atractivas por su buen desempeño y bajo costo computacional. Un beneficio adicional es que este algoritmo es libre de restricciones de licenciamiento a diferencia de las técnicas de *SIFT* y *SURF*.

oFAST: Puntos clave *FAST* orientados.

FAST es una técnica descrita anteriormente que permite detectar esquinas, sin embargo este es un operador que no entrega información referente a la orientación, para ello usualmente se emplean técnicas como cálculo del gradiente o histogramas.

En el caso de *ORB* se usa *FAST-9*, el cual tiene buen desempeño tal y como se demuestra en [7]. *FAST* no produce medidas para los espacios sin esquinas, y se ha comprobado que tiene amplia respuesta a lo largo de los bordes, por lo tanto se emplea la medida del detector de bordes Harris[27] para ordenar los puntos clave de *FAST*.

FAST no produce características en múltiples escalas, por lo que se emplea una pirámide de escalas de la imagen tal y como se construye para *SIFT* y se producen características *FAST* (filtradas por Harris) en cada uno de los niveles de la pirámide.

El enfoque para dar orientación a *FAST* es sencillo y se denomina centroide de intensidad [34]. El centroide de intensidad asume que la intensidad de la esquina es offset desde su centro, y este vector puede ser usado para atribuir una orientación. Se define el momento de una región en la imagen como aparece en la Ecuación 2.12.

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{2.12}$$

[8] *Unrotated SURF*

Y con estos momentos se puede encontrar el centroide.

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \tag{2.13}$$

Adicionalmente se construye un vector $\vec{OC}$ desde el centro de la esquina O hasta el centroide C. La orientación de la región en la imagen es la que se presenta en la Ecuación 2.14.

$$\theta = atan2\left(m_{01}, m_{10}\right) \tag{2.14}$$

rBRIEF*: *Rotation-Aware Brief [9]

BRIEF es un reciente descriptor de características mencionado previamente que usa una evaluación simple binaria entre píxeles en una imagen suave. Su desempeño es similar a *SIFT* en muchos aspectos, incluyendo robustez a iluminación, a difuminación y distorsión de perspectiva. Sin embargo, es muy sensible con la rotación en el plano. Considere una región en la imagen suavizada p. Y un test binario τ definido por la Ecuación 2.10. La característica es definida como un vector de n test binarios definida en la Ecuación 2.11.

Varios tipos de distribución fueron considerados en el artículo original de *BRIEF*, sin embargo uno de los de mejor desempeño fue la distribución Gaussiana alrededor del centro de la región en la imagen. Adicionalmente en el caso de *ORB* se trabaja con $n = 256$.

Es importante tener suavizada la imagen antes de realizar el test. En este caso el suavizado se realiza mediante imágenes integrales, donde cada punto es una sub-ventana de 5x5 de una región de 31x31 píxeles.

Para permitir que *BRIEF* sea invariante a la rotación en el plano es dirigir *BRIEF* hacia la orientación de los puntos claves. Para cualquier conjunto de características n del test binario de la localización (x_i, y_i), se define la matriz $2xn$.

$$S = \begin{pmatrix} x_1, & \ldots, & x_n \\ y_1, & \ldots, & y_n \end{pmatrix} \tag{2.15}$$

Usando la orientación del parche θ y la correspondiente matriz de rotación R_θ, se construye una versión dirigida de S, S_θ.

$$S_\theta = R_\theta S \tag{2.16}$$

Ahora el operador *BRIEF* dirigido se convierte en:

$$g_n\left(p, \theta\right) = f_n\left(p\right) \mid \left(x_i, y_i\right) \in S_\theta \tag{2.17}$$

Se discretiza el ángulo de incremento a $\frac{2\pi}{30}$ (12 Grados), y se construye una tabla de *lookup* precalculada de los patrones de *BRIEF*. Mientras la orientación de los puntos clave θ es consistente a lo largo de las vistas, el conjunto correcto de puntos S_θ va a ser usado para calcular el descriptor.

[9] La traducción sugerida es: rotación consiente de BRIEF.

2.4.1.7. Puntos clave binarios invariantes escalables robustos (*BRISK*)

Descomponer una imagen en regiones locales de interés o características es una técnica aplicada ampliamente en la visión por computador, usada para reducir la complejidad mientras se explotan las propiedades de apariencia. El detector de puntos clave ideal encuentra regiones de la imagen sobresalientes tales que sean detectables aun al cambiar el punto de vista.

La dificultad inherente en la extracción de características está en dar balance a dos importantes objetivos: alta calidad descriptiva y bajo costo computacional requerido. Como sean mencionado en las secciones previas, algunas de las técnicas mencionadas buscan reducir el tiempo de proceso, en especial al realizar la comparación con *SIFT*, el cual es el descriptor más usado en la literatura.

BRISK se presenta en tres fases, la primera de detección de característica, la segunda de composición de descriptor y finalmente el emparejamiento de los puntos clave.

Detección de puntos clave en espacio escala.

Esencialmente *BRISK* usa *FAST*, con el objetivo de lograr no varianza en la escala, la cual es crucial para puntos clave de alta calidad, se busca la máxima no solo en los planos imagen, sino también en el espacio-escala usando la puntuación *FAST* como medida de prominencia.

En el marco *BRISK*, las capas de la pirámide en espacio-escala consisten en n octavas c_i y n intra-octavas d_i, para $i = \{0,1,\ldots,n-1\}$ y típicamente $n = 4$. Las octavas están formadas por un muestreo a la mitad progresivo de la imagen original (correspondiente a c_0). Cada intra-octava d_i es localizada entre capas c_i y c_{i+1}. La primera intra-octava d_0 es obtenida haciendo un sub-muestreo de la imagen original c_0 por un factor de 1,5, mientras el resto de las capas de intra-octavas son derivadas por un muestreo a la mitad sucesivo. Por lo tanto si t denota la escala luego $t(c_i) = 2^i$ y $t(d_i) = 2^i \cdot 1{,}5$.

Por otra parte *BRISK* maneja una máscara de 9-16, lo que requiere que al menos 9 píxeles consecutivos en el círculo de 16 píxeles sean suficientemente claros u oscuros que el píxel central para que el criterio de *FAST* sea cumplido. Inicialmente el detector FAST 9-16 es aplicado a cada octava e intra-octava separadamente usando el mismo umbral T para identificar las regiones de potencial interés. Luego, los puntos que pertenecen a estas regiones son sujetos de una supresión no máxima, en el espacio escala.

Considerando la prominencia en la imagen como una cantidad continua, no solo a través de la imagen sino también a lo largo de la dimensión escala, se desarrolla una escala de refinamiento de sub-píxel y continua para cada máximo detectado. Con el objetivo de limitar la complejidad del proceso de refinamiento, primero se adecua una función cuadrática en 2D en sentido de mínimos cuadrados por cada uno de los tres marcadores (el obtenido en la capa del punto de interés, la capa de encima y la de abajo), resultando en tres sub-píxeles refinados con la prominencia máxima. Por otra parte, para evadir el re-muestreo se considera un parche de 3x3 de resultados en cada capa. Luego estos resultados refinados son usados para obtener una función parabólica en 1D a lo largo del eje espacio produciendo la calificación final estimada y la escala estimada en su máximo. Como paso final, se vuelven a interpolar las

coordenadas de la imagen entre los parches en las capas al lado de la escala determinada.

Descriptores de puntos clave.

Dado un conjunto de puntos clave el descriptor *BRISK* está compuesto de una palabra binaria al concatenar los resultados de un test simple de comparación de brillo. El concepto del descriptor en *BRISK* hace uso de un patrón usado para muestrear la vecindad del punto clave. El patrón es ilustrado en la Figura 2.10[10], se definen N posiciones de igual separación en círculos concéntricos con el punto clave.

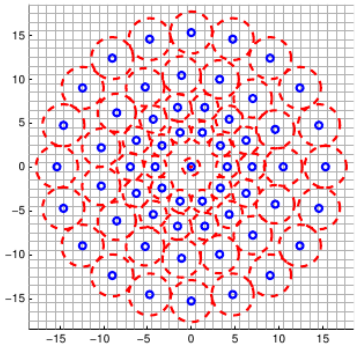


Figura 2.10: Patrón de muestreo para descriptor de *BRISK*

Para evadir los efectos de *aliasing* cuando se muestrea la intensidad en la imagen en el punto p_i en el patrón, se aplica un suavizado Gaussiano con desviación estándar σ_i proporcional a la distancia entre los puntos del respectivo círculo. Posicionamiento y escalamiento del patrón de acuerdo al punto clave k particular en la imagen, se considera uno de los $\frac{N\cdot(N-1)}{2}$ pares de puntos muestreados (p_i, p_j). Los valores de intensidad suavizados en estos puntos son $I(p_i, \sigma_i)$ y $I(p_j, \sigma_j)$ respectivamente, y son usados para estimar el gradiente $g(p_i, p_j)$ mediante la Ecuación 2.18.

$$g(p_i, p_j) = (p_j - p_i) \cdot \frac{I(p_i, \sigma_i) - I(p_j, \sigma_j)}{\|p_j - p_i\|^2} \tag{2.18}$$

Considerando un conjunto $\mathscr{A}$ de todos los pares de puntos muestreados: $\mathscr{A} = \{(p_i, p_j) \in \Re^2 x \Re^2 \mid i < N \wedge j < i \wedge i, j \in \Re\}$ se define un subconjunto de los pares de corta distancia $\mathscr{S}$ y otro subconjunto de pares de larga distancia $\mathscr{L}$.

$$\begin{array}{rcl} \mathscr{S} & = & \{(p_i, p_j) \in \mathscr{A} \mid \|p_j - p_i\| < \delta_{max}\} \subseteq \mathscr{A} \\ \mathscr{L} & = & \{(p_i, p_j) \in \mathscr{A} \mid \|p_j - p_i\| > \delta_{min}\} \subseteq \mathscr{A} \end{array} \tag{2.19}$$

Los umbrales de las distancias son $\delta_{max} = 9{,}75t$ y $\delta_{min} = 13{,}67t$ con t la escala de k. Iterando a través de los puntos pares en $\mathscr{L}$, se estima la dirección sobre la totalidad de las características del patrón del punto clave k con la Ecuación 2.20.

[10]Imagen tomada de [9]

$$g = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \frac{1}{L} \cdot \sum_{(p_i, p_j) \in \mathscr{L}} g\left(p_i, p_j\right) \tag{2.20}$$

Solo se emplean los pares de larga distancia, ya que se considera que los de corta distancia se cancelan entre ellos y no aportan información de la orientación global del punto de interés. Para la formación de un descriptor de rotación y escala normalizado, *BRISK* aplica una rotación al patrón de muestreo de $\alpha = arctan2\left(g_y, g_x\right)$ alrededor del punto clave k. El vector de bit descriptor d_k está compuesto por el resultado de comparaciones de intensidad de los puntos pares de corta distancia $\left(p_i^\alpha, p_j^\alpha\right) \in \mathscr{S}$, tales que cada bit b corresponde al resultado de la ecuación 2.21.

$$b = \begin{cases} 1, & I\left(p_i^\alpha, p_j^\alpha\right) > I\left(p_i^\alpha, \sigma_i\right) \\ 0, & otro\ caso \end{cases} \quad \forall \left(p_i^\alpha, p_j^\alpha\right) \in \mathscr{S} \tag{2.21}$$

Emparejamiento de descriptores.

Para emparejar dos descriptores *BRISK* se realiza el cálculo de la distancia *Hamming* tal y como se hace con los descriptores *BRIEF*, el número diferente de bits entre los dos descriptores es la medida de su factor de no similitud, lo que en simples palabras se convierte en una operación *XOR* seguida de un conteo de bits, lo cual puede llegar a ser muy eficiente con las arquitecturas actuales.

2.4.1.8. Puntos clave rápidos de retina (*FREAK*)

Un gran número de aplicaciones en visión confían en el emparejamiento de puntos clave a través de las imágenes. En la última década se han desarrollado varios detectores de características que buscan puntos clave que permitan ser más rápidos y más robustos. El propósito de *FREAK* es proponer un detector inspirado en la visión humana, más precisamente la retina buscando requerir menos memoria y ser más robustos que técnicas tales como *SIFT*, *SURF* o *BRISK*.

Progresos recientes en representación de imagen han mostrado que una simple comparación de intensidades en varios píxeles puede ser lo suficientemente bueno para describir y emparejar parches en imágenes. Por otra parte, se cree que la retina humana extrae detalles de las imágenes usando diferencia de Gaussianas *DoG* de varios tamaños y codifica diferencias con acciones potenciales, *FREAK* imita la misma estrategia para el descriptor de la imagen, en la Figura 2.11[11] se presenta el sistema de detección y su homología con la visión humana.

[11] Imagen tomada de [10]

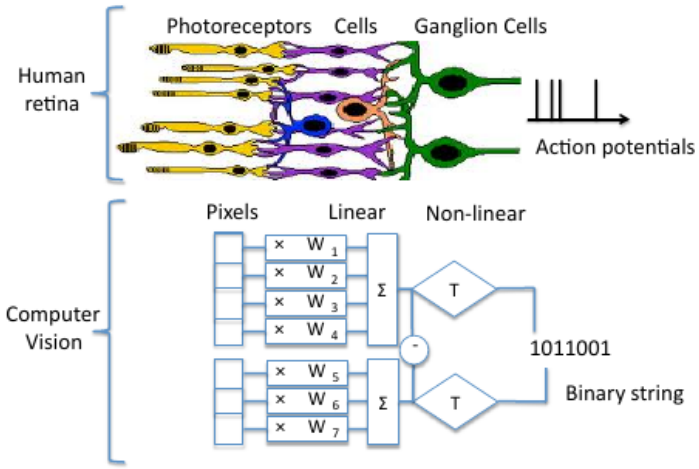


Figura 2.11: Topología del sistema *FREAK,* que imita la estrategia de visión humana.

Patrón de muestreo de retina.

Se tiene que el patrón de muestreo para *FREAK* es una grilla circular con mayor cantidad de puntos a muestrear alrededor del centro del punto de interés, a diferencia del patrón empleado en *BRISK*. Tal como se muestra en la Figura 2.12[12], es importante además suavizar cada punto muestreado para hacerlo menos sensible al ruido tal y como se planteo en *BRISK*. Cada círculo en la figura representa la desviación estándar de *kernels* Gaussianos aplicados a los correspondientes puntos de muestreo. Se ha comprobado experimentalmente que cambiar los tamaños de los *kernels* Gaussianos con respecto al patrón de retina log-polar conduce a mejor desempeño. Además adicionar redundancia entrega mayor información de potencia.

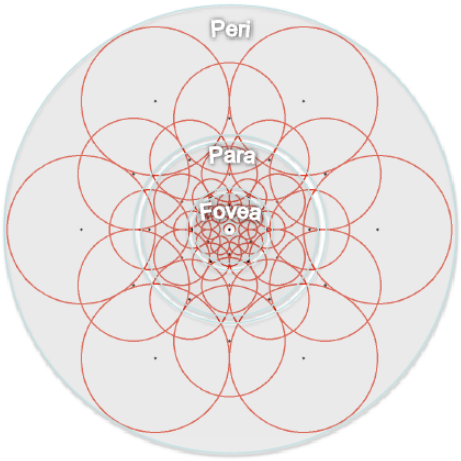


Figura 2.12: Patrón con estructura de retina para *FREAK*

[12]Imagen tomada de [10]

Se considera las intensidades I_i medidas en al campo receptivo A, B *y* C donde: $I_A > I_B, I_B > I_C$ *y* $I_A > I_C$, si los campos no se sobrelapan luego la última evaluación $I_A > I_C$ no adiciona ninguna información que aporte a la solución. En cambio, si los campos se sobrelapan parcialmente nueva información puede ser codificada.

Descriptor de lo grueso a lo fino.

Se construye el descriptor F por el umbral de diferencia entre pares de campos receptivos con su correspondiente *kernel* Gaussiano. En otras palabras, F está compuesto por una palabra binaria de secuencias de un bit de *DoG*.

$$F = \sum_{0 \leq a \leq N} 2^a T(P_a) \tag{2.22}$$

donde P_a es un par de campos receptivos, N es el tamaño deseado del descriptor y $T(P_a) = \begin{cases} 1 & si\,(I(P_a^{r1}) - I(P_a^{r2})) > 0 \\ 0 & otro\ caso \end{cases}$, con $I(P_a^{r1})$ es la intensidad suavizada del primer campo receptivo del par P_a.

Se podría llegar a construir un descriptor de gran tamaño con muy pocas muestras. Sin embargo, esto haría más difícil la tarea de emparejamiento y no todos los descriptores aportan a la solución, es por ello que se aplica un algoritmo para aprender los mejores pares de los datos de entrenamiento:

1. Se crea una matriz D de aproximadamente 50.000 puntos clave extraídos. Donde cada columna corresponde al descriptor del punto clave compuesto por todos las posibles parejas logradas con el patrón presentado en la figura 2.12.

2. Se calcula la media de cada columna, en orden de tener discriminadas las características. La alta variabilidad es deseada en este caso. Una media de 0,5 conduce a la más alta varianza de distribución binaria.

3. Se ordenan las columnas con respecto a las más altas varianzas.

4. Se mantiene la mejor columna e iterativamente se adicionan las columnas remanentes teniendo en cuenta la baja correlación con las columnas seleccionadas.

De forma interesante las primeras agrupaciones involucran los campos periféricos principales de recepción mientras las últimas implican altamente los campos centrales. Lo que implica que primero se usan los campos receptivos perifoveales para estimar la localización del objeto de interés en la imagen, y luego la validación se desarrolla con la distribución de campos receptivos más densa en el área de la fovea.

Búsqueda continua.

Los humanos no observamos la escena con estabilidad fija, sus ojos se mueven alrededor con movimientos discontinuos. Se imita esta característica dividiendo el descriptor en varias

pasos. Se comienza con la búsqueda de los 16 primeros *bytes* del descriptor *FREAK* representante de la información gruesa. Si la distancia es más pequeña que un umbral, se continua con la comparación con los siguientes *bytes* para analizar la información fina. Como resultado, las cascadas de comparaciones implementadas se desempeñan de forma más acelerada incluso con el paso de emparejamiento, más del 90 % de los candidatos son descartados con los primeros 16 *bytes*.

Orientación

Con el propósito de estimar la rotación del punto clave, se suman los gradientes locales estimados sobre los pares seleccionados. Se seleccionan los pares principales con campos receptivos simétricos con respecto al centro. Sea G el conjunto de todos los pares usados para calcular el gradiente local:

$$O = \frac{1}{M} \sum_{P_o \in G} \left(I\left(P_o^{r_1}\right) - I\left(P_o^{r_2}\right) \right) \frac{P_o^{r_1} - P_o^{r_2}}{\left\| P_o^{r_1} - P_o^{r_2} \right\|} \tag{2.23}$$

Donde M es el numero de pares en G y $P_o^{r_1}$ es el vector 2D en el espacio coordenado del centro del campo receptivo. Se seleccionan 45 pares en comparación a *BRISK* que requiere cientos.

Capítulo 3

Especificaciones

El proyecto desarrollado cumple con el propósito de construir un mapa o imagen, obtenida por medio de un mosaico de imágenes pertenecientes a una zona. La zona elegida contiene una maqueta que emula líneas de transmisión tal y como se presenta en la Figura 3.1.

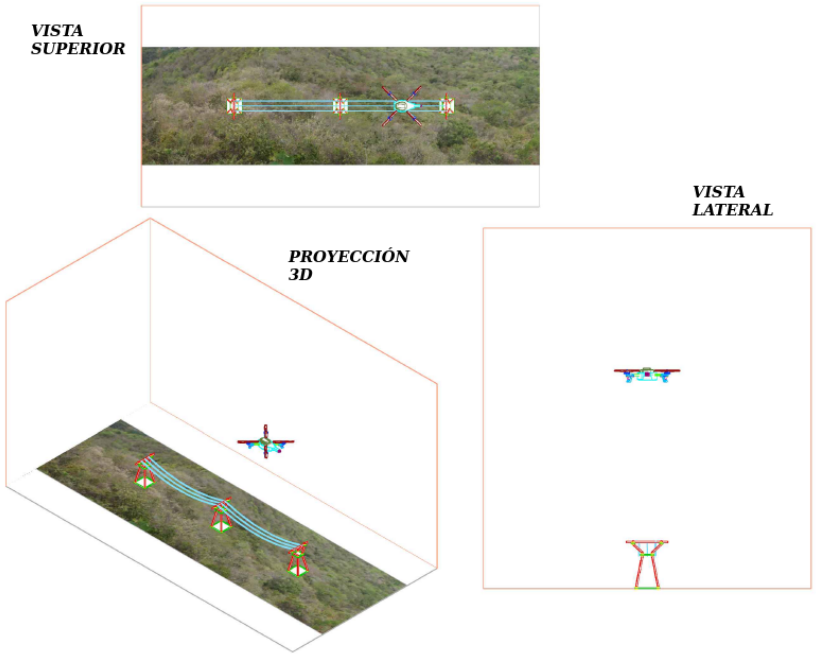
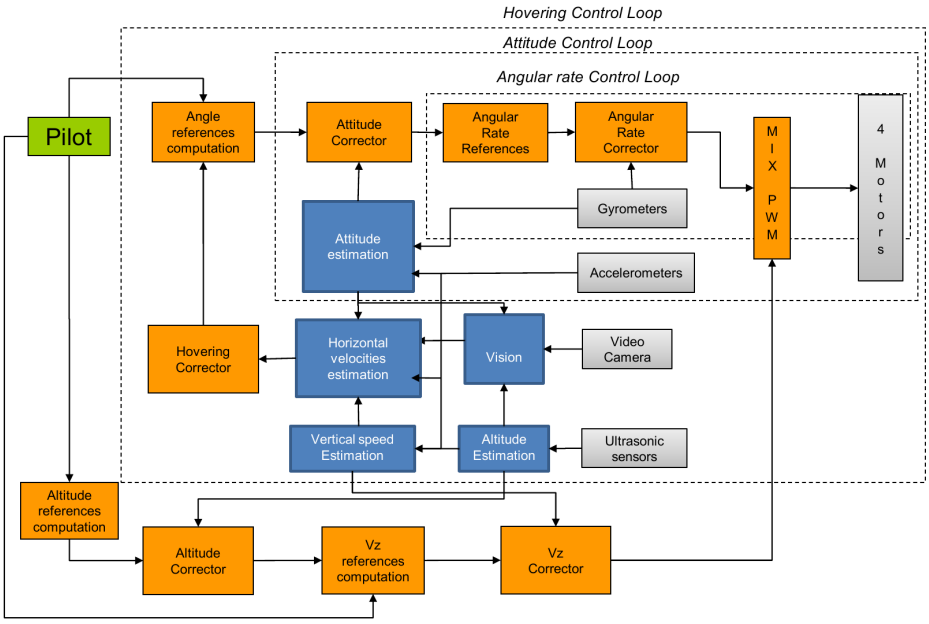


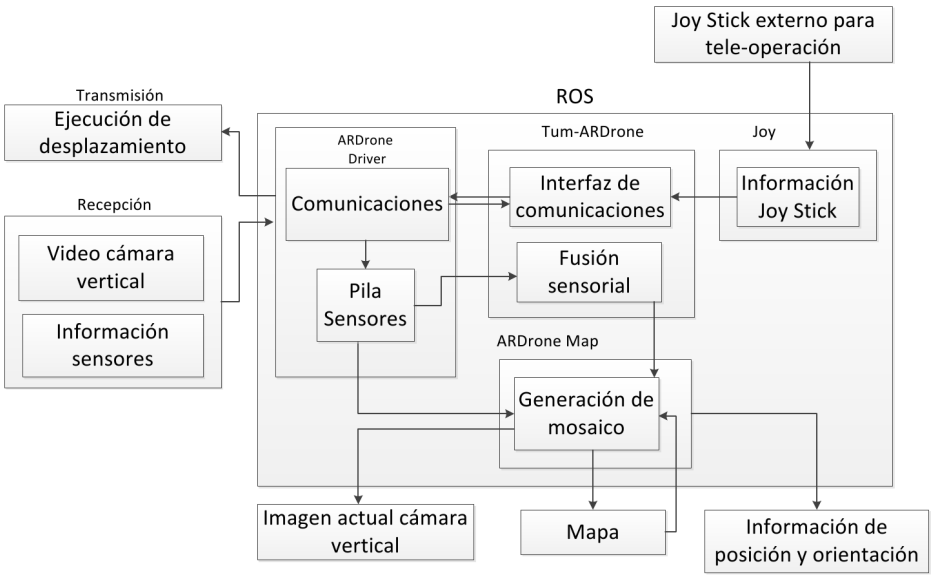
Figura 3.1: Proyecciones isométricas del escenario de pruebas con modelo en software Autocad

Para ello se dispone de un sistema que consta de un equipo portátil el cual se denomina estación base en el cual se lleva a cabo los procesos de planteamiento de trayectoria y fusión de imágenes. Además, de una plataforma aérea comercial denominada *ARDrone* 2.0, tal y como se presenta en la Figura 3.2.



(a) Arquitectura en diagrama de bloques de la plataforma comercial *ARDrone 2.0*

(b) Esquema de interacción de los equipos para el sistema de Mosaico definitivo.



(c) Diagrama de bloques general de los procesos ejecutados en la estación base, para el proceso de mosaico

Figura 3.2: Detalle del sistema de captura de información y generación de mapa en la estación base

Las imágenes 3.2a y 3.2c hacen referencia a los diagramas en bloques por cada una de las partes, entre ellos se tiene un enlace inalámbrico dedicado ilustrado en la Figura 3.2b, por el cual se reciben a la estación base los datos de los sensores embarcados en la plataforma y se envían los datos para realizar la traslación en el espacio de la plataforma.

En la Figura 3.3, se presenta la máquina de estados de los procesos generales que contemplan la interfaz con el usuario y la arquitectura interna de funcionamiento el *ARDrone* en la Figura 3.2a[1] las cuales son explicadas en detalle en [1]. Por otra parte, en la figura 3.2c se presenta un diagrama de bloques de los procesos ejecutados en la estación base durante el desplazamiento de la plataforma sobre la zona. Este desplazamiento, fue realizado de forma tele-operada, prestando mayor atención a la información capturada por la cámara para generar el mapa de la región deseada.

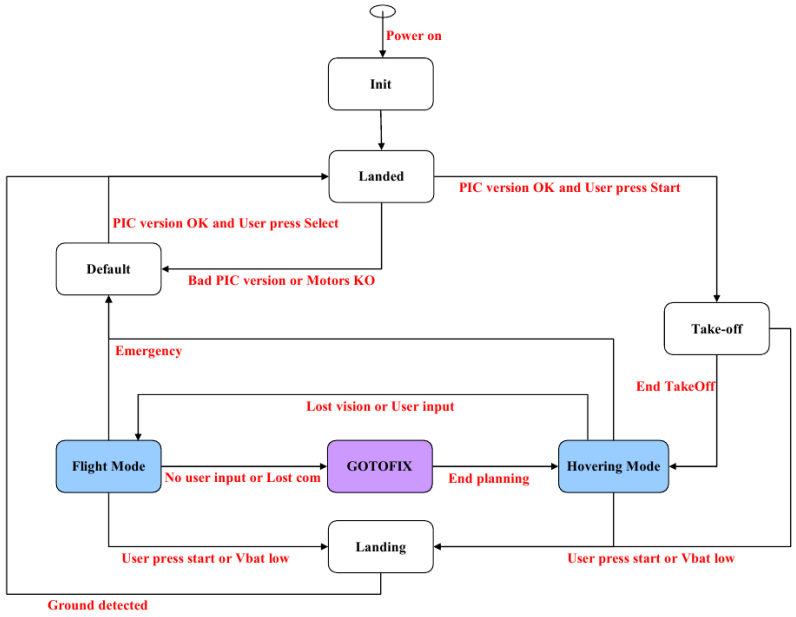


Figura 3.3: Funcionamiento del ARDrone 2.0 descrita con una máquina de estados, tomada de [1]

Por otra parte, durante el desplazamiento de la plataforma sobre la región deseada, se almacenan las imágenes capturadas, los datos sensoriales y el resultado de cada uno de los mosaicos generados por el sistema, desde los cuales se podría realizar una futura detección de fallas en las líneas de transmisión. Este escenario, fue simulado mediante la construcción

[1]Imágenes tomadas de [1]

de una maqueta, la cual se presenta en la sección 4.1.1.3. Ninguna modificación de software o hardware fue realizada sobre la plataforma aérea.

Con respecto a la plataforma, se tiene una cámara que está orientada para observar al suelo con una resolución *QVGA*[2] y una velocidad de captura de 60 fps, la cual es empleada para realizar la captura de imágenes de la zona. Se debe añadir, que la plataforma también dispone de una cámara frontal que no se empleó, con alta definición[3] de 720p y velocidad de captura de 30 fps, con un objetivo gran angular de 92°.

Por otra parte, se empleó *ROS* que es un software que provee las bibliotecas y herramientas para ayudar a los desarrolladores de software a crear aplicaciones en robótica. Este proporciona abstracción de hardware, maneja los drivers de los dispositivos propios de la plataforma, tiene visualizadores de sensores y desplazamiento de la plataforma, presenta el intercambio de mensajes, gestión de paquetes, y otras herramientas. *ROS* es distribuida bajo licencia de código abierto y fue elegido para este proyecto por todas sus características, las cuales permiten implementar un sistema versátil, de libre acceso y de fácil replicación en trabajos futuros. Permitiendo aprovechar los desarrollos previos relacionados con la plataforma comercial *ARDrone* 2.0 y poder realizar la captura de la información sensorial.

Entre los desarrollos existentes basados en *ROS*, se emplea como parte del proyecto el paquete de manejo de driver y comunicación con la plataforma aérea. Y en la tarea de simulación de escenarios es usado el paquete desarrollado para ejecución de odometría visual en entornos cerrados [35, 36]. Este algoritmo obtiene la localización de la plataforma, para ejecutar una trayectoria o navegar empleando la fusión de información de los sensores embarcados en la plataforma, cuyo código es de libre acceso en internet[4] [13, 37, 38].Una de las limitaciones para el uso de este algoritmo es que las imágenes que captura con la cámara frontal contengan información que le permita estimar su movimiento y corregir su posición, lo que implica trabajar con ambientes controlados que incluyan elementos que pueda ser usados por el software como marcadores visuales que pueden llegar a ser artificiales o propios del escenario, además aunque la plataforma disponga de dos cámaras el paquete de comunicación solo permite emplear una sola de ellas, lo que implica que este paquete no se puede emplear en simultáneo con el nodo de *ROS* desarrollado en este trabajo. Haciendo que la tarea de seguimiento de trayectoria en entornos internos no se pueda ejecutar de manera autónoma, para este proyecto la ejecución autónoma no es un objetivo.

Mientras, el *UAV* ejecuta una trayectoria se adquieren los datos de los sensores embarcados en la plataforma, en la estación base a través del nodo de comunicaciones de *ROS* con la plataforma, para con ellos, con el sistema de fusión sensorial y generador de mosaico implementado con funciones de *OpenCV* [39] y ejecutado desde *ROS,* se construya el mapa de la zona en cuestión. Al video de la cámara vertical le son extraídas imágenes que son corregidas con la información de la calibración realizada fuera de línea (parámetros intrínsecos y extrínsecos de la cámara, ver sección 4.1.1.2), en las cuales se presenta información de la zona y presentan efectos de perspectiva por la forma de navegación del *UAV* que son corregidos con la información sensorial de orientación pasada por un filtro de

[2]Resolución gráfica de 320x240 píxeles.

[3]Resolución gráfica de 1280x720 píxeles.

[4]Vinculo en la web: http://www.ros.org/wiki/tum_ardrone

Kalman [40].

Se evaluó en el sistema de mosaico diferentes detectores de puntos clave y descriptores de puntos en la imagen recientemente propuestos en el estado del arte, [6, 7, 8, 9, 10], además son evaluadas en este proyecto por la diversidad de las técnicas que usan (ver sección 2.4.1), la diversidad de manejo de información de puntos o regiones clave detectada y/o descrita de las imágenes, y por ser empleadas y desarrolladas para aplicaciones en tiempo real. Por otra parte, se realiza el contraste con otras propuestas ya usadas en la aplicación de mosaico con cámara embarcada en *UAV*, [28, 29]. Cada una de las técnicas propuestas fueron adaptadas para su uso en el algoritmo de mosaico implementado y con las características del problema.

Los parámetros elegidos para realizar la evaluación y posterior selección, fueron elegidos por la información que aportan al proceso de mosaico, comparando el desempeño de cada uno de los descriptores previamente mencionados con ejemplos de las imágenes a ser empleadas en el sistema de mosaico, además por el tiempo de ejecución del conjunto detector-descriptor y el proceso de emparejamiento, dado que se busco que el sistema de fusión se ejecutará en simultaneo con el desplazamiento de la plataforma, para mayor detalle de estos parámetros ver sección 4.3.3.1.

Adicionalmente, para la selección final del conjunto detector-descriptor se tuvo en cuenta que las técnicas que se emplearan fueran recientes en el estado del arte y fueran de libre acceso y desarrollo, lo que implica que aunque se les incluyo en el proceso de evaluación a las técnicas de *SIRF* y *SURF* no hacen parte del conjunto de detectores-descriptores a emplear en el algoritmo de mosaico implementado. Y como se menciono previamente, se usan para tener información de contraste, por ser las técnicas de *SIRF* y *SURF* las que hacen parte de las aplicaciones actuales en el proceso de mosaico con UAV.

Otros parámetros cuantitativos de evaluación del proyecto, consistieron en verificar que el mapa construido de la región en un tiempo t_1 es similar a un mapa construido de la misma zona en un tiempo t_2, lo que implica consistencia en la ejecución del sistema o repetibilidad; además, se comprueba que las características de la maqueta no han sido modificadas en el mapa capturado y que se tiene continuidad en la unión de las imágenes, lo cual puede ser evaluado verificando que el mapa construido conserva la información de una imagen global de toda la escena. A este último tipo de imágenes, se les conoce como *ground truth* y son útiles para tener la posibilidad de comparar los resultados obtenidos, comparar con trabajos futuros y caracterizar los alcances del sistema final.

Capítulo 4

Desarrollos

4.1. Herramientas para el desarrollo del proyecto

4.1.1. Herramientas *hardware.*

4.1.1.1. Características de la estación base.

En el Cuadro 4.1 se presenta el resumen técnico de las características del equipo que va a ser empleado como estación base, se menciona la información del procesador, sistemas de almacenamiento y tarjeta de comunicación inalámbrica. Medio por el cual se realiza el enlace a la plataforma aérea.

Equipo	Portátil Dell System Inspiron N411Z de 14’
CPU	Intel(R) Core(TM) i5-2450M
	@ 2501MHz de 64 Bits
	2 Cores y 4 Threads
Cache	L1 64KB
	L2 256KB
	L3 3MB
Memoria	6 GB @ 1333MHz
Disco	ATA Toshiba- 465GiB (500GB)
Alimentación	Batería de iones de litio de 6 celdas (64380mWh)
Interfaz inalámbrica	Centrino Wireless-N 1030 [Rainbow Peak]
	@ 33MHz de 64 Bits

Cuadro 4.1: Características técnicas del sistema de procesamiento de la estación base

4.1.1.2. Calibración de las cámaras de la plataforma.

Para la calibración de cada cámara de la plataforma se capturan 25 imágenes con las cuales se obtuvo un error de reproyección de 0.349323 píxeles para la cámara frontal y de

0.643273 píxeles para la cámara vertical, valores con los cuales se tiene que la información obtenida del proceso de calibración tiene un error inferior a un píxel.

Por otra parte para el proceso de calibración se empleó un patrón cuadriculado en forma de ajedrez (ver Figura 4.1) cuyas medidas físicas son conocidas y presentadas en el Cuadro 4.2. Estos valores son empleados por el algoritmo de calibración el cual se encuentra descrito en detalle en el capitulo 11 de la referencia [39].

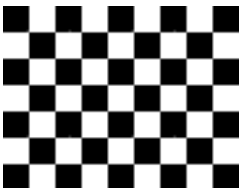


Figura 4.1: Estilo del patrón de calibración empleado para calibrar las cámaras embarcadas en la plataforma aérea.

Ancho de la cuadricula (L_x)	27 mm
Alto de la cuadricula (L_y)	27 mm
Número de esquinas internas verticales (E_y)	6
Número de esquinas internas horizontales (E_x)	8

Cuadro 4.2: Medidas físicas del patrón de calibración.

En la figura 4.2 se presenta la plataforma y el patrón empleado como material básico para realizar la calibración. Por otra parte, en la Figura 4.3 el conjunto de imágenes capturadas, para la cámara frontal de la plataforma, la cual fue capturada con la aplicación software *ARFreeFlight* del sistema operativo *Android* y en la Figura 4.4 para la cámara vertical, las cuales fueron capturadas a través de *ROS* para garantizar el tamaño original del sensor, dado que la aplicación software del *ARDrone* está diseñado para que la imagen capturada sea exclusivamente la cámara frontal. Además, el video capturado tiene un tamaño fijo y esto hace que las imágenes de la cámara vertical sean modificadas de las originales, modificando la geometría de la imagen.

Los resultados obtenidos de la calibración de las cámaras se dividen en dos grupos, los parámetros intrínsecos que están directamente relacionados con la construcción interna de la cámara los cuales se presentan en una matriz denominada intrínseca (M): $\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$ y los parámetros de distorsión propios del lente de la cámara presentados en la matriz de distorsión: $\begin{bmatrix} p_1 & p_2 & 0 \\ k_1 & k_2 & k_3 \end{bmatrix}$. En el Cuadro 4.3 se presentan los parámetros obtenidos para cada una de las cámaras.

Cámara	Frontal	Vertical
Resolución	1280x720	640x360
Error	0.349323	0.643273
c_x	635,9288	319,7204
c_y	327,0447	203,3014
f_x	1132,743	685,6904
f_y	1127,504	682,5724
k_1	-0.5373315	3.332811e-02
k_2	0.3734356	3.661670e-02
k_3	8.507365e-04	-3.940875e-04
k_4	1.246487e-03	-5.114084e-03

Cuadro 4.3: Resultados de la calibración para las cámaras embarcadas en la plataforma aérea, todos los parámetros están dados en píxeles

Figura 4.2: Escenario para la captura de imágenes y elementos empleados para la etapa de calibración.

Figura 4.3: Imágenes empleadas para realizar la calibración, tomadas con la cámara frontal de la plataforma aérea.

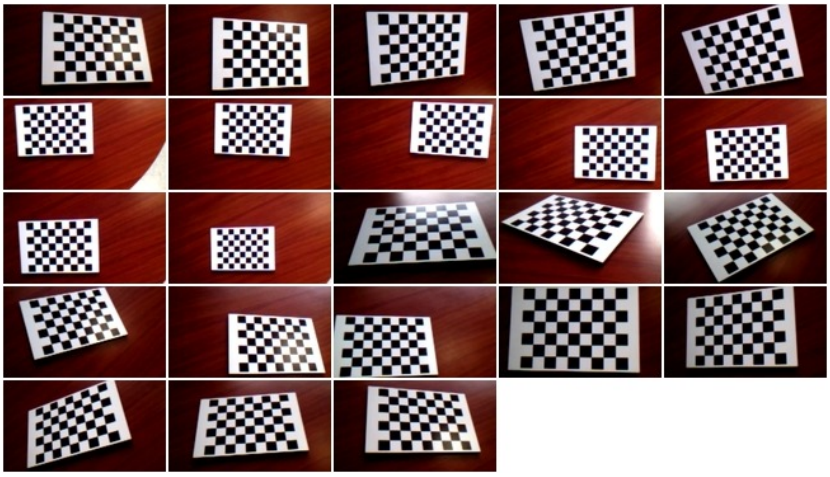


Figura 4.4: Imágenes empleadas para realizar la calibración, tomadas con la cámara vertical de la plataforma aérea.

4.1.1.3. Escenario de pruebas.

Diseño del escenario

Entre las posibles aplicaciones del vehículo aéreo no tripulado, se encuentra desplazarse sobre lineas de transmisión de energía eléctrica y capturar información de la zona, la cual podrá ser empleada para detectar problemas en los tendidos o en las estructuras. Este sistema

ha sido elegido como escenario de pruebas para la generación de mapa de superficie, para lo cual se construye una maqueta[1] que permitirá tener imágenes sintéticas de este tipo de ambientes.

En la figura 4.5[2] se presentan diferentes tipos de torres de transmisión con su altura aproximada y el voltaje de transmisión típicos según su forma.

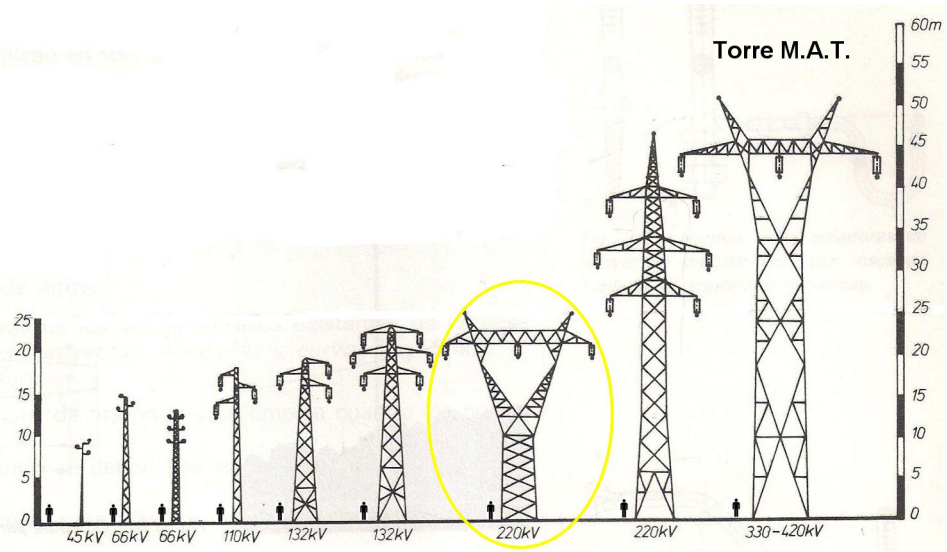


Figura 4.5: Datos de altura,capacidad y forma de las torres de transmisión eléctrica más usuales, en el círculo amarillo se encierra la estructura a diseñar para la maqueta de líneas de transmisión.

En un círculo amarillo se encierra la estructura de la torre seleccionada que fue construida para la maqueta, la cual tiene una altura aproximada de 25 metros. El diseño de la estructura es un modelo a escala, en el cual 1cm implica 0,8m. Este se presenta en las figuras 4.6 y 4.7. En la primera imagen se tiene las vistas lateral, frontal, superior y el volumen final a construir, por otra parte en la segunda imagen se presentan las dimensiones de construcción de la estructura dadas en milímetros y grados. Las barras seleccionadas para la construcción de la estructura tienen un diámetro de 3/16' y las láminas intermedias son de 2 mm de grosor. Por otra parte, el cable sin cobertura es de varios hilos y de alambre dulce de un grosor de 1/16' el cual será empleado para simular la línea de transmisión en conjunto con tres poleas colocadas en la barra superior de la estructura.

[1]Los recursos de la construcción de la correspondiente maqueta son provenientes de un proyecto de investigación, financiado por Colciencias.

[2]Imagen tomada de página web: http://www.sectorelectricidad.com/5612/tipos-de-estructuras-para-alta-media-y-baja-tension/

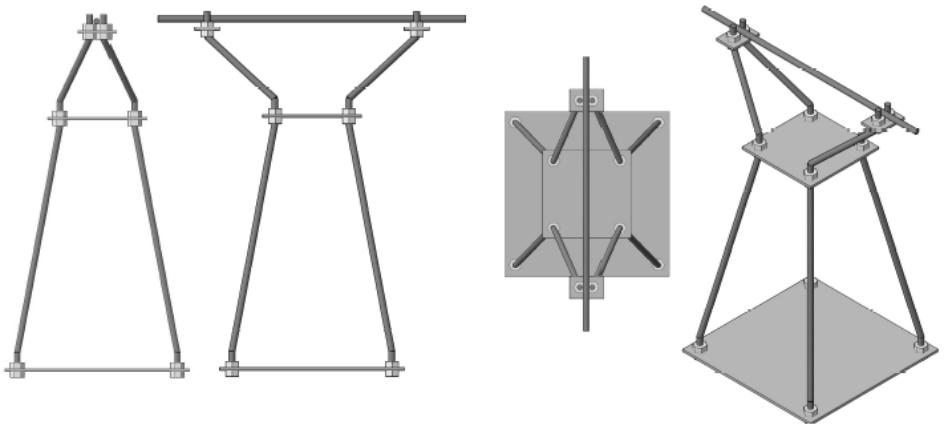


Figura 4.6: Imagen de las vistas isometricas del diseño de la torre de transmisión, a ser empleada en la maqueta.

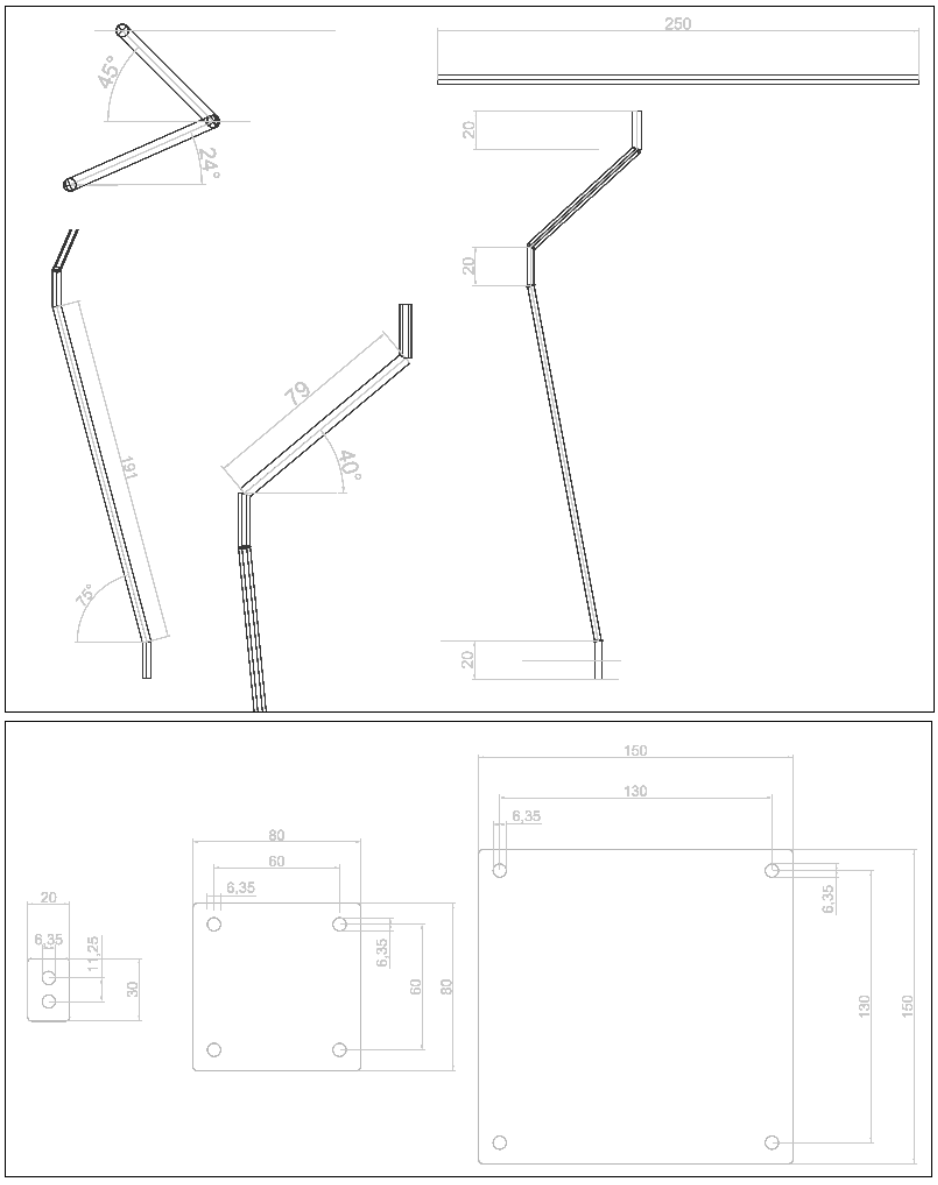


Figura 4.7: Medidas del diseño final de la torre de transmisión, parte de la maqueta de líneas eléctricas.

La maqueta contiene 3 torres de transmisión, con las cuales se puede simular diferentes orientaciones del cableado:

- Líneas rectas.
- Giro hacia la derecha.
- Giro hacia la izquierda.

Adicionalmente, para generar un ambiente de pruebas más real en entornos cerrados se empleará una imagen de fondo en la cual se tenga vegetación propia encontrada en Colombia. Se manejaron en total tres fondos impresos en un tamaño de 3 X 1 m.

Construcción del escenario real.

En la Figura 4.8 se presentan las tres torres construidas con los materiales previamente mencionados, y con las dimensiones del diseño presentado. Además en la Figura 4.10 se tiene una vista superior de la totalidad de la maqueta con tendido en línea recta con el fondo 1 (ver Figura 4.9) y en la Figura 4.12 con el fondo 2 (ver Figura 4.11) en el cual se presenta un tendido de los cables en derivación lateral emulando las lineas de transmisión. Además se cuenta con un fondo 3 presentado en la Figura 4.13.

Figura 4.8: Torres construidas para emular líneas de transmisión en la maqueta.

Por otra parte, la maqueta construida tiene la facilidad de ser tendida en ambientes exteriores, dando una longitud máxima entre torres externas de 8 metros. En el caso del ambiente interior solo tiene una longitud de 3 metros como se mencionó en la sección de diseño, distancia dada por la longitud de la imagen de fondo.

Figura 4.9: Primer imagen empleada como fondo para maqueta de líneas de transmisión

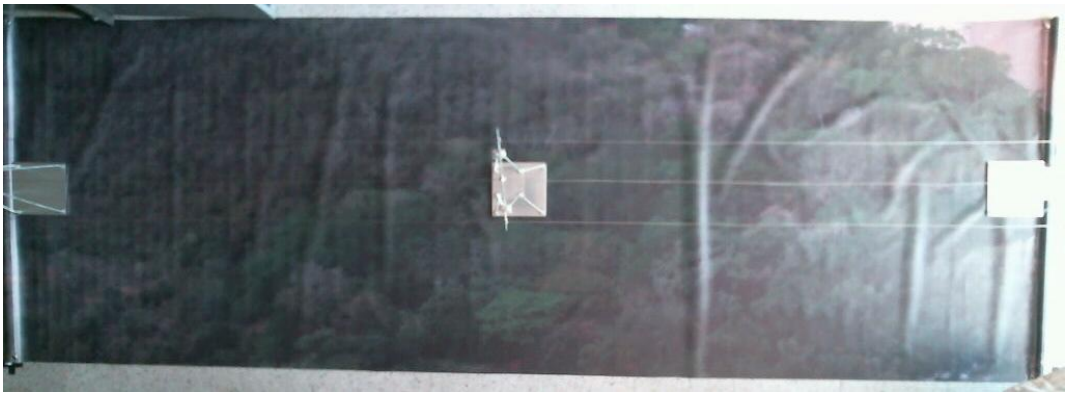


Figura 4.10: Vista superior maqueta líneas de transmisión en ambiente interior con fondo 1, presentada en la Figura 4.9.

4.1.2. Herramientas *software*.

4.1.2.1. Características de estación base.

El sistema operativo en la estación base es Linux Ubuntu 12.04 LTS, al cual se le ha realizado la instalación del siguiente software adicional para su funcionamiento en conjunto con la plataforma aérea:

- ***ROS:*** La versión Fuerte Turtle es en la cual se encuentran las diferentes aplicaciones tales como: comunicación con la plataforma y escenarios de simulación previamente desarrolladas para la plataforma *ARDrone*, es por ello que se lleva a cabo su instalación descrita en las líneas de código presentadas en la sección 7.1 el *Script* 1. Además, se incluye la instalación de paquetes adicionales relacionados directamente con la plataforma en la sección 7.1 el *Script* 2.

- ***OpenCV:*** La versión instalada es la 2.4.8, la cual contiene la librería correspondiente a la fusión de imágenes y es la última versión al momento de la realización del presente trabajo. Se requiere descargar el paquete directamente de la pagina oficial[3], se

[3]http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.8/opencv-2.4.8.zip/download

Figura 4.11: Segunda imagen empleada como fondo para maqueta de líneas de transmisión

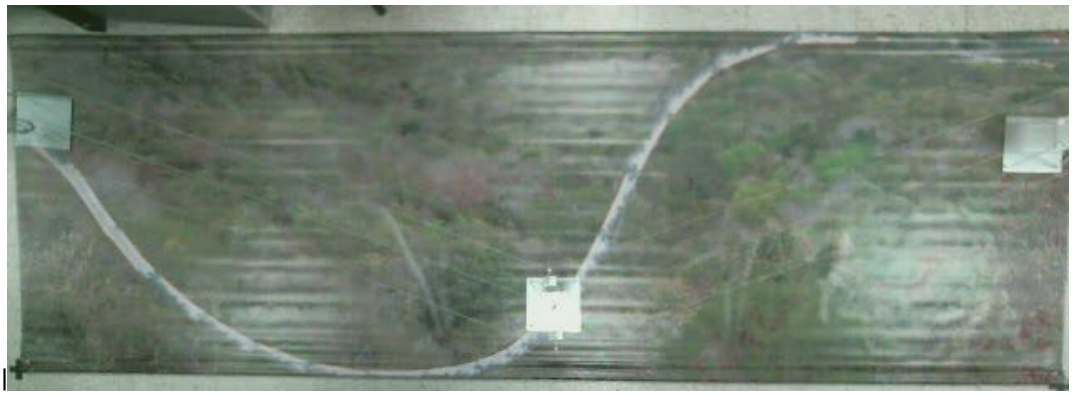


Figura 4.12: Vista superior maqueta líneas de transmisión en ambiente interior con fondo 2, presentada en la Figura 4.11.

descomprime y se ejecutan las líneas de código del *Script* 3 ubicado en la sección 7.1, teniendo en cuenta que se tengan los paquetes que se mencionan en la sección 7.1 el *Script* 4. Adicionalmente para hacer que la instalación de esta versión de *OpenCV* sea empleada por *ROS*, es necesario modificar en el archivo CMakeLists.txt el directorio de instalación por /opt/ros/fuerte, el cual actualiza los archivos de ejecución.

Después de realizar estas instalaciones se comprobó el funcionamiento individual de cada aplicación y la funcionalidad de la ejecución de un programa parte de *OpenCV* como nodo de ejecución o tarea dentro de *ROS*.

4.1.2.2. Escenarios de prueba en entorno de simulación.

Entre las etapas de desarrollo del presente proyecto, se tuvo en cuenta la fase de simulación. Para ello se desarrollaron en su totalidad dos escenarios de prueba, basados en escenarios reales. Conservando las características métricas de los escenarios y los detalles para incluir los posibles obstáculos en la navegación de la plataforma sobre la zona.

Para la construcción de los escenarios de prueba se empleó el software *open source* denominado *Gazebo*, el cual comenzó a ser desarrollado en el otoño 2002 en la Universidad

Figura 4.13: Tercera imagen de empleada como fondo para maqueta de líneas de transmisión.

del Sur de California bajo la necesidad de simular robots en ambientes externos bajo diferentes condiciones en un simulador paso a paso de alta fidelidad, es decir que fuera capaz de emular las características de los escenarios con detalle. En el 2009, John Hsu investigador de *Willow Garage* integró esta herramienta a *ROS* y desde 2011 este laboratorio empezó a proveer financiación para el desarrollo de este software. Sin embargo, desde el 2012 *OSRF(Open Source Robotics Foundation)* se ha hecho cargo del desarrollo de esta herramienta.

En la imagen 4.14 se presenta la estructura general del software, en el cual se pueden identificar la integración de programas externos de terceros los cuales aportan a diferentes fases de ejecución del sistema llamadas mediante el uso de librerías, entre estos se tiene *ODE* el cual está encargado de realizar el cálculo de la física en el entorno, *OGRE* que es el software que permite presentar en entorno con un modelo 3D mediante superficies y *QT* que se encarga de presentar el sistema mediante una interfaz gráfica [4].

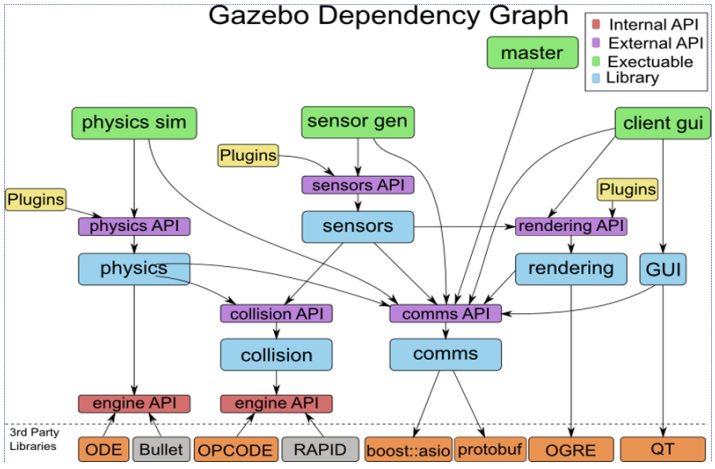


Figura 4.14: Diagrama de las dependencias del software de simulación asociado a *ROS*, denominado *Gazebo*

[4]Información tomada de la página http://www.gazebosim.org/

La versión de *Gazebo* (1.0.2) empleado en este proyecto es que viene en conjunto con la instalación de *ROS* Fuerte , la cual se localiza en el paquete denominado simulator_gazebo. Adicionalmente se tiene la existencia de otros paquetes de *ROS* Fuerte que hacen uso de *Gazebo* e integran plataformas robóticas entre ellos se encuentran:

- hector_gazebo
- rail_gazebo
- tum_ardrone

Este último, es el empleado en el presente proyecto por disponer del modelo de la plataforma aérea *ARDrone* para simulación empleando *Gazebo y ROS.*

Los entornos de simulación en *Gazebo* son archivos de texto que están descritos mediante un lenguaje *XML* que cumple con el formato *SDF* [5], se tienen dos tipos de archivo. Los archivos con extensión *.world* son los encargados de contener todos los elementos del escenario y usualmente están compuestos por el segundo tipo de archivos, los cuales tienen como extensión *.model*. Estos últimos tienen como objetivo facilitar el llamado del mismo objeto en el mismo o en otros escenarios. En el llamado de los *model* en el *world* es posible modificar su orientación (θ, φ, ψ) o posición espacial (x, y, z).

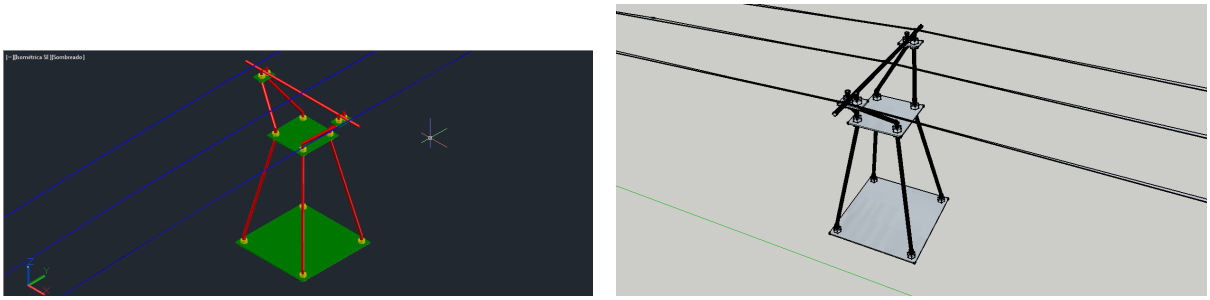
Para la creación de objetos dentro de un escenario, se dispone de dos opciones. La primera, es emplear los elementos básicos de los que dispone *Gazebo* como son una caja, una esfera o un cilindro y como segunda opción es crear grillas o *meshes*. Estas pueden ser creadas con un programa externo, teniendo en cuenta que el software solo recibe uno de estos dos tipos de extensión .dae[6] y .stl[7]. Adicionalmente hay que tener en cuenta que el entorno a visualizar en *Gazebo* maneja unidades métricas y sus coordenadas son: +Z hacia arriba, +X hacia la pantalla y +Y hacia la izquierda.

En el caso del presente proyecto se desarrolló en *Autocad* 2013 (.dwg) el modelo en 3D conservando la escala del escenario, y se exportó a un archivo .stl, el cual no contiene ninguna información de texturas o materiales y se emplea para indicar las colisiones del escenario dentro del llamado del modelo. Después se importó el archivo de *Autocad* al programa *Sketchup* en el cual se incluyen los materiales o texturas y se exportan de manera directa a la extensión .dae. Este último formato, es empleado para dar la geometría al modelo en 3D dentro del software *Gazebo*. Un ejemplo de este proceso se detalla en la figura 4.15, en la cual se presenta la torre de la maqueta previamente construida.

[5] Contenido en detalle de este lenguaje en http://gazebosim.org/sdf.html

[6] COLLADA de *collaborative design activity* es un tipo de archivo de intercambio para aplicaciones interactivas en 3D, define un estandar abierto con un esquema XML usualmente identificado con un .dae (digital asset exchange). Tomado de http://en.wikipedia.org/wiki/COLLADA

[7] STL siglas provinientes del inglés "STereo Lithography" es un formato de archivo informático de diseño asistido por computadora (CAD) que define geometría de objetos 3D, excluyendo información como color, texturas o propiedades físicas que sí incluyen otros formatos CAD. Tomado de http://es.wikipedia.org/wiki/STL.



(a) Torre desarrollada en Autocad (b) Torre con materiales en SketchUp

Figura 4.15: Ejemplo del detalle de las torres en el escenario de simulación.

Escenario de simulación escenario interno.

A continuación se presentan tres grupos de imágenes del escenario desarrollado para las pruebas en un entorno interior, en este caso se emplea como escenario el laboratorio de Robótica del Departamento de Electrónica, localizado en el 4to piso del edificio de la Facultad de Ingeniería de la Pontificia Universidad Javeriana.

1. En la Figura 4.16 se presenta la vista superior en cada uno de los programas de desarrollo (*Autocad* y *Sketchup*) y la ejecución en *Gazebo*.

2. En la Figura 4.17 se presentan las imágenes de cada programa presentado el escenario en diferentes vistas isométricas.

3. En la Figura 4.18 se presenta mayor detalle del escenario en *Gazebo*, incluyendo en la Figura 4.18c la información que ve la cámara frontal del *ARDrone* en el entorno de simulación, mediante una ventana externa.

Escenario de simulación escenario externo.

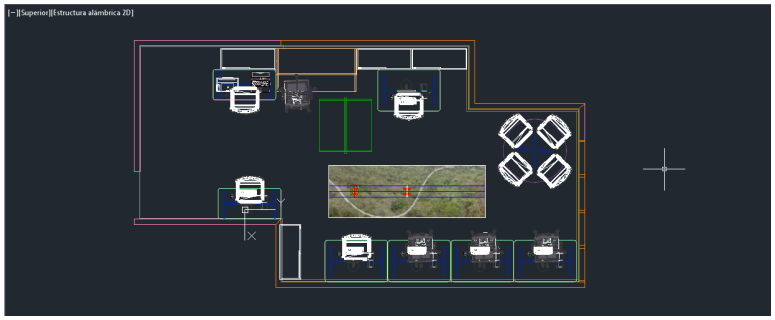
Como complemento al escenario en interiores se tiene el caso de un escenario exterior, en este caso se emplea como fondo pasto, en una superficie y las torres desarrolladas para la maqueta con una extensión entre las torres más externas de 8m. Haciendo alusión al empleo de la maqueta en la cancha de fútbol de la Pontificia Universidad Javeriana.

1. En la Figura 4.19 se presenta la vista superior en uno de los programas de desarrollo (*Autocad*) y la ejecución en *Gazebo*.

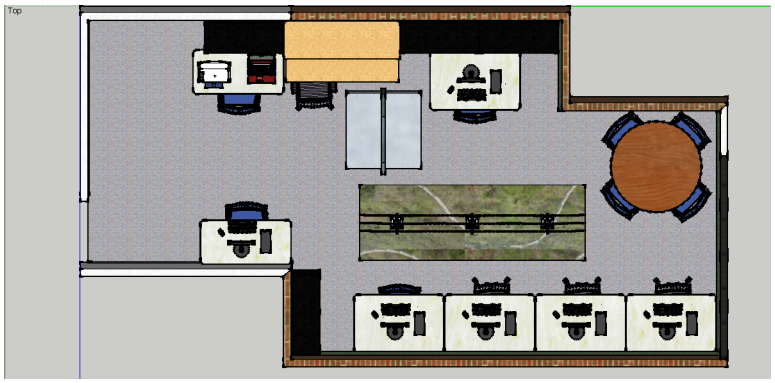
2. En la Figura 4.20 se presenta la vista lateral en uno de los programas de desarrollo (*Autocad*) y la ejecución en *Gazebo*.

3. En la Figura 4.21 se presentan las imágenes de cada programa (*Autocad* y *Sketchup*) presentado el escenario en diferentes vistas isométricas.

4. En la Figura 4.22 se presenta en mayor detalle el escenario en *Gazebo*, incluyendo la información que ve la cámara frontal del *ARDrone* en el entorno de simulación, mediante una ventana externa.



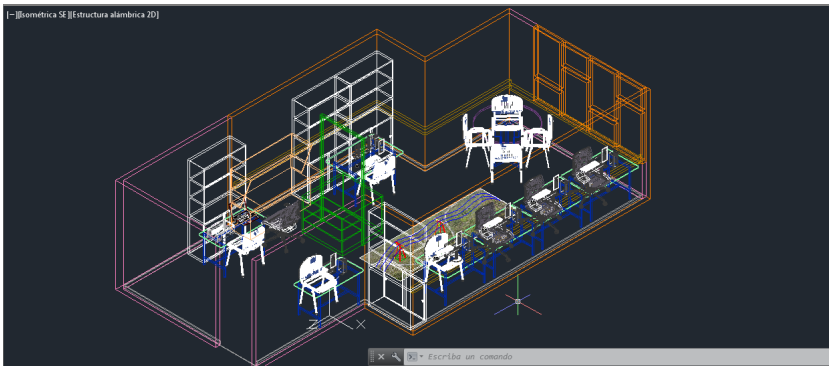
(a) Escenario desarrollado en Autocad



(b) Escenario desarrollado en SketchUp

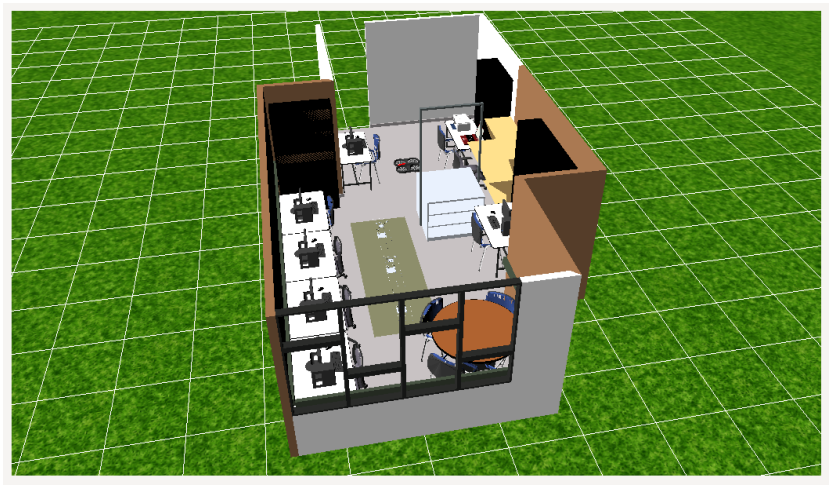
(c) Escenario ejecutado en *Gazebo*

Figura 4.16: Vista superior del escenario con cada herramienta software empleada para recrear el laboratorio de robótica, como ejemplo de un entorno interno real.



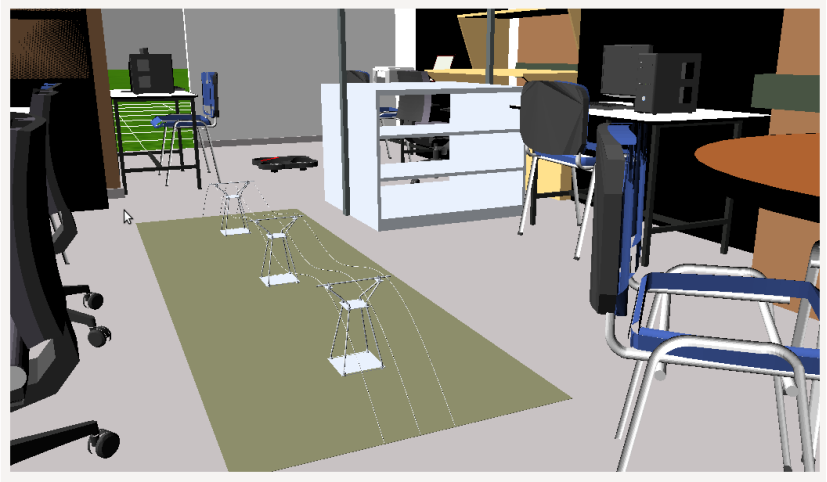
(a) Escenario desarrollado en Autocad

(b) Escenario desarrollado en SketchUp



(c) Escenario ejecutado en *Gazebo*

Figura 4.17: Diferentes vistas del escenario con cada herramienta software empleada para recrear el laboratorio de robótica, como ejemplo de un entorno interno real.



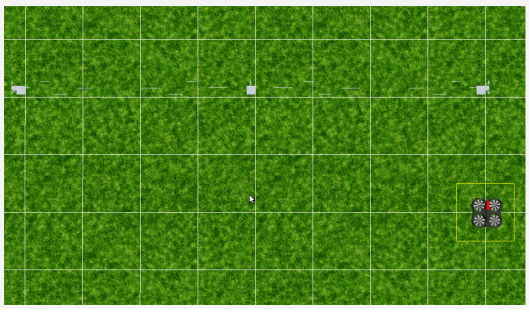
(a) Detalle vista desde la ventana

(b) Detalle visto desde la puerta

(c) Detalle información desde el ARDrone

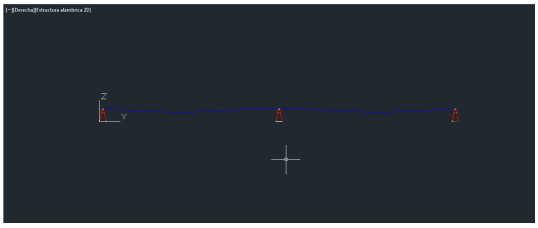
Figura 4.18: Diferentes vistas para dar mayor detalle en el escenario en un entorno interno, con la herramienta software *Gazebo*

(a) Escenario desarrollado en Autocad

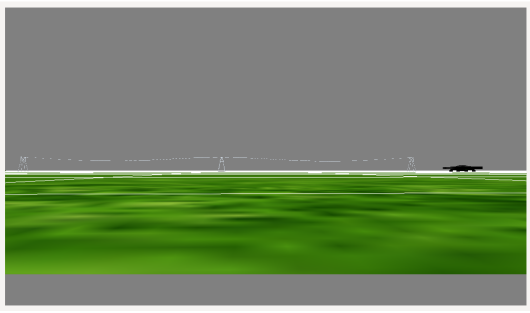


(b) Escenario ejecutado en *Gazebo*

Figura 4.19: Vista superior escenario de simulación en un entorno externo, con la maqueta de lineas de transmisión.

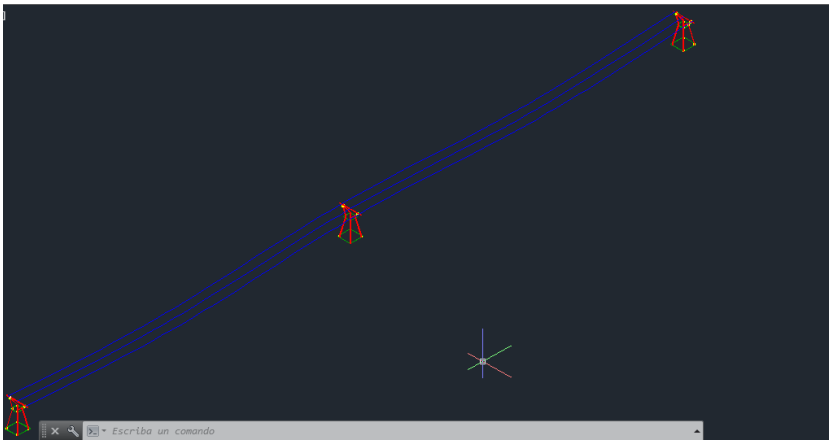


(a) Escenario desarrollado en Autocad

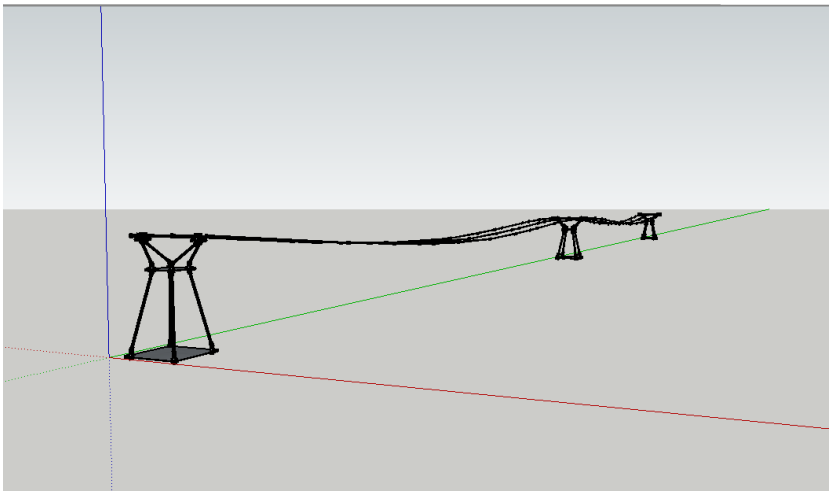


(b) Escenario ejecutado en *Gazebo*

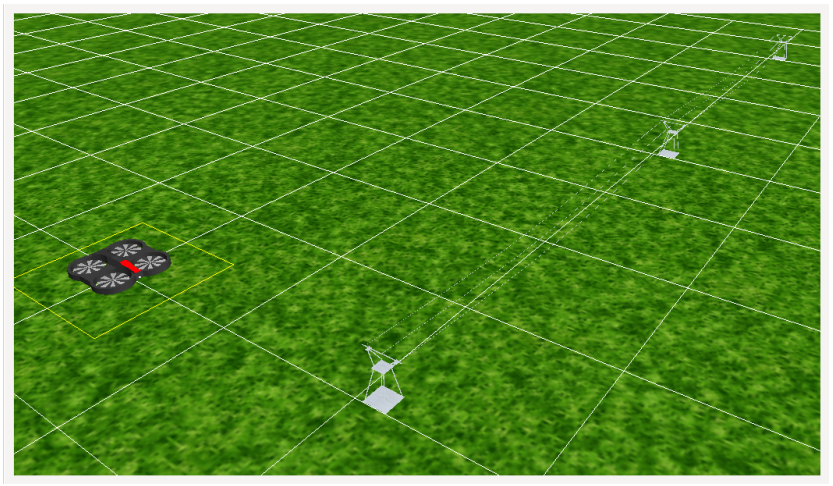
Figura 4.20: Vista lateral escenario de simulación en un entorno externo, con la maqueta de lineas de transmisión.



(a) Escenario desarrollado en Autocad



(b) Escenario en SketchUp



(c) Escenario en simulación en *Gazebo*

Figura 4.21: Diferentes vistas del escenario con cada herramienta software empleada para recrear un entorno externo.

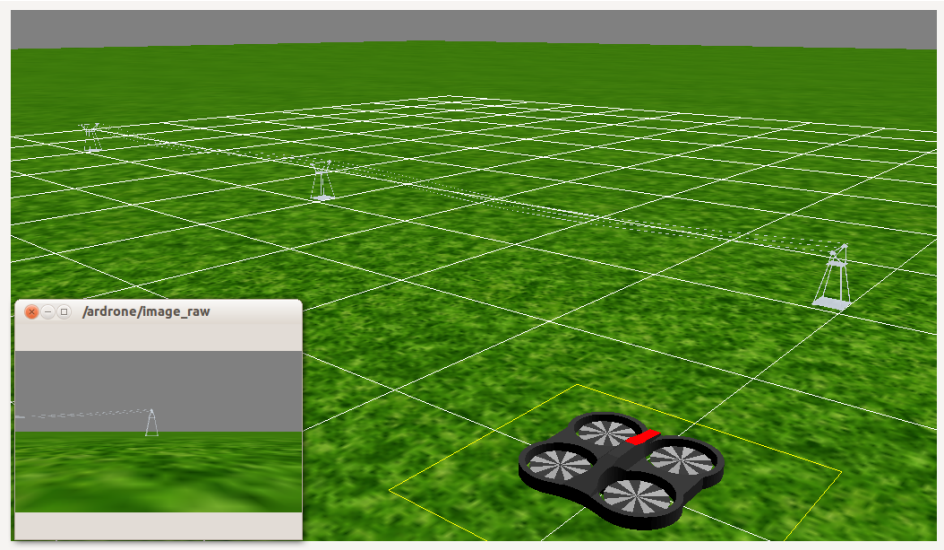


Figura 4.22: Escenario de simulación en entorno de exteriores con detalle de la información de la cámara frontal de la plataforma *ARDrone 2.0*

4.2. Generación de trayectoria

Se deben tener en cuenta dos consideraciones diferentes en la generación de una ruta a ser seguida por el quadrotor, con el objetivo de generar un mosaico a partir de las imágenes de la cámara inferior.

Primero, la altitud del quadrotor debe ser dependiente de la altura máxima de los obstáculos en el entorno de operación, esto con el fin de no chocar contra ellos, la selección de una altitud de operación, también fija la resolución espacial capturada por la cámara, como se muestra en la Figura 4.23, mayor altitud tiene como resultado una mayor área de cobertura y una menor relación de píxeles sobre metro. El área observada por la cámara inferior A está dada por la Ecuación 4.1 y depende de la altitud de la plataforma l y de las aperturas del lente de la cámara, dadas por los ángulos θ y ϕ. En el Cuadro 4.4 se presentan los valores obtenidos de los ángulos de apertura.

$$A = W_\theta W_\phi = 2ltan\left(\frac{\theta}{2}\right)2ltan\left(\frac{\phi}{2}\right) = 4l^2 tan\left(\frac{\theta}{2}\right)tan\left(\frac{\phi}{2}\right) \tag{4.1}$$

θ	49,08
ϕ	28,64

Cuadro 4.4: Valores de ángulos de apertura cámara inferior *ARDrone 2.0*

Una segunda consideración involucra el comportamiento en tiempos de operación del hardware de captura y del algoritmo de creación del mosaico. Estos tiempos limitan la

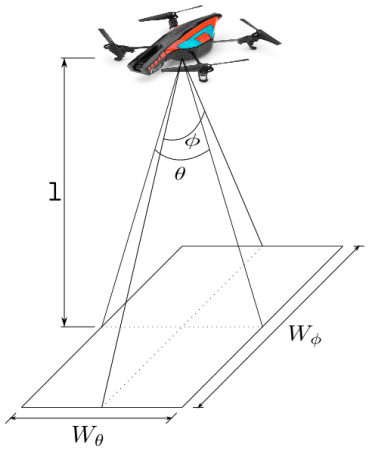


Figura 4.23: Relación entre altitud del quadrotor, apertura del lente de la cámara y área observada por la cámara inferior de la plataforma.

velocidad máxima a la cual puede trasladarse el quadrotor. Se define el tiempo de captura de la imagen y envío a la plataforma como t_c, además el tiempo de procesamiento de la imagen capturada para la creación del mosaico t_m. Si se toma en consideración las dos longitudes del entorno vistas por la cámara, $\mathbf{W} = \left(W_\theta, W_\phi\right)^T$ es posible encontrar una velocidad máxima a la cual se puede trasladar la plataforma, dada por:

$$\mathbf{v_d} = \left(\frac{\rho}{t_m + t_c}\right)\mathbf{W} \tag{4.2}$$

donde ρ es una constante, $0 < \rho < 1$, y se selecciona de acuerdo a la cantidad de características de la superficie del terreno, visualmente esta constante permite fijar el porcentaje de ocupación de la imagen anterior con la imagen siguiente en $(1-\rho)$, como puede verse en la figura 4.24,

En la implementación práctica la altitud es considerada constante y es controlada por la misma plataforma. La velocidad máxima a la cual se puede desplazar el quadrotor se limita mayormente por el tiempo de creación del mosaico t_m, un análisis de estos tiempos puede verse en el diagrama de caja en la Figura 5.1, la velocidad máxima de desplazamiento se fijó de manera manual siguiendo las ecuaciones 4.1 y 4.2, en general cualquier trayectoria que cumpla 4.2 puede cumplir el objetivo de generar el mosaico de la zona, esto ya que las imágenes repetidas de un mismo sitio, son evitadas en la generación de un mosaico de imágenes redundantes.

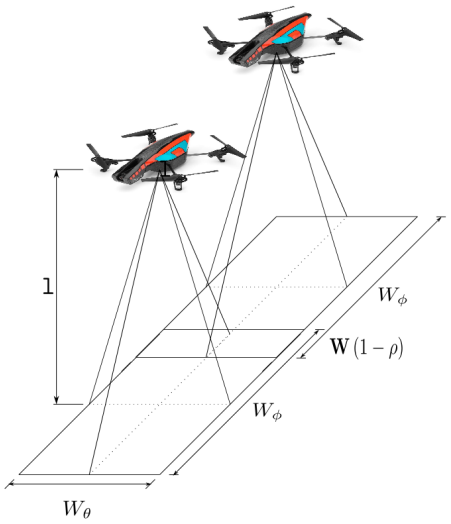


Figura 4.24: Áreas comunes de ocupación dependientes de la velocidad, en los tiempo t y $t+1$, teniendo una altitud constante de desplazamiento

4.3. Generación de Mapas

4.3.1. Descripción teórica del algoritmo de Mosaico.

El algoritmo de mosaico de imágenes o generación de panorama o *Stitching*[8] está compuesto por dos grandes bloques presentados en la Figura 4.25 donde se indica la conexión entre ellos y sus entradas. La entrada al sistema de mosaico son imágenes, como mínimo se requieren dos de ellas y son usadas por ambos bloques, el de registro y de composición para tener a la salida una sola imagen compuesta por la unión de las imágenes de entrada.

A continuación se presenta el funcionamiento de cada una de estas secciones, como complemento de la sección 2.4.

4.3.1.1. Registro.

Esta fase dentro del proceso de mosaico también es usada para otras aplicaciones en visión artificial como es la estabilización de video [41], el reconocimiento de objetos, la reconstrucción de una escena 3D, *SLAM*, odometría visual, entre otros. Este proceso es el que permite identificar los elementos semejantes entre imágenes para posteriormente realizar el proceso de unión entre ellas. En la Figura 4.26 se presenta un diagrama de bloques que resume este proceso.

Inicialmente se requiere alinear las imágenes, para ello es necesario extraer características. La primera opción es emplear un método de alineación directa, el cual consiste en extraer parámetros globales de las imágenes y posteriormente buscar la relación

[8]Traducción: Puntadas

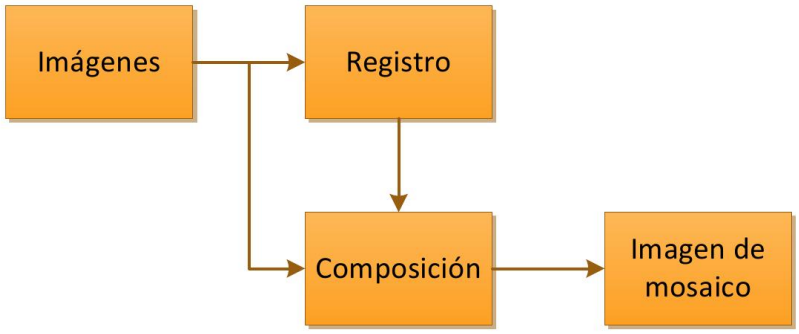


Figura 4.25: Diagrama de bloques del proceso general de *Stitching*

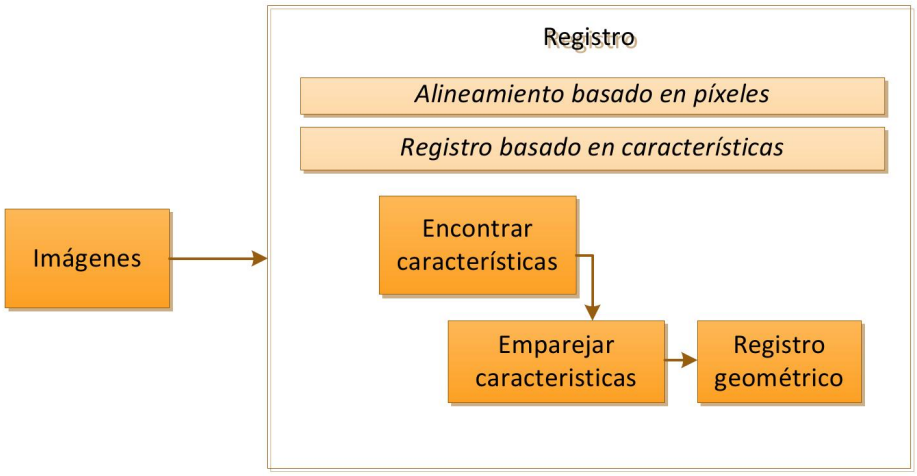


Figura 4.26: Diagrama de la sección de registro como primera fase del algoritmo de Mosaico

entre ellas mediante métricas de error o emparejamiento, entre estos métodos se tienen los histogramas, las proyecciones integrales [41], alineación basada en coeficientes de *Fourier*, entre otros. Y entre las métricas de error se suele usar *SSD*, *SAD* o correlación entre imágenes. Sin embargo, este tipo de características suelen presentar problemas ante oclusiones de la escena.

Por otra parte, se tiene la alineación basada en características locales. Esta técnica es la que se eligió para el presente proyecto, por presentar ventajas en el tiempo ejecución, definición de las características, constante uso en aplicaciones de ejecución en tiempo real y resultados confiables. Las características locales idealmente deben cumplir con las siguientes propiedades, la repetibilidad entre imágenes, la distinción entre la misma imagen, la localización única y local, la suficiente cantidad de descriptores para detectar elementos pequeños, la exactitud sin ser afectada por la forma o la escala, la eficiente detección, la detección ante deformaciones y ser robustas ante cambios de iluminación. Sin embargo, el cumplimiento de todos estos elementos hace que esta etapa de proceso requiera de mayor número de recursos, lo que implica que dependiendo de la aplicación se suele prestar a

algunas propiedades más atención que a otras.

Para detectar este tipo de características existen tres grandes grupos de algoritmos, las basadas en bordes, las basadas en puntos de interés o esquinas y las basadas en bloques o regiones de interés. Además se tienen los algoritmos que se encargan de realizar una descripción de estas características entre los más usuales y clásicos en el estado del arte se encuentran *SIFT* y *SURF*. A continuación se presenta en el Cuadro 4.5, los detectores (*Feature Detector*) y los descriptores de características (*Descriptor Extractor*) que se evaluaron en el presenten proyecto y de los cuales se discutió en mayor detalle en la sección 2.4, además son sujetos a una evaluación, selección y adecuación para ser usados en el algoritmo definitivo de mosaico en la sección 4.3.3.

Detector	**Descriptor**
SIFT [28]	*SIFT [28]*
SURF [29]	*SURF [29]*
ORB [8]	*ORB [8]*
MSER [6]	*BRISK [9]*
FAST [7]	*BRIEF [30]*
	FREAK [10]

Cuadro 4.5: Detectores/descriptores elegidos para ser evaluados en el presente proyecto

Posterior a la fase de detección y descripción de los puntos clave, se tiene la de emparejamiento (*Matching*) en esta fase se pueden usar las mismas técnicas que se mencionaron para la de alineación de las características globales, como son *SSD*, *SAD* o correlación. Adicionalmente, se pueden usar otras como son las de cercanía ya sea geométrica en distancia como L1 o L2 o la de cercanía por cambio en un dato binario como es la distancia Hamming, dependiendo de la forma de la construcción del descriptor. Otra técnica es emplear herramientas de clasificación no paramétrica como es *KNN*, esta última técnica es usada a través de un algoritmo optimizado para realizar el computo a mayor velocidad denominada *Flann* [42, 43][9]. Como complemento a este emparejamiento es usual emplear *RANSAC*, el cual permite determinar cuales de los puntos de emparejamiento si son parte de la solución o si corresponden y descartar a los posibles puntos que no lo son, que hacen que la alineación no sea exacta.

Como elemento final del registro se tiene el registro geométrico, que es la estimación de los parámetros de movimiento, lo cual hace que dos imágenes se relacionen. Existen varios modelos de movimiento los cuales se resumen en el Cuadro 4.6 y son ejemplificados en la imagen 4.27[10]. Se tiene que cada una de estas transformaciones tienen un número de *DOF,* este dato hace referencia al tipo de movimiento que puede ejecutar la cámara desde la cual se realiza la captura de las imágenes, por ejemplo la transformación de traslación solo presenta dos grados de libertad los cuales se relacionan directamente con la traslación en los ejes x y y, lo que implica que la información en la imagen no presenta variaciones en su forma

[9]Página web oficial: http://www.cs.ubc.ca/research/flann/

[10]Imagen tomada de [22]

geométrica y la proyección de la información de la imagen en un tiempo t_1 a un tiempo t_0 se puede realizar aplicando la matriz correspondiente. En comparación la transformación de homografía, la cual es empleada en este proyecto, tiene un total de 8 grados de libertad, lo que implica que esta matriz de transformación incluye la información de la rotación en cada uno de los ejes ($\theta, \phi\, y\, \psi$) y la traslación en el espacio (x, y y z), proyectadas en el espacio 2D de la imagen.

Las transformaciones de traslación, similaridad y afín tienen una relación lineal entre el movimiento y los parámetros desconocidos de las matrices presentadas en el Cuadro 4.6, en esos casos la estimación de los parámetros desconocidos se pueden obtener mediante una regresión lineal.

Transformación	Matriz	*DOF*
Traslación	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$	2
Rigida/Euclideana	$\begin{bmatrix} c_\theta & -s_\theta & t_x \\ s_\theta & c_\theta & t_y \end{bmatrix}$	3
Similaridad	$\begin{bmatrix} 1+a & -b & t_x \\ b & 1+a & t_y \end{bmatrix}$	4
Afin	$\begin{bmatrix} 1+a_{00} & a_{01} & t_x \\ a_{10} & 1+a_{11} & t_y \end{bmatrix}$	6
Proyectiva/perspectiva	$\begin{bmatrix} 1+h_{00} & h_{01} & h_{02} \\ h_{10} & 1+h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix}$	8

Cuadro 4.6: Tabla resumen de las transformaciones de movimiento entre imágenes y sus grados de libertad, en la proyección geométrica de la imagen.

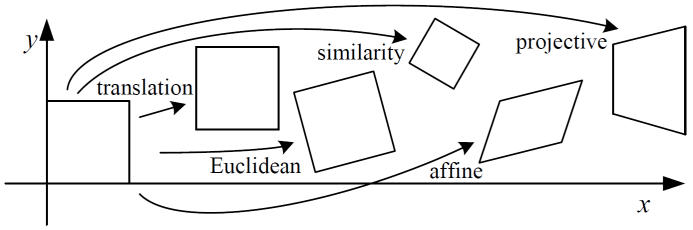


Figura 4.27: Transformación gráfica de las transformaciones presentadas en el Cuadro 4.6

Sin embargo, para el caso del presente proyecto estas transformaciones no son suficientes para lograr un registro entre las imágenes debido al movimiento de la plataforma Robótica. La transformación más adecuada es la de proyección o de perspectiva, cuya matriz es conocida como de homografía (H). La estimación de esta matriz se puede llevar a cabo mediante un proceso de aproximación Gauss-Newton o empleando el conocimiento del movimiento ejecutado, en el caso del presente proyecto se emplea la información filtrada de la *IMU* y se

estima el movimiento entre imágenes. Para ello se tiene que la matriz de homografía H se puede definir como aparece en la Ecuación 4.4 según [44, 45], teniendo en cuenta que:

- El vector unitario $\widehat{k}$ está definido como $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$.
- El vector p_n es la posición del marco de navegación en coordenadas cartesianas $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$.
- Se tiene que t es el tiempo actual y t_0 es el tiempo previo.
- El subíndice en la matriz de rotación $R_{n/b}$ hace referencia a la matriz que describe el movimiento desde el marco de referencia inicial hasta el marco de referencia de navegación. Además se tiene que cumple con la relación $R_{b/n}la = R_{n/b}^T$, y se define en la Ecuación 4.3[11].

$$R_{n/b} = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{4.3}$$

$$H = R + \frac{1}{d} T N^T \tag{4.4}$$

se define que R es la matriz de rotación definida como aparece en la Ecuación 4.5. Además d es la distancia hasta la plataforma desde el suelo y está definida en la Ecuación 4.6 considerando que la superficie de captura de la imagen es plana; T es la traslación incluyendo la información de rotación definida en la Ecuación 4.7 y en la Ecuación 4.8 se define N como el vector normal al plano de captura.

$$R = R_{b/n}(t) R_{n/b}(t_0) \tag{4.5}$$

$$d = -\widehat{k}^T p_n(t_0) \tag{4.6}$$

$$T = R_{b/n}(t) [p_n(t_0) - p_n(t)] \tag{4.7}$$

$$N = R_{b/n}(t_0) \widehat{k} = \begin{bmatrix} -s_\theta \\ s_\phi c_\theta \\ c_\phi c_\theta \end{bmatrix} (t_0) \tag{4.8}$$

Para la correcta implementación en el algoritmo final de mosaico, es necesario pasar la información de homografía H a píxeles para ello se requiere aplicar la matriz intrínseca K obtenida en el proceso de calibración previamente mencionado en la sección 4.1.1.2, tal

[11] Para mayor comodidad se define $cos(\beta) = c_\beta$ y $sen(\beta) = s_\beta$

como se indica en [22, 45], dando como resultado la matriz de homografía G, definida en la Ecuación 4.9.

$$G = KHK^{-1} \tag{4.9}$$

Después de conocer la matriz de homografía, se realiza un proceso de actualización de distancias focales y matrices de rotación que permite generan la corrección de la transformación de perspectiva [22].

4.3.1.2. Composición.

En esta segunda fase del proceso de mosaico, se tienen como entradas: las imágenes que entran al algoritmo de mosaico sin ningún tipo de alteración y la información obtenida en la fase de registro, como es la información de correspondencia y la información de las cámaras por cada imagen. En la Figura 4.28 se presenta un resumen de los elementos más importantes de esta sección.

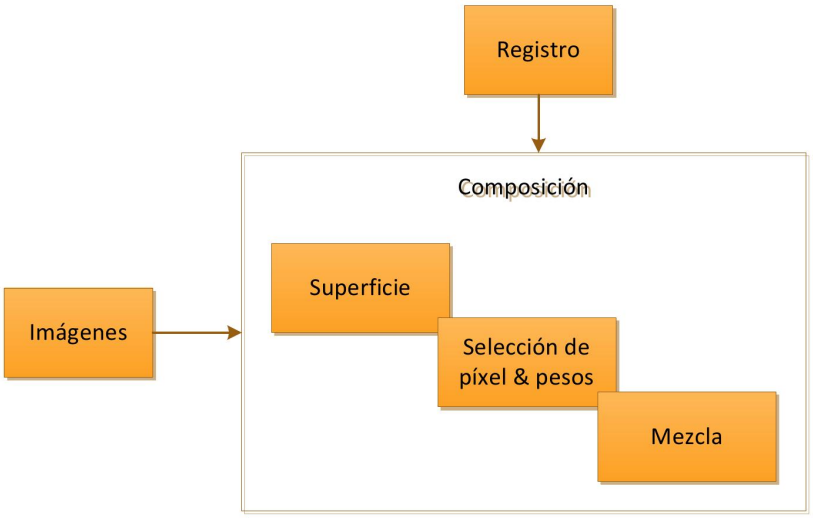


Figura 4.28: Diagrama de la sección de composición como segunda etapa en el algoritmo de Mosaico

Como primer elemento está la proyección de las imágenes en una superficie. Del proceso previo, tenemos cuál es la relación entre las imágenes partiendo de las características dentro de la imagen y el movimiento que transforma el contenido de la imagen 1 a la 2. Sin embargo, no se tuvo en cuenta qué tipo de forma geométrica está describiendo la cámara que captura, se tienen típicamente dos casos la cilíndrica y la esférica. Sin embargo en el caso del presente proyecto, no se describe ninguna de estas dos formas, ya que el desplazamiento es lineal sobre la zona deseada, lo que implica una región plana.

Para este tipo de curvatura se tiene que realizar inicialmente una proyección regida por la Ecuación 4.10, de cada una de las esquinas de la imagen indicadas en la ecuación con x_i

y y_i con $i = 0,\ldots 3$, en la que R_c es la rotación de la cámara y K es la matriz de parámetros intrínsecos.

$$\begin{bmatrix} x_{ni} \\ y_{ni} \\ z_{ni} \end{bmatrix} = \left(R_c K^{-1}\right) \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{4.10}$$

Posteriormente, se realiza una normalización para tomar los valores máximos y mínimos de los resultados en cada uno de los ejes como se presenta en la Ecuación 4.11. Y se invierte la proyección como se presenta en la Ecuación 4.12 normalizada en 2D como se realizó en el proceso de proyección en $x = \frac{x_{rj}}{z_{rj}}$ y en $y = \frac{y_{rj}}{z_{rj}}$, sobre la región resultante entre la resta de los valores máximo y mínimo de la Ecuación 4.11 de los ejes x y y.

$$min, máx\left(\frac{x_{n0}}{z_{n0}} \ldots \frac{x_{n3}}{z_{n3}}\right), min, máx\left(\frac{y_{n0}}{z_{n0}} \ldots \frac{y_{n3}}{z_{n3}}\right) \tag{4.11}$$

$$\begin{bmatrix} x_{rj} \\ y_{rj} \\ z_{rj} \end{bmatrix} = \left(K R_c^{-1}\right) \begin{bmatrix} u_j \\ v_j \\ 1 \end{bmatrix} \tag{4.12}$$

en la cual u_j está definido desde el $min\left(\frac{x_{n0}}{z_{n0}} \ldots \frac{x_{n3}}{z_{n3}}\right)$, hasta el $máx\left(\frac{x_{n0}}{z_{n0}} \ldots \frac{x_{n3}}{z_{n3}}\right)$ y v_j está definido desde el $min\left(\frac{y_{n0}}{z_{n0}} \ldots \frac{y_{n3}}{z_{n3}}\right)$, hasta el $máx\left(\frac{y_{n0}}{z_{n0}} \ldots \frac{y_{n3}}{z_{n3}}\right)$. Finalmente, se realiza la proyección de la imagen original sobre la región obtenida haciendo que el aporte al mosaico ya presente la rotación e integre la curvatura adecuada.

El segundo elemento en la composición es la selección de los píxeles y el peso dentro de la imagen final, se define como píxel de interés el peso de 1 y píxel de no interés al peso de 0. Para distribuir el peso dentro de la imagen existen varias técnicas en el estado del arte. En la figura 4.29 [12] se presenta un ejemplo de comparación entre ellas tomado de [22], y son: el promedio (*average*) (a), la mediana (*median*) (b), promedio de plumas (*feathered average*) (c), *p-norm =10* (d), el esquema Voronoi (e), la región de diferencias (*weighted ROD vertex cover with feathering*) (f), mezcla de costuras gráficas cortadas con Poisson *(graph cut seams with Poisson blending) (g)* y con mezcla mediante pirámide *(pyramid blending)* (h). Para mayor detalle de la comparación entre estas técnicas referirse directamente al documento [22].

Se puede ver en la Figura 4.29 que el mejor resultado, para el caso de panoramas es el caso (h), por tener un equilibrio de iluminación en la imagen definitiva y conservar la información que es constante en cada una de las imágenes de entrada. Caso contrario en los casos (a) y (d), donde aun se ve información del movimiento continuo de las personas. Y no es como el caso (b), en el cual se obtienen resultados radicales, como cortar información de las personas en la escena.

Como proceso final de la etapa de composición se encuentra el *blending*[13] el cual lleva a cabo la unión entre las imágenes con las máscaras de pesos generadas previamente y realiza

[12] Imagen tomada de [22]

[13] Traducción: mezcla

los retoques finales como son la compensación de las diferencias en exposición y la corrección de los desalineamientos que no se lograron compensar en las fases previas (Imagen inferior de la Figura 4.30).

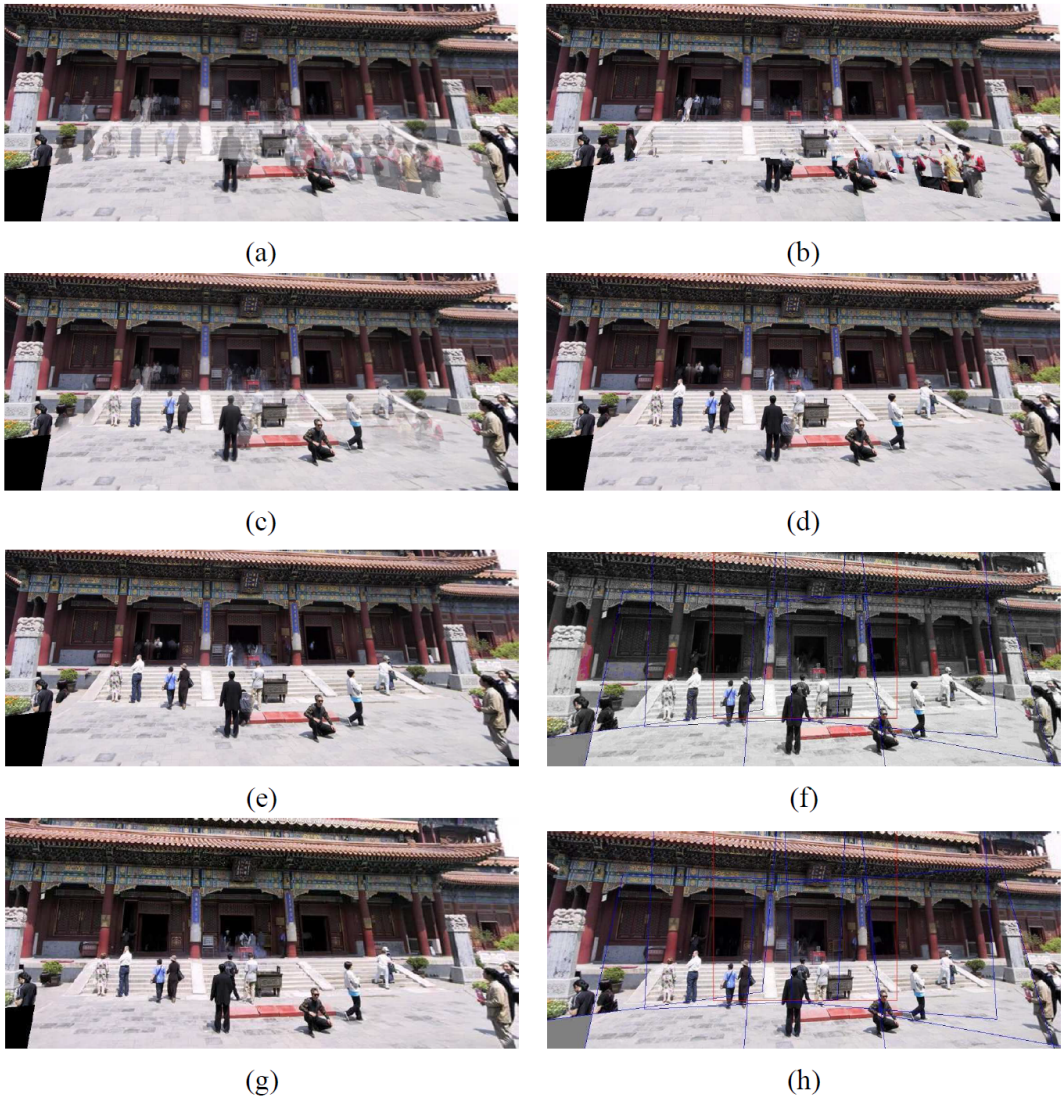


Figura 4.29: Ejemplo de algoritmos de distribución de pesos fase composición distribuidos así: el promedio (*average*) (a), la mediana (*median*) (b), promedio de plumas (*feathered average*) (c), *p-norm =10* (d), el esquema Voronoi (e), la región de diferencias (*weighted ROD vertex cover with feathering*) (f), mezcla de costuras gráficas cortadas con Poisson *(graph cut seams with Poisson blending) (g)*

Un ejemplo del proceso de salida en la selección de píxeles con imágenes sintéticas y la salida después del proceso de mezcla *(blending)* del algoritmo de mosaico definitivo, es presentado en la Figura 4.30. En la parte superior de la figura se presentan 4 imágenes, cada

una con un tamaño de 640 X 360[14], las cuales hacen parte de una de las imágenes de los fondos de la maqueta. En el medio, se muestran las máscaras de los pesos de los píxeles, en los cuales se interpreta al color blanco como la información que van a aportar cada una de las imágenes a la imagen de salida de mosaico, y el color negro a los píxeles que no. Se puede identificar que este proceso, permite tomar información de cada imagen, y por la forma de las máscaras, se tiene que el proceso de distribución de pesos no es ninguno de los mencionados en la referencia [22], sino por el contrario emplea una técnica que toma las regiones compartidas y distribuye la información que cada imagen aporta sin mezclar información entre regiones.

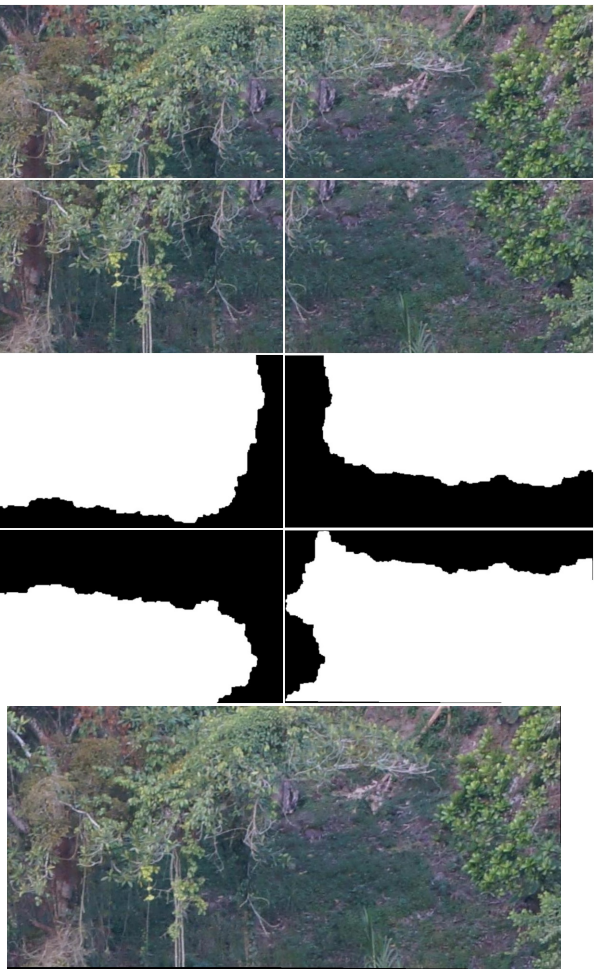


Figura 4.30: Ejemplo de las máscaras obtenidas después del proceso de selección de píxeles, en la etapa de composición en el algoritmo definitivo.

El proceso de composición es el de mayor dispersión en el consumo en tiempo de máquina como se presenta en la Figura 5.2, además, los últimos procesos de unión de mejor resultado de iluminación y equilibrio en la información de la imagen del estado del arte suelen emplear

[14]Tamaño de la imagen obtenida de la cámara inferior del ARDrone

un filtrado Gaussiano en cada una de las imágenes para lograr una imagen final totalmente corregida. Sin embargo, estos procesos no son necesarios para el presente proyecto, dado que el sistema no tiene como entrada todas las imágenes a ser usadas en la imagen de mosaico. Sino por el contrario, maneja a la entrada imágenes de ingreso secuencial, para ser unidas con una imagen global de mosaico. Lo que requiere que la imagen de salida o mosaico parcial no pierda cantidad de la información de las imágenes originales, para volver a ser empleada como entrada al proceso de mosaico. Por otra parte, el realizar el proceso de composición en su forma básica se reduce el tiempo de ejecución en esta etapa para que sea comparable con la etapa de registro.

4.3.2. Algoritmo de Mosaico *OpenCV*

El proceso de mosaico o *stitching* está escrito en las librerías de *OpenCV* en C++ como una clase, en la Figura 4.31 se presenta su esquema en bloques, el cual es coincidente con el proceso previamente descrito. Por otra parte, el enfoque original de este algoritmo es el descrito en la referencia [21], tal y como se menciona en la documentación de *OpenCV*. Y tiene como objetivo construir una imagen panorámica con las imágenes de entrada, sin importar su tiempo de ejecución y teniendo múltiples imágenes de entrada. Tampoco es de importancia el tamaño final de la imagen o si la imagen panorámica conserva la resolución de las imágenes originales. El algoritmo original busca generar el panorama con las imágenes de entrada y garantizar el equilibrio de iluminación en el resultado. A diferencia de la implementada en el presente proyecto, en la cual se realiza el proceso de mosaico con técnicas que buscan una imagen de mosaico de una superficie plana, con técnicas de menor demanda de tiempo y se actualiza la imagen de salida a medida que se capturan nuevas imágenes.

El algoritmo de mosaico original de *OpenCV,* dispone de una estructura global en la cual se emplean técnicas como por ejemplo el detector y descriptor de *SURF,* en caso de manejar librerías no gratuitas y *ORB* en caso contrario, además se realiza la estimación de la matriz de homografía a partir de las imágenes. Lo cual difiere con el algoritmo implementado, en el cual después de una evaluación presentada en la sección 5.1 se eligió a la técnica *FAST* como detector y a *BRISK* como descriptor y se calcula la matriz de homografía empleando la información filtrada de la *IMU* de la plataforma.

El algoritmo de mosaico original de *OpenCV* emplea una proyección geométrica esférica y aplica en el proceso de mezcla *(blender)* la técnica de multi-banda, lo cual difiere del algoritmo del proyecto en el cual se emplea la proyección en una superficie plana y la mezcla de las imágenes se realiza con el proceso más simple, el cual consiste en unir las imágenes a partir de las máscaras obtenidas con el proceso de selección de píxeles y pesos, como se puede ver en el ejemplo presentado en la sección 4.3.1.2 permitiendo reducir el tiempo de ejecución.

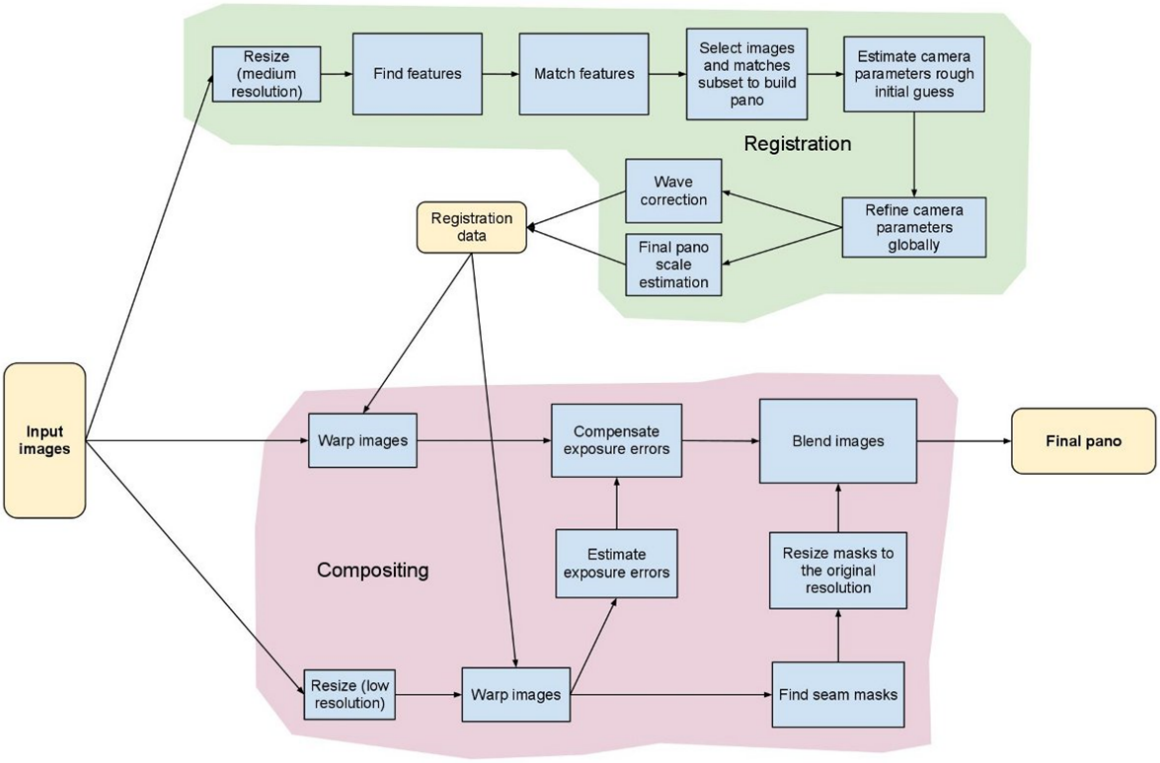


Figura 4.31: Diagrama de bloques del proceso de *Stitching* en el conjunto de librerias *OpenCV* versión *2.4.8*

4.3.3. Evaluación de detectores y descriptores.

Para la etapa de registro que pertenece al algoritmo de mosaico, se realizó la evaluación de los detectores y descriptores de puntos en la imagen, que aparecen en el Cuadro 4.5. Para lo cual, se eligen los parámetros de evaluación a describir en la sección 4.3.3.1, además se construye un algoritmo en el cual se aplican estos conceptos y se almacena los resultados, los cuales son presentados mediante parámetros estadísticos en la sección 5.1, estos resultados permiten realizar la mejor selección del conjunto detector-descriptor para la aplicación especifica del presente proyecto. El algoritmo implementado para la evaluación se describe en detalle en la sección 4.3.3.2.

4.3.3.1. Definición de parámetros de evaluación

Para la evaluación de los detectores y descriptores de puntos en la imagen seleccionados, se toman entre los criterios de evaluación propuestos en [46], cuyos principios son propuestos en [47], adicionalmente se tienen en cuenta los tiempos de ejecución del conjunto detector-descriptor, los tiempos de emparejamiento y el parámetro de *confidence*[15] propio de la etapa de registro como primera fase del proceso de mosaico, propuesto en [21].

[15]Traducción sugerida: confianza.

- Para la evaluación del detector se toma el criterio de evaluación denominado repetibilidad[16] definido en la Ecuación 4.13, el cual permite determinar la capacidad que tiene un detector de encontrar los mismos puntos clave[17] en una imagen A y una imagen B, siendo esta última el resultado de modificadores sobre la imagen A como iluminación y/o rotaciones y/o traslaciones. Se puede identificar que se requiere como complemento el concepto de correspondencia, que consiste en identificar cuáles de los puntos clave de la imagen A e imagen B que se sobrelapan, o pertenecen al mismo punto, lo cual se logra con la información de los de los *inliers* pertenecientes a los *matchers*[18]. Como se ha mencionado este evaluador integra varios elementos del proceso de emparejamiento, por esta razón fue elegido para este proceso de selección.

$$Repetibility(Rep) = \frac{Correspondence}{min(KeypointsA, KeypointsB)} \tag{4.13}$$

- Como criterio de evaluación del descriptor se selecciona *MAP* el cual es el valor medio de las pruebas, de los promedios de cada una de las pruebas Q, definida en la Ecuación 4.14 y es mencionado en la referencia [48]. Como complemento a la definición, se requiere la definición del promedio de los valores *Ave* de precisión P la cual se encuentra en la Ecuación 4.15, en la cual $P(k)$ se refiere al valor de precisión del evento $k-esímo$, cuya definición se presenta en la Ecuación 4.16 y $\triangle Rc(k)$ que puede tomar dos posibles valores $\triangle Rc(k) = 1$ si el valor de *recall* (definición en la Ecuación 4.17) en k ha tenido cambio en comparación con el valor en $k-1$ y $\triangle Rc(k) = 0$ en caso contrario. Como en el caso del evaluador previo se selecciona *MAP*, por la cantidad de información que aporta para realizar la selección de un descriptor de puntos en la imagen.

$$MAP = \frac{1}{|Q|} \sum_{q \in Q} AveP(q) \tag{4.14}$$

$$AveP(q) = \sum_{k=1}^{N} P(k) \triangle Rc(k) \tag{4.15}$$

$$1 - precision(P) = \frac{\#falsematches}{\#correctmatches + \#falsematches} \tag{4.16}$$

$$Recall(Rc) = \frac{\#correctmatches}{\#correspondences} \tag{4.17}$$

- Para la selección del conjunto detector-descriptor, también se tiene en cuenta el tiempo de ejecución t_{dd} y el tiempo que requiere en dar resultados el *matcher* t_m seleccionado para el algoritmo de la etapa de registro, parte del algoritmo de mosaico. Se toman en cuenta estos parámetros de ejecución de tiempo, dado que es parte del presente proyecto ejecutar en línea el proceso de mosaico.

[16] De la palabra en ingles *repetibility*
[17] *Keypoints*
[18] Operación definida como correspondencia o *correspondence* en ingles.

- Como elemento de evaluación asociado directamente con la etapa de registro en el algoritmo de mosaico, es el valor de *confidence* C_{on} definido en la Ecuación 4.18. El cual es el resultado de aplicar un modelo de probabilidad binomial explicado en detalle en [21], el cual determina si una nueva imagen a la entrada del algoritmo, aporta nueva información visual o por el contrario la información visual se encuentra representada en el conjunto de imágenes previamente capturadas o por la imagen de mosaico previamente generada. Para ello, se tiene que para una imagen cuyo valor de *confidence* es superior a 3, valor obtenido empíricamente y empleado en el algoritmo de mosaico en *OpenCV,* indica que ya está contenida en otra u otras imágenes. Y si es inferior a 3, implica que contiene nueva información, sin embargo, en presencia de pocos *inliers* la imagen también es rechazada por no tener regiones comunes para una posterior unión. Este parámetro fue seleccionado, dado a la importancia dentro de la etapa de registro y favorecer la ejecución del algoritmo de mosaico, además esté permite determinar si las imágenes a capturar en el escenario contienen puntos clave suficientes.

$$Confidence\,(C_{on}) = \frac{\#inliers}{8 + 0,3 * \#matches} \tag{4.18}$$

4.3.3.2. Algoritmo para evaluación

Para aplicar los parámetros de evaluación descritos en la sección 4.3.3.1, se construyó un algoritmo cuya descripción simbólica se realiza mediante un diagrama de flujo. Este se presenta en la Figura 4.34 en cual se muestra la estructura general de los ciclos de evaluación de cada uno de los detectores, con respecto a cada uno de los descriptores. Y en la Figura 4.33 se presenta el detalle de la evaluación por cada conjunto de detector-descriptor, indicando los datos almacenados en archivos separados por coma (*.csv*).

El tamaño de cada una de las imágenes de prueba es de 640 X 360 píxeles, dado que son capturadas con la cámara inferior del *ARDrone 2.0* a una altura aproximada de un metro, sin los motores activos de la plataforma. Se emplea la maqueta de líneas de transmisión cuyo diseño y construcción fue previamente mencionada en un entorno cubierto y controlado, la vista superior de la maqueta previa a la captura de las imágenes se presenta en la Figura 4.12. En total se emplean para la evaluación 27 imágenes de prueba que se presentan en la Figura 4.32.

Figura 4.32: Imágenes tomadas sobre la maqueta de líneas de transmisión con la cámara inferior del *ARDrone 2.0* a una altitud aproximada de un metro, con diferentes iluminaciones para la evaluación de los conjuntos detector-descriptor.

Se tiene que cada una de las imágenes de entrada al algoritmo son evaluadas por un conjunto de detector-descriptor, cuyos casos evaluados son presentados en su totalidad en el Cuadro 4.7, dando un total de 28 casos conjunto detector-descriptor por evaluar. Por otra parte, en el Cuadro 4.8, se presentan los casos de conjunto detector-descriptor de los cuales no se realizó la evaluación, dado que no eran compatibles entre sí o los descriptores no son compatibles con el algoritmo de emparejamiento o *matcher* elegido para el algoritmo principal de mosaico.

#	Detector	Descriptor	#	Detector	Descriptor
1	SIFT	SIFT	15	ORB	BRISK
2	SIFT	SURF	16	ORB	BRIEF
3	SIFT	BRISK	17	FAST	SIFT
4	SIFT	BRIEF	18	FAST	SURF
5	SIFT	FREAK	19	FAST	ORB
6	SURF	SIFT	20	FAST	BRISK
7	SURF	SURF	21	FAST	BRIEF
8	SURF	ORB	22	FAST	FREAK
9	SURF	BRISK	23	MSER	SIFT
10	SURF	BRIEF	24	MSER	SURF
11	SURF	FREAK	25	MSER	ORB
12	ORB	SIFT	26	MSER	BRISK
13	ORB	SURF	27	MSER	BRIEF
14	ORB	ORB	28	MSER	FREAK

Cuadro 4.7: Casos evaluados del conjunto detector-descriptor

#	Detector	Descriptor
1	*SIFT*	*ORB*
2	*ORB*	*FREAK*
3	*BRISK*	*SIFT*
4	*BRISK*	*SURF*
5	*BRISK*	*ORB*
6	*BRISK*	*BRISK*
7	*BRISK*	*BRIEF*
8	*BRISK*	*FREAK*

Cuadro 4.8: Casos detector-descriptor no evaluados por incompatibilidad

En la Figura 4.33, se tiene el diagrama de flujo que indica que se realizan por imagen un # Pruebas por imagen, en el caso de la evaluación realizada, se tomaron un total de 5 casos, y es en esta fase en la que se realiza el emparejamiento y se determina la efectividad de los

detectores y descriptores. Por cada uno de los casos se almacena una imagen en la cual se puede ver los puntos clave conectados en caso de ser *inliers* y los que no.

Se realizaron un total de 135[19] pruebas por cada conjunto detector-descriptor, para tener en definitiva 3780[20] pruebas para garantizar la correcta selección del detector-descriptor para el algoritmo de mosaico, los resultados se presentan en la sección 5.1.

En la Figura 4.34 se presenta el detalle de la subtarea del diagrama de flujo presentado en el diagrama de la Figura 4.33, en este se tiene el detalle de la información que se almacena por cada conjunto detector-descriptor.

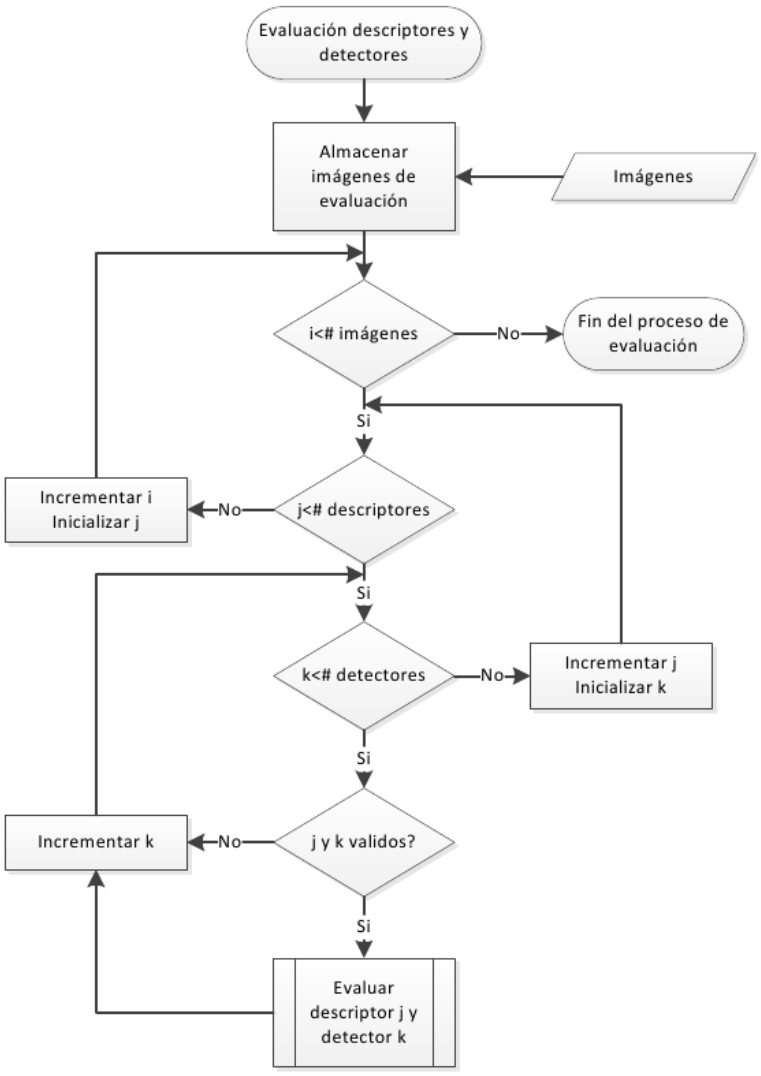


Figura 4.33: Diagrama de flujo general del algoritmo para la evaluación del conjunto de detectores y descriptores presentados en el Cuadro 4.7.

[19]Número resultado de 27 imágenes de entrada y 5 casos de rotación por imagen.

[20]Número resultado de 27 imágenes de entrada, 28 casos detector-descriptor y 5 casos de rotación por imagen.

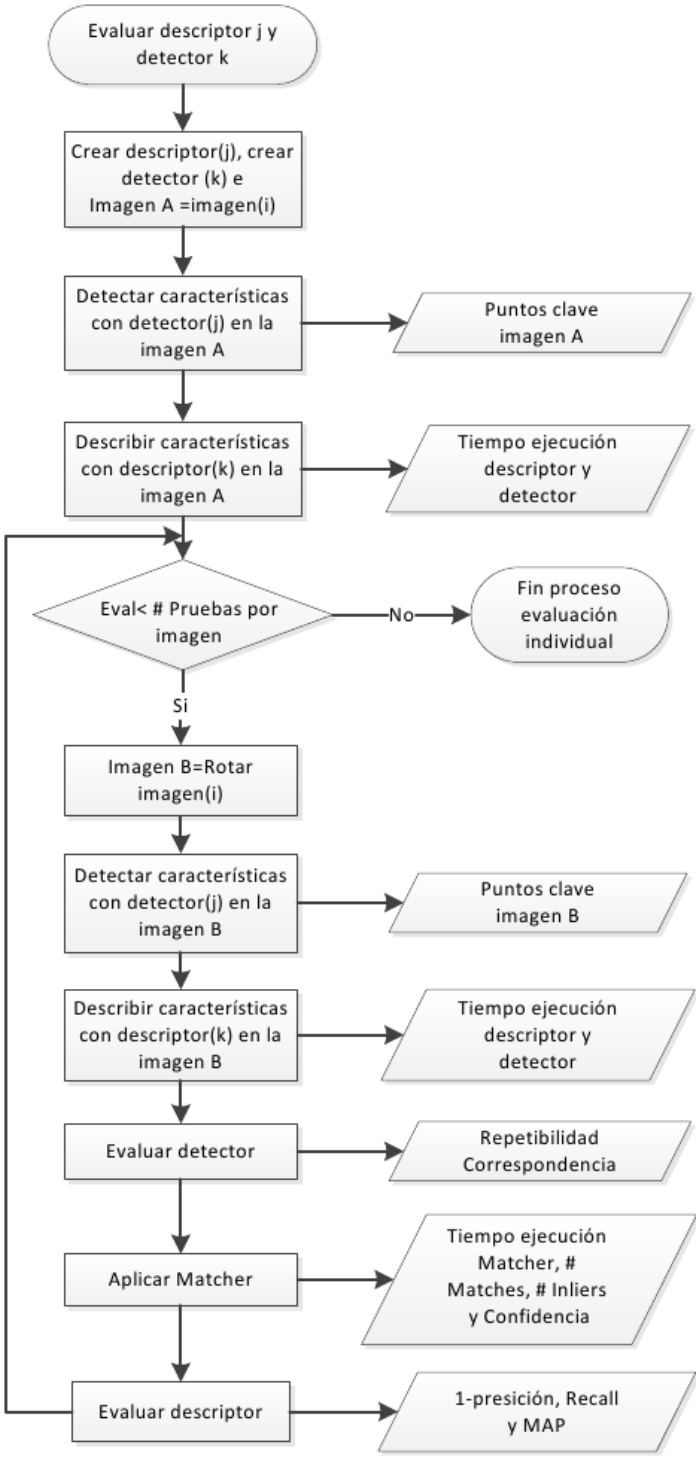


Figura 4.34: Diagrama de flujo del detalle del algoritmo de evaluación para cada detector y descriptor, como subtarea del diagrama de flujo presentado en la Figura 4.33.

4.3.4. Algoritmo de Mosaico con la Integración en *ROS*.

En la Figura 4.35 se presenta el diagrama de flujo general del nodo desarrollado en *ROS* en el cual se dividió en dos fases el proceso de mosaico. Inicialmente, se indica por teclado el momento de arranque de captura de la información de los sensores embarcados en la plataforma. Para asegurar el correcto funcionamiento del proceso de captura, se da el comando de inicio a la altitud en la cual se va a realizar el desplazamiento, además se debe garantizar que la información contenida en la imagen hace parte de la zona deseada. Dado que la primera imagen de entrada del algoritmo es la base de comparación para los procesos subsiguientes del algoritmo.

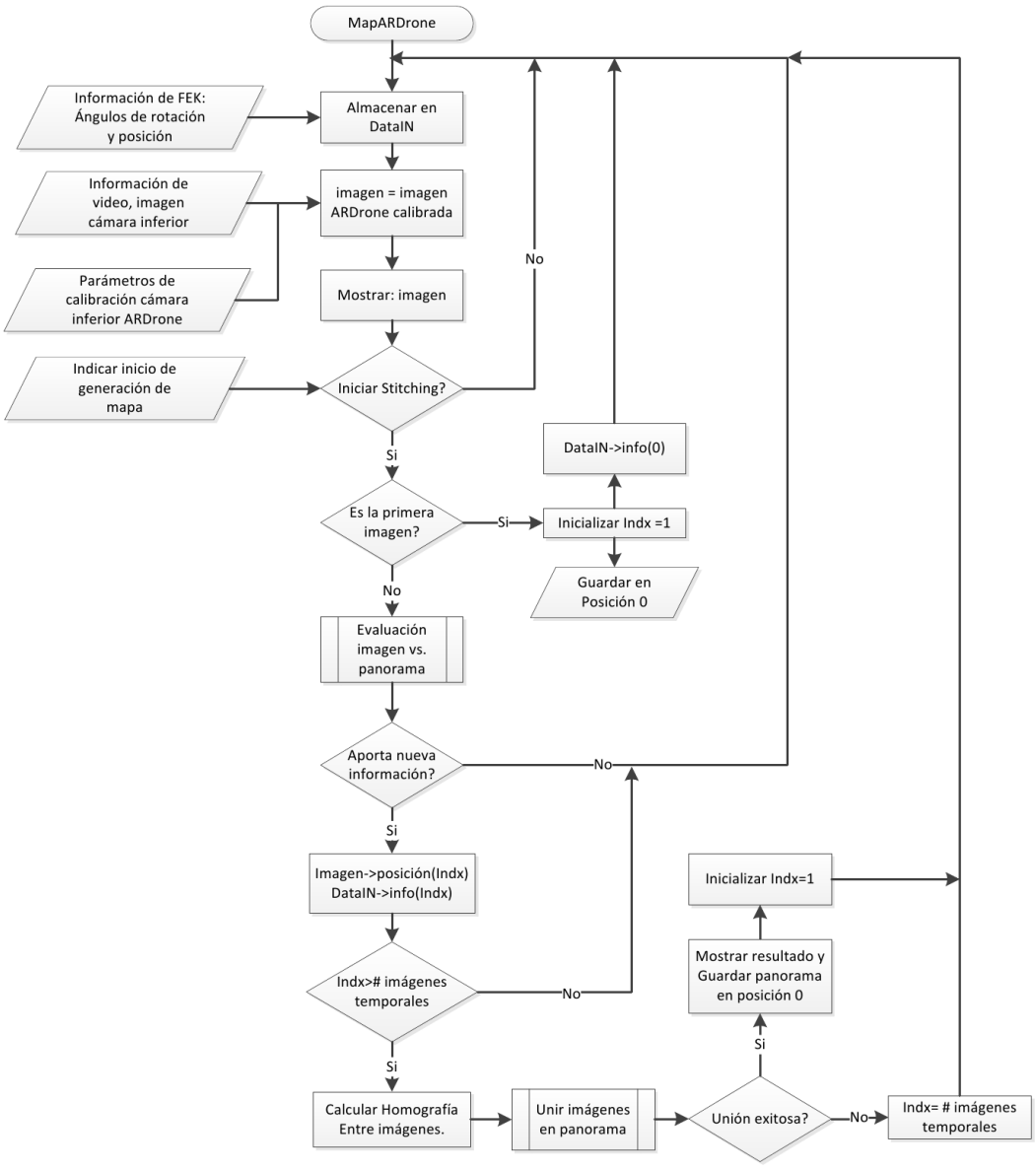


Figura 4.35: Diagrama de flujo del algoritmo principal para generar los Mapas de superficie

Posterior al proceso de arranque del algoritmo, se ejecuta la primera etapa del mosaico de imágenes, la cual está asociada a la fase de registro descrita en la sección 4.3.1.1, además se relaciona directamente con los resultados del algoritmo de evaluación presentados en la sección 4.3.3. Se presenta el algoritmo de la fase de registro, en el diagrama de flujo de la Figura 4.36, esta etapa es la encargada de evaluar si la imagen capturada proporciona nueva información al proceso de mosaico aplicando la Ecuación 4.18 o por el contrario no es coincidente, determinado a partir de la información de correspondencia. Si la nueva imagen cumple con los requerimientos, se almacena la información sensorial asociada a ella y se prosigue a evaluar una nueva imagen o a realizar el proceso de composición. Esta última selección depende del número de imágenes que ya se tengan almacenadas, en el caso implementado se tomó el valor de 3 imágenes.

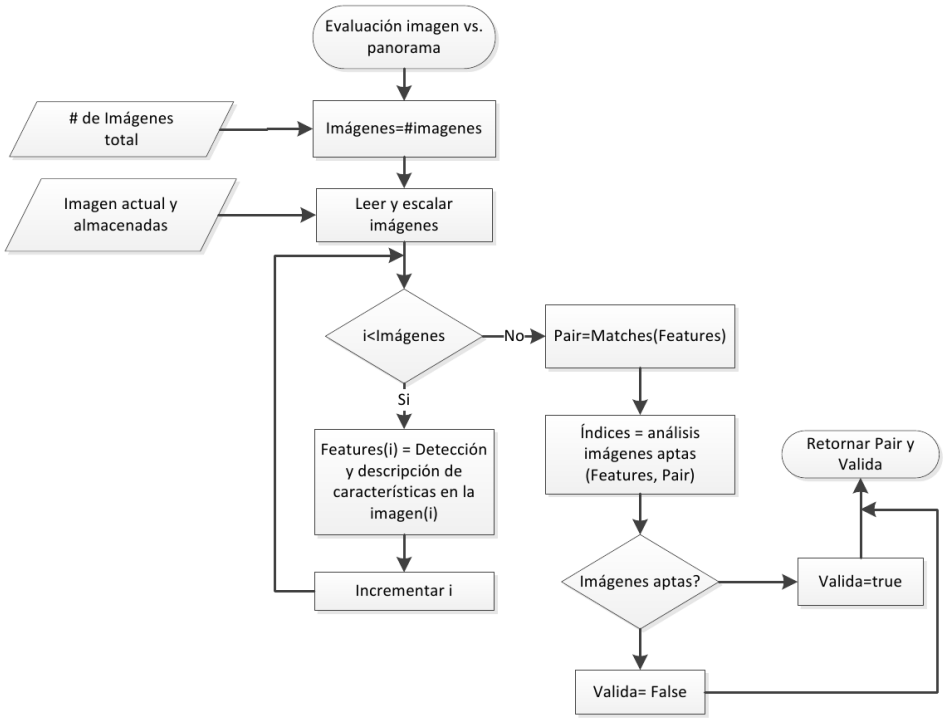


Figura 4.36: Diagrama de flujo de la subtarea Evaluación imagen vs. panorama del diagrama de flujo de la Figura 4.35. En esté diagrama se presenta el algoritmo evaluación realizada a cada nueva imagen capturada por la plataforma.

Para finalizar el proceso de mosaico en *ROS*, se tiene el algoritmo asociado al proceso de composición descrito en la sección 4.3.1.2 y presentado en el diagrama de flujo de la Figura 4.37, el cual recibe la información de emparejamiento extraída en la etapa previa, con la homografía entre imágenes calculada con la Ecuación 4.9 a partir de la información sensorial filtrada previamente por un FEK.

La etapa de composición, tiene como propósito integrar las imágenes de entrada en una sola, considerando las rotaciones y la forma final de la imagen deseada, que en el caso del

presente proyecto es un plano. En caso de tener éxito en el proceso de composición, se almacena la imagen obtenida en la posición de la imagen origen sin compresión para evitar pérdidas, de tal manera que sea empleada en el siguiente ciclo de mosaico. Sin embargo, existen casos en los cuales no se obtiene éxito al unir las imágenes, esto ocurre en los casos en que la información de las imágenes no se logra proyectar sin dañar la información.

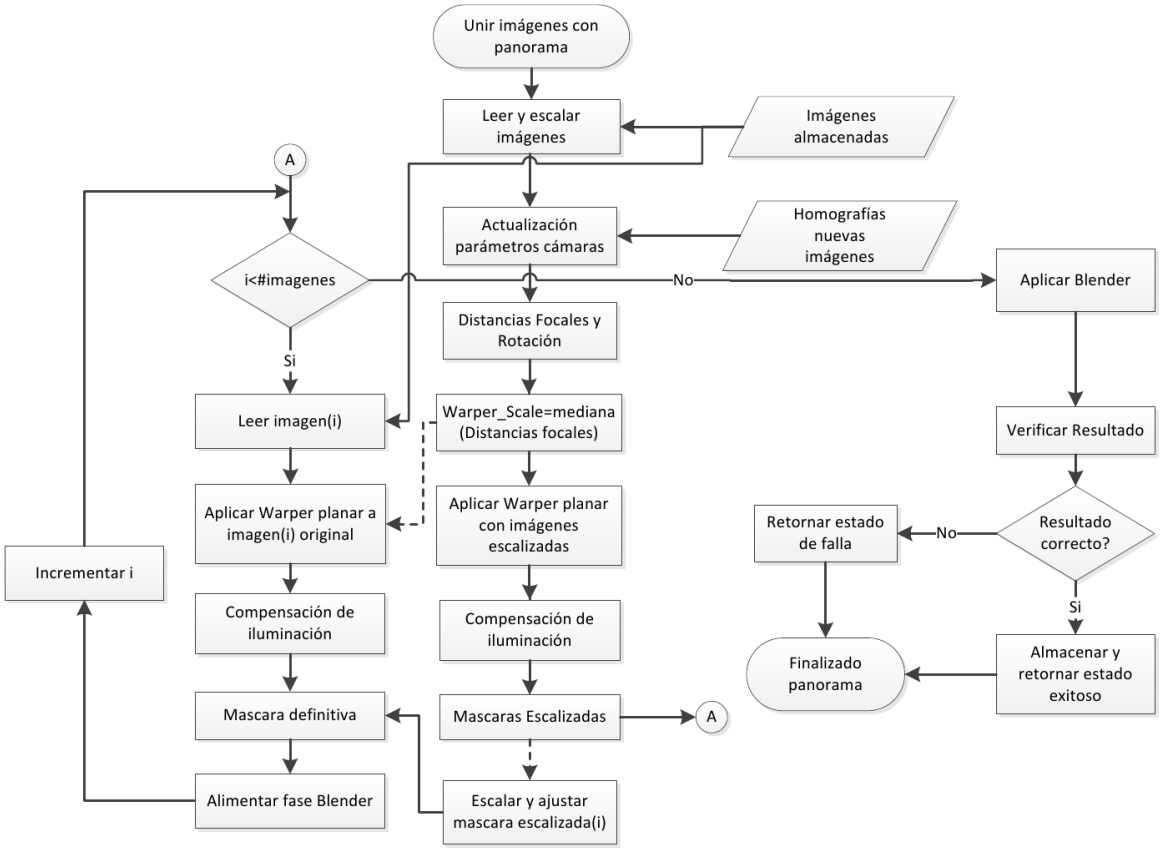


Figura 4.37: Diagrama de flujo de la subtarea Unir imágenes con panorama del diagrama de flujo de la Figura 4.35 . Esté diagrama ilustra el funcionamiento de la etapa de composición.

En conclusión, con el proceso de registro se verifica si la nueva imagen, contiene información adicional o ninguna con respecto a la imagen de mosaico y las imágenes candidatas a ser parte del mosaico. Sin embargo, este proceso no asegura que se va a tener éxito en la etapa de composición, dado que este último implica modificaciones geométricas que pueden dañar la información contenida en las imágenes.

4.4. Pruebas de calidad y desempeño del sistema

Para las pruebas de calidad del sistema y el adecuado desempeño del sistema se presenta a continuación tres fases: la primera con los pasos de ejecución iniciales para la toma de

datos con el sistema en total operación, la segunda con las características que garanticen la calidad de la captura de los datos y funcionamiento del sistema. Para finalmente, realizar una evaluación comparativa entre las imágenes tomadas con condiciones controladas y las imágenes obtenidas a la salida del sistema de mosaico de imágenes.

4.4.1. Verificación de condiciones iniciales

1. Verificar el estado de las comunicaciones entre estación base y plataforma aérea.
2. Instalar las torres de transmisión en el escenario de pruebas para realizar la toma de datos con la plataforma aérea.
3. Revisar que el escenario de pruebas no presente rastros de arena, polvo, sustancias líquidas, para evitar fallos en la plataforma al aterrizar o despegar.
4. Verificar que no se presenten obstáculos en la zona de vuelo de la plataforma, y garantizar que no se hallan objetos en movimiento en la zona de interés, para permitir una ejecución del algoritmo de mosaico en forma constante.
5. Realizar una toma de imágenes controladas con una cámara externa a la plataforma a una altura superior a la de vuelo y de mayor resolución a la cámara embarcada en la parte inferior de la plataforma, para la evaluación posterior.
6. Ejecución inicial de los nodos de *ROS* requeridos para el funcionamiento de la plataforma y del sistema de navegación, para compensar el filtro de Kalman.
7. Inicialización de nodo *ROS - OpenCV*, etapa de inicio de mosaico de imágenes.

4.4.2. Condiciones de calidad durante la ejecución de trayectoria.

1. Estado activo de todos los nodos de ROS, durante la ejecución de la trayectoria.
2. Mosaico de imágenes parcial de la zona recorrida, presentado en tiempo de ejecución al usuario.
3. Almacenamiento de información de imágenes y sensores embarcados en la plataforma durante la ejecución.
4. Aterrizaje de la plataforma en un aérea externa a la simulación de líneas de transmisión.

4.4.3. Condiciones de calidad después de la ejecución.

1. Plataforma sin afectaciones físicas producidas por problemas en ejecución de trayectoria.
2. Mosaico de imágenes final (mapa) de la zona recorrida por el vehículo aéreo.

Capítulo 5

Análisis de resultados

5.1. Resultados evaluación conjunto detector-descriptor

En esta sección se presentan los resultados obtenidos de los criterios de evaluación de los conjuntos detector-descriptor presentados en el Cuadro 4.7, además la selección final del conjunto detector-descriptor empleado en la fase de registro del algoritmo de mosaico definitivo, cuya descripción teórica se encuentra en la sección 4.3.1.1 e implementación práctica, hace parte del diagrama de flujo del algoritmo de mosaico implementado en *ROS* (Figura 4.35), como primera sub-tarea del algoritmo. La implementación práctica de la etapa de registro es presentada en detalle con en el diagrama de flujo de la Figura 4.36.

En el Cuadro 5.1, se presentan los resultados obtenidos de cada uno de los criterios de evaluación, elegidos y descritos en la sección 4.3.3. A continuación se explica en detalle el contenido de este cuadro:

- La columna *Caso*, hace referencia al número del conjunto detector-descriptor evaluado cuyo listado se presentó en el Cuadro 4.7.

- En cada uno de los criterios de evaluación, se indica el valor promedio $\mu_{Criterio}$ de las muestras[1] en cada uno de los casos y la posición $P_{Criterio}$ obtenida en el criterio, en comparación con la totalidad de los casos. Esté valor va de 1 a 28 y en caso de empate se duplica el valor de la posición. El mejor conjunto detector-descriptor en cada uno de los criterios de evaluación se presenta en color azul.

- La columna *Rep*, hace referencia al criterio de evaluación de repetibilidad (ver Ecuación 4.13). Cuyo rango de operación va desde 0 hasta 1, en donde 1 es el caso de 100 % repetibilidad y 0 indica que no presenta ninguna repetibilidad. Como complemento a la información del cuadro, se tienen dos gráficas. La primera, en la Figura 7.3 en la cual se presenta en forma gráfica la información de valor medio y desviación estándar de cada uno de los casos (eje horizontal) y la segunda gráfica, es

[1] Número de muestras por cada criterio de evaluación es de 135, resultado de 27 imágenes de prueba (ver Figura 4.32) sobre la maqueta de líneas de transmisión (ver sección 4.1.1.3) y 5 casos de rotación por imagen.

la Figura 7.4 en la que se encuentra el diagrama de caja[2], los mejores casos de este parámetro son el caso 17 (Detector: *FAST*, Descriptor: *SIFT*) y el caso 18 (Detector: *FAST*, Descriptor: *SURF*). Adicionalmente, es notable que el detector *FAST* está en la mayoría de los mejores resultados, dados por este criterio.

- La columna *MAP*, hace referencia al criterio de evaluación *MAP* (ver Ecuación 4.14). Cuyo rango de operación va desde 0 hasta 1, en donde 1 es el caso de mayor precisión o el 100 % de precisión media y 0 indica que no tiene ninguna precisión. Como se mencionó en el criterio de evaluación previo se dispone de dos gráficas, en la Figura 7.5 se presenta la información de valor medio y desviación estándar de cada uno de los casos y en la Figura 7.6 se presenta el diagrama de caja, el mejor caso de este parámetro es el caso 1 (Detector: *SIFT*, Descriptor: *SIFT*). Para esté criterio de evaluación se tiene que en promedio los mejores resultados se dan con el descriptor *BRISK,* sin afectar el detector asociado.

- La columna C_{on}, hace referencia al criterio de evaluación de *confidence* (ver Ecuación 4.18). En el cual, el valor de *confidence* mayor o igual a 3 implica que el proceso de registro ha encontrado que las imágenes son coincidentes, es decir contienen la misma información. En el caso de este proceso de evaluación y selección, es de interés los casos que superan esta cota, ya que se emplea la misma imagen de entrada modificada por una matriz de homografía para realizar la evaluación. Por otra parte, en la Figura 7.7 se presenta la información de valor medio y desviación estándar de cada uno de los casos y en la Figura 7.8 se presenta el diagrama de caja, el mejor caso de este parámetro es el caso 9 (Detector: *SURF*, Descriptor: *BRISK*). En este criterio se tiene que el descriptor *BRISK*, presenta los mejores resultados.

- La columna t_{dd}, hace referencia al criterio de evaluación de tiempo de ejecución del conjunto detector-descriptor, en este parámetro se da la mejor posición al menor tiempo dado en segundos y la última posición al mayor tiempo de ejecución. Además, en la Figura 7.9 se presenta la información de valor medio y desviación estándar de cada uno de los casos y en la Figura 7.10 se presenta el diagrama de caja, el mejor caso de este parámetro es el caso 21 (Detector: *FAST*, Descriptor: *BRIEF*). En general, para este criterio de evaluación el detector *FAST* hace parte de los de menor duración de ejecución.

- La columna t_m, hace referencia al criterio de evaluación de tiempo de ejecución del conjunto detector-descriptor con el algoritmo de emparejamiento *Flann*, en este parámetro se dan las posiciones como en el criterio anterior, la mejor posición al menor tiempo dado en segundos y la última posición al mayor tiempo de ejecución. Y como en los casos previos se tienen dos gráficas en la Figura 7.11 y en la Figura 7.12

[2]Un diagrama de caja es un gráfico, basado en cuartiles, mediante el cual se visualiza un conjunto de datos. Está compuesto por un rectángulo, la "caja", y dos brazos, los "bigotes". Es un gráfico que suministra información sobre el los valores mínimo y máximo, los cuartiles Q1, Q2 o mediana y Q3, y sobre la existencia de valores atípicos y la simetría de la distribución. Definición tomada de [49]

en las cuales se tiene la información del valor medio con desviación estándar y el diagrama de caja, el mejor caso de este parámetro es el caso 28 (Detector: *MSER*, Descriptor: *FREAK*).

- En la última columna $P_{osición}$ se lista la posición obtenida, al tomar cada una de las posiciones de los criterios de evaluación, promediar su resultado y ordenar de menor a mayor, para poder elegir el conjunto detector-descriptor que brinde mayor equilibrio entre los criterios de evaluación. En esta columna, se indica el caso final seleccionado en color verde.

Después de obtener los resultados de los criterios de evaluación elegidos, se tiene que entre los 28 casos de conjunto detector-descriptor se elige el caso #20, el cual hace referencia al detector *FAST* con descriptor *BRISK*. Este conjunto es elegido por tener como detector a *FAST* el cual está entre los conjuntos de mayor repetibilidad y está entre los tiempos más rápidos de ejecución del detector-descriptor. Adicionalmente, se tiene *BRISK* que se encuentra entre los descriptores de mayor *confidence* y en promedio entrega los mejores resultados en *MAP* independientemente al detector empleado, afirmación previamente realizada sobre un conjunto de imágenes diferentes en la referencia [48]. Por otra parte, el conjunto detector-descriptor #20 está en las primeras posiciones de la columna *P* y es el primer conjunto en el que no se emplea ni *SIFT*, ni *SURF*, ya que como se indicó en el capítulo 3 de especificaciones, es de interés del presente proyecto emplear un detector-descriptor de libre desarrollo y que haga parte de las técnicas más recientes del estado del arte, dado que estas fueron desarrolladas e implementadas para aplicaciones en tiempo real, parámetro relacionado en forma directa con el presente proyecto.

Caso	Rep		MAP		C_{on}		$t_{dd}(s)$		$t_m(s)$		$P_{osición}$
#	μ_{Rep}	P_{Rep}	μ_{MAP}	P_{MAP}	$\mu_{C_{on}}$	$P_{C_{on}}$	$\mu_{t_{dd}}$	$P_{t_{dd}}$	μ_{t_m}	P_{t_m}	
1	0,6723	19	0,9727	1	4,5640	6	0,1165	16	0,0074	7	3
2	0,6723	19	0,9475	3	4,1234	9	0,0786	12	0,0059	4	2
3	0,6807	17	0,9610	2	4,5330	7	0,1305	22	0,0081	8	7
4	0,6784	18	0,9351	5	4,1128	10	0,0685	10	0,0053	3	1
5	0,6846	13	0,1944	27	0,2643	27	0,0759	11	0,0036	2	18
6	0,7841	3	0,9320	6	5,2324	5	0,7770	28	0,0437	26	16
7	0,7841	3	0,9256	8	5,6408	2	0,2748	26	0,0338	23	12
8	0,7354	7	0,9270	7	5,5142	4	0,0913	14	0,0434	25	8
9	0,6965	10	0,9454	4	5,7649	1	0,1494	24	0,0387	24	14
10	0,7454	6	0,9137	9	5,5563	3	0,0887	13	0,0489	27	9
11	0,7159	9	0,4177	25	0,7589	25	0,0945	15	0,0336	22	24
12	0,6810	16	0,5707	24	1,6053	24	0,1187	17	0,0106	13	22
13	0,6832	14	0,8307	14	3,4177	15	0,1284	20	0,0090	10	17
14	0,6832	14	0,8268	15	3,3699	16	0,0072	4	0,0089	9	9
15	0,5603	24	0,6905	23	2,5147	21	0,0637	9	0,0070	5	19
16	0,6956	10	0,8339	13	3,3139	17	0,0041	3	0,0105	11	6
17	0,8088	1	0,8634	12	4,0227	12	0,0515	7	0,0264	20	4
18	0,8088	1	0,7564	22	3,5411	14	0,0155	6	0,0235	19	12
19	0,6677	21	0,8183	16	4,1014	11	0,0028	2	0,0217	17	15
20	0,7744	5	0,8768	11	4,4551	8	0,0630	8	0,0309	21	5
21	0,6877	12	0,8156	17	3,9455	13	0,0025	1	0,0234	18	11
22	0,7201	8	0,2910	26	0,6819	26	0,0100	5	0,0771	28	21
23	0,5810	22	0,7884	19	2,5148	20	0,3128	27	0,0131	16	27
24	0,5810	22	0,8040	18	2,5872	19	0,1481	23	0,0105	12	22
25	0,4584	26	0,7672	21	2,2674	23	0,1214	18	0,0112	14	26
26	0,4498	27	0,8939	10	2,6237	18	0,1811	25	0,0073	6	20
27	0,4703	25	0,7825	20	2,3170	22	0,1218	19	0,0115	15	25
28	0,3359	28	0,1092	28	0,1308	28	0,1295	21	0,0034	1	28

Cuadro 5.1: Información de posición obtenida de cada conjunto detector-descriptor, para cada uno de los criterios de evaluación, en el cual se indica de color azul las primeras posiciones y en color verde el caso elegido.

5.2. Resultados algoritmo de Mosaico

Esta sección es para presentar los resultados obtenidos, posterior a la implementación del algoritmo de mosaico, presentado en la sección 4.3.4. Inicialmente, se determina el tiempo de ejecución entre actualizaciones de la imagen de mosaico de salida. Para ello se presenta la información en la Figura 5.1 y el Cuadro 5.2, en las cuales se muestran los tiempos obtenidos

dependiendo del número de mosaicos ejecutados por el algoritmo, eje *x* en la figura y primera columna en el cuadro, esto dado que se espera que el tiempo aumente a medida que se ha condensado mayor número de imágenes en la imagen de mosaico.

En la Figura 5.2 se presentan los diagramas de caja por etapa de ejecución en el mosaico, en la cual se puede ver que los tiempos de ejecución son semejantes. Sin embargo, la etapa de composición presenta mayor proporción de casos de dispersión en los cuales el tiempo de ejecución se acerca a 1 segundo.

Por otra parte, en el Cuadro 5.2 se tiene el número de muestras que se emplearon para encontrar la mediana en cada uno de los casos. Estas muestras son igualmente empleadas en la Figura 5.1.

Se puede ver que el tiempo de ejecución entre actualización de mosaicos no es afectado linealmente por el número de mosaicos previos, sino que depende directamente de la información de las nuevas imágenes capturadas en comparación con el mapa actual, es decir, si las imágenes capturadas no presentan puntos clave que permitan ser asociadas con la imagen de mosaico, o la información de las nuevas imágenes ya hace parte de la imagen de mosaico. Lo cual, es notable en la dispersión presentada con el diagrama de caja, que no es simétrica y se presentan casos de *outlier*s.

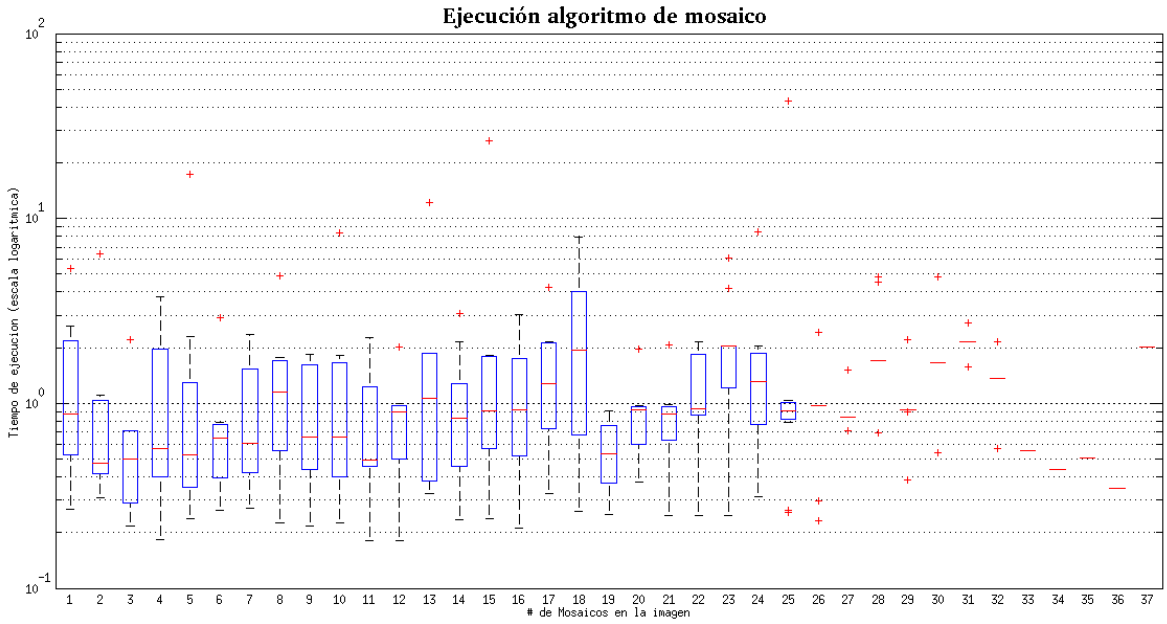


Figura 5.1: Diagrama de caja de tiempo de ejecución del algoritmo de Mosaico vs. número de mosaicos realizados

#	*Muestras*	*Med*(*s*)	#	*Muestras*	*Med*(*s*)	#	*Muestras*	*Med*(*s*)
1	11	0,8762	13	10	1,0677	25	6	0,9162
2	11	0,4740	14	10	0,8355	26	5	0,9749
3	11	0,4999	15	10	0,9049	27	5	0,8462
4	11	0,5671	16	10	0,9281	28	5	1,6925
5	11	0,5266	17	10	1,2843	29	5	0,9253
6	11	0,6475	18	9	1,9441	30	3	1,6689
7	11	0,6092	19	7	0,5331	31	2	2,1472
8	11	1,1497	20	7	0,9258	32	2	1,3610
9	10	0,6618	21	7	0,8801	33	1	0,5558
10	10	0,6589	22	7	0,9359	34	1	0,4404
11	10	0,4936	23	7	2,0305	35	1	0,5067
12	10	0,8976	24	6	1,3152	36	1	0,3453

Cuadro 5.2: Tabla de los datos de las muestras de los tiempos de ejecución algoritmo mosaico

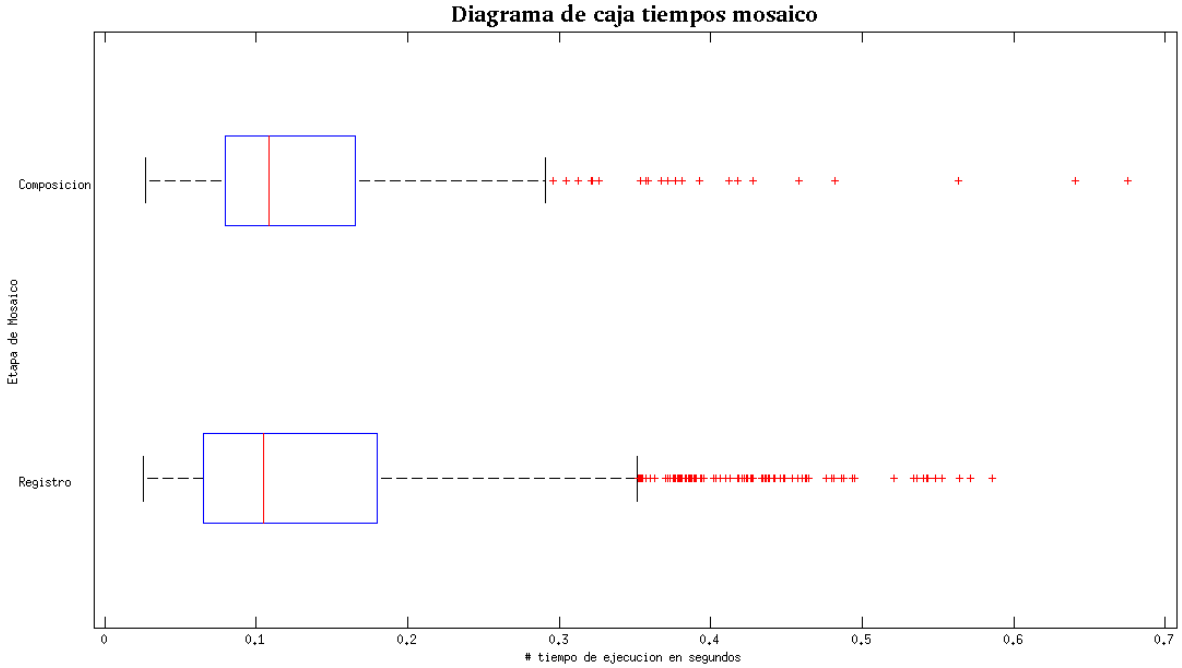


Figura 5.2: Diagrama de caja de tiempo de ejecución del mosaico por etapas.

Por otra parte, en esta sección se presentan imágenes ejemplo de mapas obtenidos con la aplicación, para ambientes internos y ambientes externos, son presentados diferentes casos y lugares.

5.2.1. Resultados de ejecución en ambientes internos

En el Cuadro 5.3, se presenta en la primera columna el número de figura a la que se hace referencia, en la segunda el tamaño en píxeles, en la tercera el número de mosaicos requeridos para llegar a la imagen presentada, en la cuarta columna se muestra la vista superior del escenario de pruebas y en la quinta columna, indica la forma de obtener la matriz de homografía, en el caso de indicar *Sensor* se obtiene la matriz con la Ecuación 4.9 a partir de la información de los sensores embarcados filtrados mediante un FEK tal y como se presenta en el diagrama de bloques de la Figura 4.25 y en caso de indicar *Imagen* es porque la matriz de homografía es estimada a partir de la información de la imagen. Cada nuevo mosaico implica 2 imágenes nuevas, por ejemplo en el caso de tener 13 mosaicos se sabe que está compuesto de 26 imágenes nuevas y una imagen base, para un total de 27 imágenes.

Figura	*Tamaño*	*Mosaico*	*Escenario*	*Homografía*
5.3	1470 x 799	10	Fig.4.13	Imagen
5.5	1580 x 411	17	Fig.5.4	Sensor
5.6	1088 x 232	11	Fig.5.4	Sensor
5.7	658 x 367	22	Fig.5.4	Sensor
5.8	2410 x 733	13	No aplica	Imagen

Cuadro 5.3: Información mosaicos ambientes internos

Figura 5.3: Ejemplo 1 escenario interno de resultado del algoritmo de mosaico en ambiente real, con homografía estimada a partir de las imágenes, compuesta por 21 imágenes.

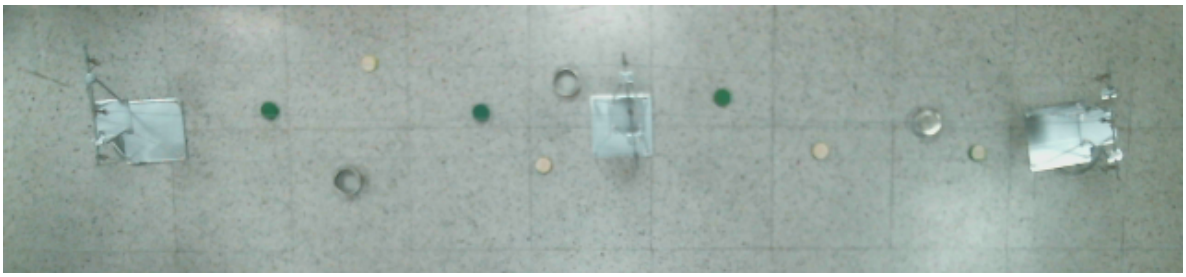


Figura 5.4: Imagen vista superior de escenario de pruebas en escenario interno

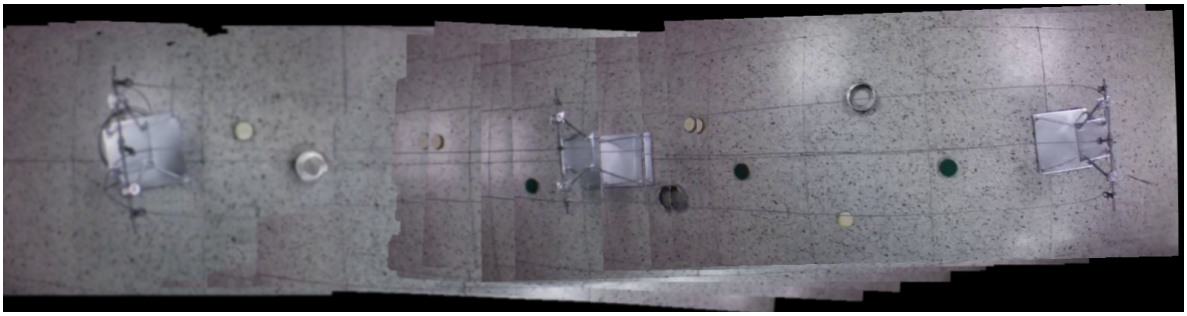


Figura 5.5: Ejemplo 2 escenario interno de resultado del algoritmo de mosaico en ambiente real, con homografía calculada con la información sensorial embarcada, compuesta por 35 imágenes.

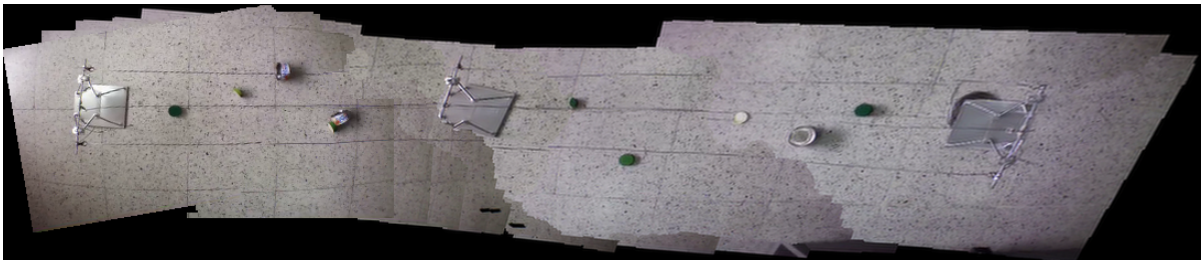


Figura 5.6: Ejemplo 3 escenario interno de resultado del algoritmo de mosaico en ambiente real, con homografía calculada con la información sensorial embarcada, compuesta por 23 imágenes.

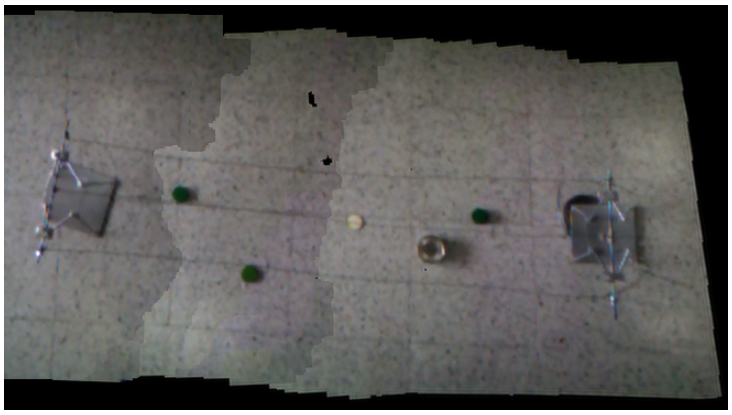


Figura 5.7: Ejemplo 4 escenario interno de resultado del algoritmo de mosaico en ambiente real, con homografía calculada con la información sensorial embarcada, compuesta por 45 imágenes.

Un ejemplo de la unión de imágenes empleada para un entorno interno es presentado en la Figura 5.8, en el cual se capturan las imágenes rotando la plataforma sobre su propio eje.

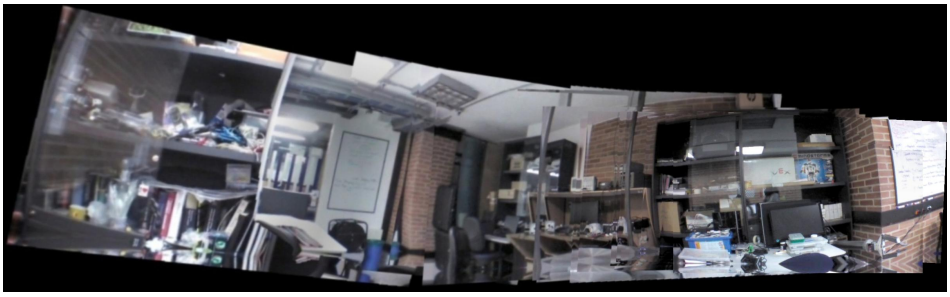


Figura 5.8: Ejemplo de imagen panorámica en escenario interno de resultado del algoritmo de mosaico en ambiente real, con homografía estimada a partir de las imágenes, compuesta por 27 imágenes.

No se presentan ejemplos del escenario de simulación porque la textura empleada para el piso en la herramienta de simulación, no contiene suficiente información y aparece en forma cíclica lo que no puede ser usado por el proceso de registro.

5.2.2. Resultados de ejecución en ambientes externos

En el Cuadro 5.4, se presenta la información de forma semejante de al Cuadro 5.3. Sin embargo, en este cuadro se incluye mayor información, dado que se tienen ejemplos con otros

detectores y descriptores. En especial, el usado en el estado del arte para la generación de mosaicos (detector y descriptor *SURF*) y el conjunto detector-descriptor que tiene la primera posición en el Cuadro 5.1 (detector: *SIFT* y descriptor: *BRIEF*).

Figura	*Tamaño*	*Mosaico*	*Detector*	*Descriptor*	*Homografía*
5.9	1652 x 1071	19	*FAST*	*BRISK*	Imagen
5.10	777 x 430	19	*FAST*	*BRISK*	Sensor
5.11	969 x 664	23	*FAST*	*BRISK*	Imagen
5.12	590 x 786	32	*SIFT*	*BRIEF*	Imagen
5.13	879 x 865	14	*SURF*	*SURF*	Imagen
5.14	561 x 1119	29	*FAST*	*BRISK*	Imagen
5.15	660 x 940	29	*FAST*	*BRISK*	Sensor
5.18	608 x 1267	26	*SIFT*	*BRIEF*	Sensor
5.17	1142 x 2205	26	*SURF*	*SURF*	Sensor
5.19e	696 x 872	41	*FAST*	*BRISK*	Sensor
5.16	498 x 383	17	*FAST*	*BRISK*	Sensor

Cuadro 5.4: Información mosaicos ambientes externos

En la primera columna del Cuadro 5.4 se tiene el número de figura a la que se hace referencia, en la segunda el tamaño en píxeles, en la tercera el número de mosaicos requeridos para llegar a la imagen presentada, en la cuarta columna el detector de puntos clave empleado, en la quinta columna el descriptor de características y en la sexta columna, indica la forma de obtener la matriz de homografía, en el caso de indicar *Sensor* se obtiene la matriz con la Ecuación 4.9 a partir de la información de los sensores embarcados filtrados mediante un FEK tal y como se presenta en el diagrama de bloques de la Figura 4.25 y en caso de indicar *Imagen* es porque la matriz de homografía es estimada a partir de la información de la imagen. Cada nuevo mosaico implica 2 imágenes nuevas, por ejemplo en el caso de tener 32 mosaicos se sabe que está compuesto de 64 imágenes nuevas y una imagen base, para un total de 65 imágenes.

Mapas obtenidos del ambiente de simulación en ambientes exterior.

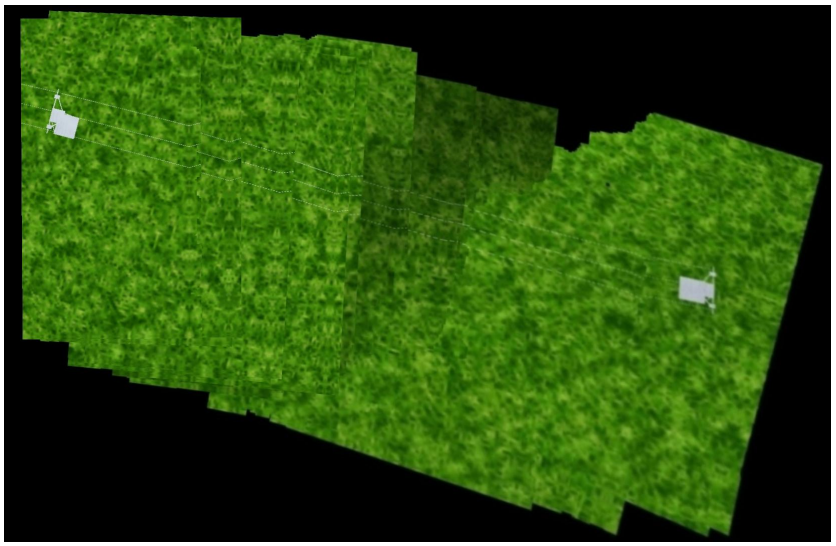


Figura 5.9: Ejemplo 1 en escenario externo del resultado del algoritmo de mosaico en ambiente de simulación a una altitud de un metro, con conjunto *FAST-BRISK* con homografía estimada a partir de las imágenes, compuesta por 39 imágenes.

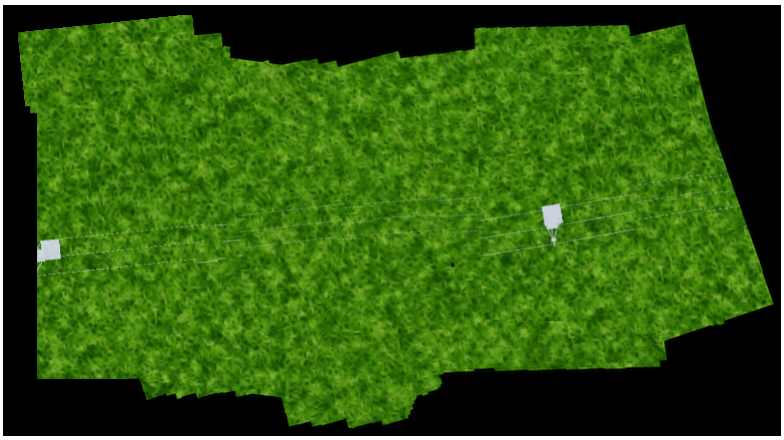


Figura 5.10: Ejemplo 2 en escenario externo del resultado del algoritmo de mosaico en ambiente de simulación a una altitud de un metro, con conjunto *FAST-BRISK* con homografía calculada con la información sensorial embarcada, compuesta por 39 imágenes.

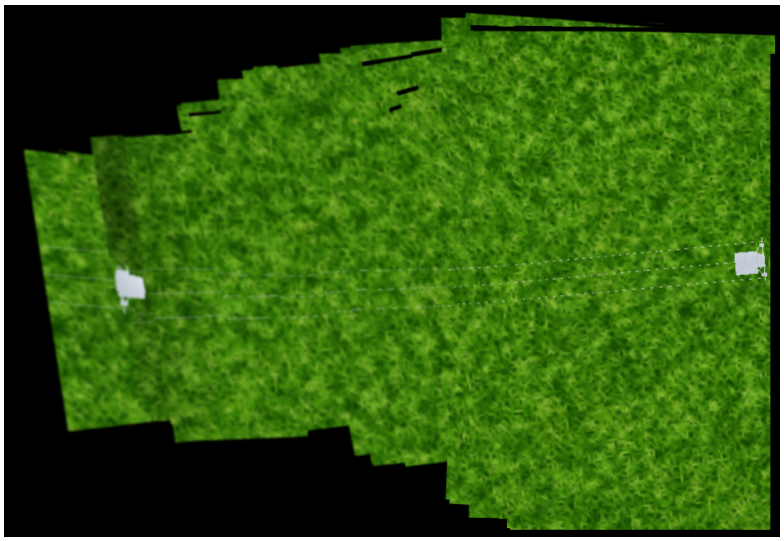


Figura 5.11: Ejemplo 3 en escenario externo del resultado del algoritmo de mosaico en ambiente de simulación a una altitud de un metro, con conjunto *FAST-BRISK* con homografía estimada a partir de las imágenes, compuesta por 47 imágenes.

Mapas en ambientes reales en el exterior, con diferentes detectores-descriptores y tipos de matrices de homografía.

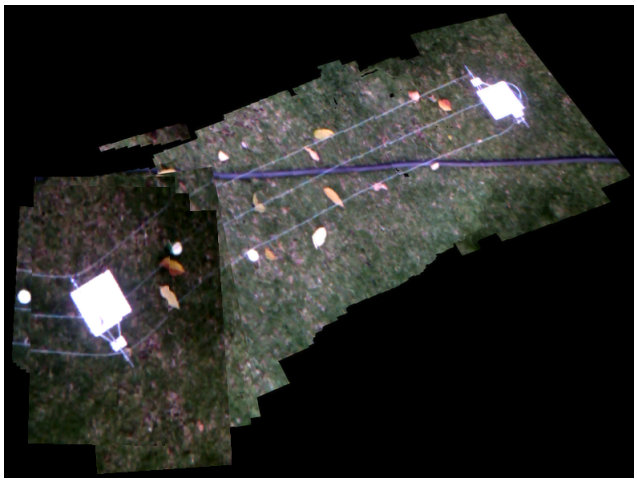


Figura 5.12: Ejemplo 1 en escenario externo del resultado del algoritmo de mosaico en ambiente real a una altitud de un metro, con conjunto *SIFT-BRIEF* con homografía estimada a partir de las imágenes, compuesta por 65 imágenes.

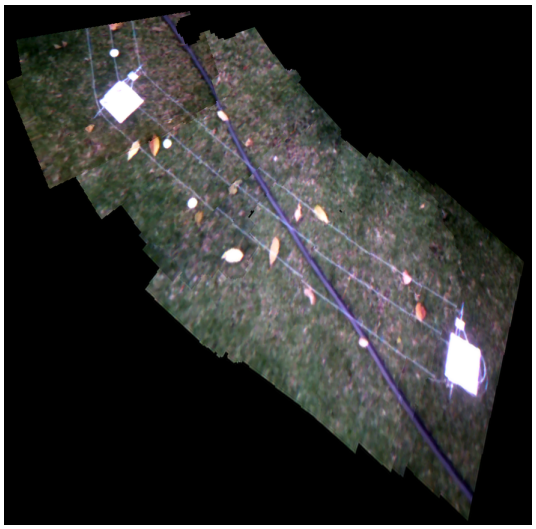


Figura 5.13: Ejemplo 2 en escenario externo del resultado del algoritmo de mosaico en ambiente real a una altitud de un metro, con conjunto *SURF-SURF* con homografía estimada a partir de las imágenes, compuesta por 29 imágenes.

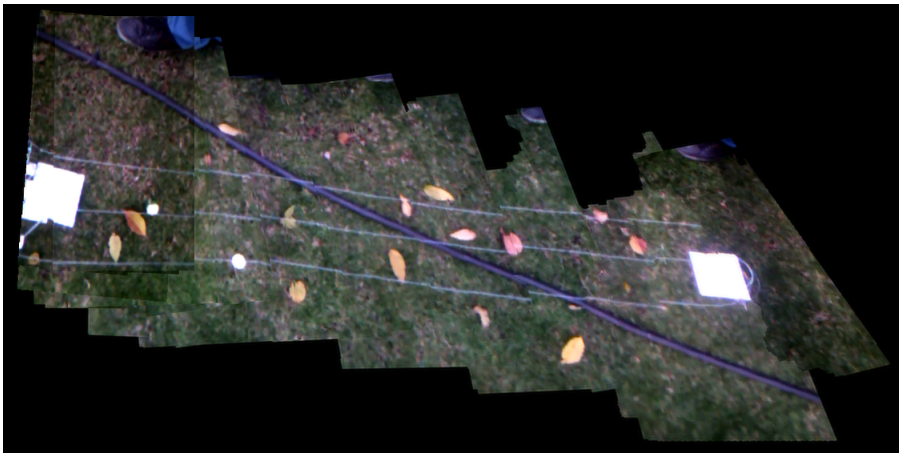


Figura 5.14: Ejemplo 3 en escenario externo del resultado del algoritmo de mosaico en ambiente real a una altitud de un metro, con conjunto *FAST-BRISK* con homografía estimada a partir de las imágenes, compuesta por 59 imágenes.

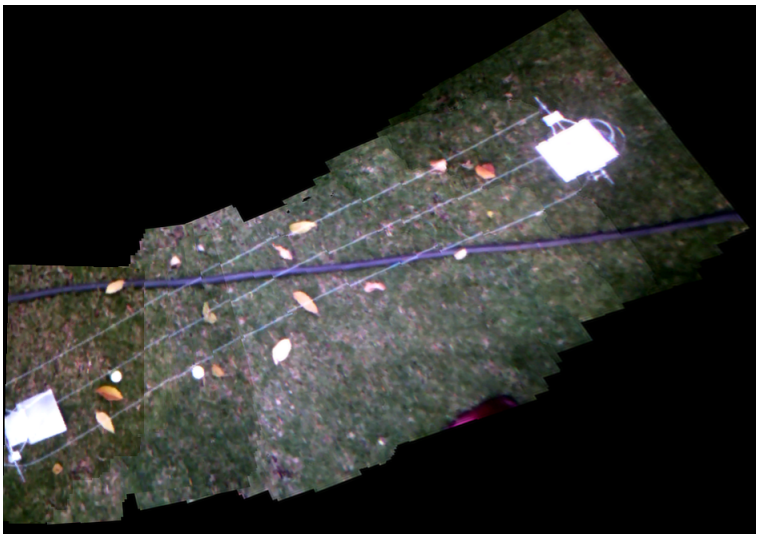


Figura 5.15: Ejemplo 4 en escenario externo del resultado del algoritmo de mosaico en ambiente real a una altitud de un metro, con conjunto *FAST-BRISK* con homografía calculada con la información sensorial embarcada,compuesta por 59 imágenes.

Figura 5.16: Ejemplo de mosaico en ambiente externo a una altitud de 5 metros, compuesta por 35 imágenes.

En la Figura 5.16, se tiene un ejemplo de imagen de mosaico a mayor altitud (5 metros), en comparación a las figuras previas donde la altitud solo fue de un metro. Lo que permitió, en menor número de imágenes tener el mapa de la zona, en detrimento de una menor relación de píxeles por metro.Un ejemplo de las imágenes de mosaico durante el proceso de

creación de mapas, es presentado en la Figura 5.19. En esta figura, se toma solo cada 10 nuevos mosaicos para poder ver el proceso de actualización del mapa. Sin embargo, durante la ejecución del desplazamiento se presentan al usuario en forma constante cada uno de las imágenes de mosaico obtenidas.Toda la información de posición, matrices de homografía, imágenes originales e imágenes de mosaico durante el proceso son almacenadas en el disco duro de la estación base.

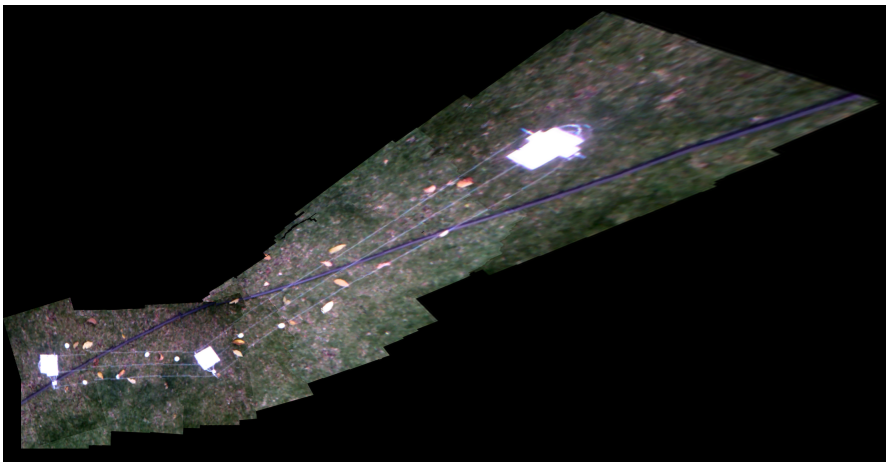


Figura 5.17: Ejemplo 6 en escenario externo del resultado del algoritmo de mosaico en ambiente real a una altitud de un metro, con conjunto *SURF-SURF* con homografía calculada con la información sensorial embarcada, compuesta por 53 imágenes.

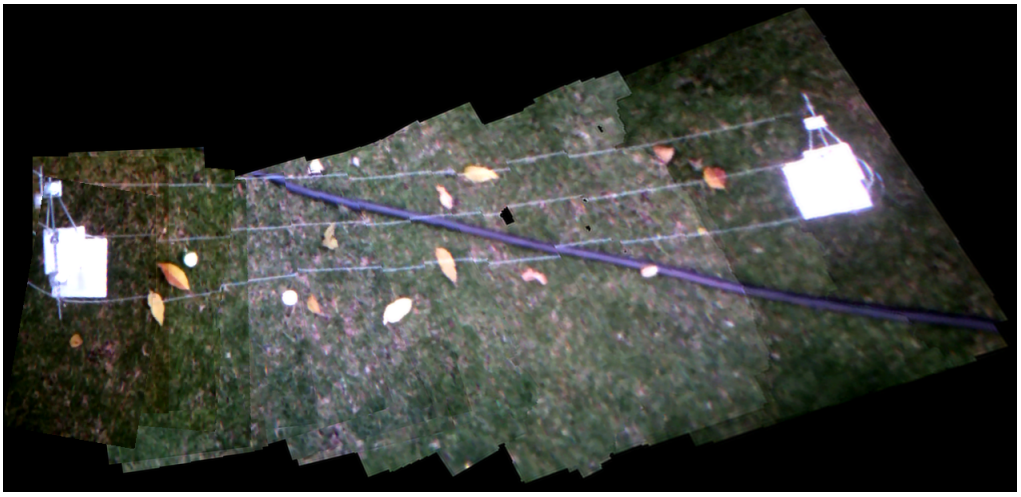
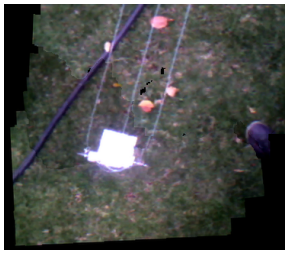


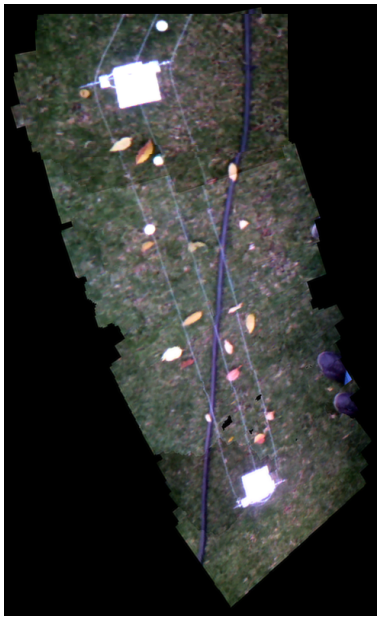
Figura 5.18: Ejemplo 5 en escenario externo del resultado del algoritmo de mosaico en ambiente real a una altitud de un metro, con conjunto *SIFT-BRIEF* con homografía calculada con la información sensorial embarcada, compuesta por 53 imágenes.

(a) Mosaico con 3 imágenes

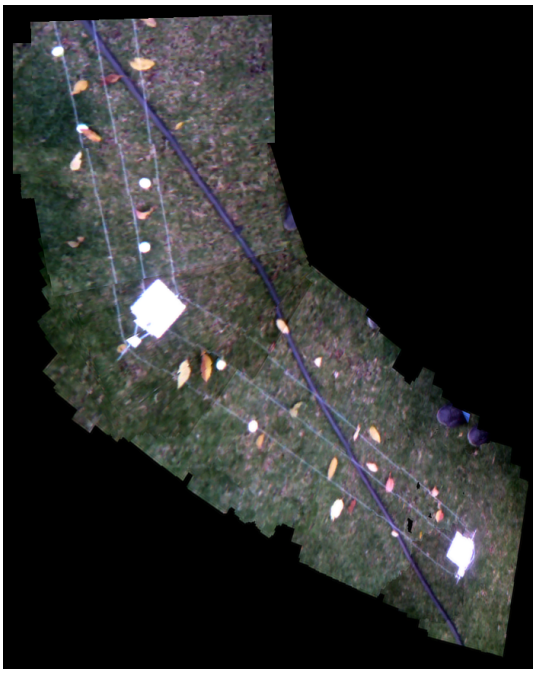


(b) Mosaico con 21 imágenes

(c) Mosaico con 41 imágenes



(d) Mosaico con 61 imágenes



(e) Mosaico con 83 imágenes

Figura 5.19: Ejemplo escenario externo a una altitud de un metro, con conjunto *FAST-BRISK* con homografía por la información sensorial.

Capítulo 6

Conclusiones y trabajos futuros

6.1. Conclusiones del trabajo de investigación

El algoritmo descrito cumple con la tarea de realizar un mapa de una zona deseada, empleando la técnica de visión artificial de mosaico de imágenes, con la integración de la información sensorial de una plataforma aérea no tripulada *UAV* comercial, que para el caso de este proyecto fue el quadrotor *ARDrone 2.0*. Con respecto a la información visual en la imagen final o mapa, se tiene que maneja la misma resolución. Además, cumple con el propósito que la persona disponga de mayor cantidad de información de la zona deseada.

La plataforma comercial *ARDrone 2.0* tiene los sensores necesarios para establecer la orientación y dos cámaras embarcadas, la primera ubicada en la parte inferior de la plataforma que fue empleada para realizar la captura de las imágenes y la segunda ubicada en la parte frontal con mayor resolución. El algoritmo de mosaico permite ser ejecutado en forma semejante con cada una de las cámaras de la plataforma, sin embargo, dada la baja calidad de imagen de la cámara ubicada en la parte inferior, la velocidad de desplazamiento que se puede lograr sobre la zona es menor, en comparación a la que se puede tener cuando se realiza el proceso de mosaico con una cámara de mejor calidad como por ejemplo la cámara frontal. Lo que implica que en la Ecuación 4.2 el valor de ρ debe ser cercano a 1, para el caso de la cámara ubicada en la parte inferior.

Otro elemento que permite incrementar la velocidad de desplazamiento, es la altitud de captura de las imágenes. Ya que la información que contiene la imagen capturada aumenta y pueden ser detectados más puntos clave, sin embargo, por la baja resolución de la cámara de la plataforma, la información visual pierde resolución y se reduce la cantidad de información entregada al usuario final.

El proceso de mosaico está compuesto por dos grandes bloques: el registro y la composición, las cuales se presentaron y detallaron en la sección 4.3.1. Para el primer bloque, se trabajó con el proceso de registro basado en características, para el cual se realizó la selección de un conjunto detector-descriptor de entre las técnicas más recientes del estado del arte [6, 7, 8, 9, 10]. Para el cual, se realizó un aporte al estado del arte generando una nueva metodología de evaluación y selección dependiente del escenario de captura de la información visual.

Para realizar la selección se requirió más de un criterio de evaluación, dado que el mejor caso de un solo criterio, puede estar entre los resultados más negativos de entre los otros criterios. Un ejemplo de esto, es el conjunto detector descriptor #28 (detector: *MSER*, descriptor: *FREAK*), cuyo tiempo de ejecución en el emparejador (*matcher*) es el menor, lo que implica que es mejor calificación de este criterio. Sin embargo, en los demás criterios seleccionados, esté conjunto hace parte de los resultados de menor rendimiento. Esto es reflejado en el resultado de la posición final presentada en la última columna del Cuadro 5.1, en el cual ocupa el último puesto entre los 28 conjuntos evaluados, los cuales se listan en el Cuadro 4.7.

La descripción teórica de criterios de evaluación como metodología de selección se presentan en la sección 5.2 y brevemente son: el criterio de repetibilidad, el cual permite la evaluación del detector, el criterio de MAP, que permite analizar el descriptor, el criterio de *confidence*, el cual también hace parte del algoritmo de mosaico para determinar el aporte de información de la imagen y un par de criterios de tiempo de ejecución, tenidos en cuenta por la característica de ejecución en tiempo real del proceso de mosaico. Estos permitieron seleccionar un conjunto detector-descriptor específicamente diseñado para ésta aplicación, mediante el uso de imágenes de los escenarios de interés.

El detector seleccionado fue *FAST,* el cual se encuentra entre los conjuntos de mayor valor de repetibilidad, lo cual es visible en las Figuras 7.5 y 7.6, lo que implica que la mayoría de la información detectada en una imagen A es detectable en una imagen B que contenga la misma información de la imagen A, lo que permite detectar los mismos puntos en escenas similares. Otra característica de *FAST*, es el corto tiempo de ejecución del conjunto detector-descriptor, lo que se puede confirmar en las Figuras 7.9 y 7.10, en las cuales el detector *FAST* reduce el tiempo de ejecución sin importar el descriptor (Ver desde #17 hasta #22).

Por otra parte, el descriptor seleccionado fue *BRISK* el cual se encuentra entre los descriptores de mayor *confidence* y en promedio entrega los mejores resultados en el criterio de evaluación *MAP* independientemente al detector empleado, esta afirmación fue realizada previamente en la evaluación presentada en [48], y es verificable con los resultados ilustrados en las Figuras 7.5 y 7.6.

Otra de las ventajas del descriptor *BRISK*, está en la cantidad de información, ya que puede ser representada en un menor espacio de memoria, en comparación a las técnicas *LoG y DoG*, estas técnicas requieren de un vector de características en el cual cada posición se encuentra en formato flotante, lo cual difiere en *BRISK* que maneja un formato binario y la representación de todo el descriptor puede hacerse en una sola cadena de bits. Lo que en conclusión, permite afirmar que descriptor *BRISK* se puede emplear en aplicaciones que requieran ejecución en tiempo de desplazamiento de una plataforma (tiempo real para este proyecto), y para aplicaciones que requieran ser embebidas en dispositivos embarcados, lo cual se proyecta como beneficio con al requerir menor espacio de memoria, al ser ejecutado.

Aunque la selección de un nuevo conjunto detector-descriptor, mediante un proceso cualitativo mejora la ejecución de la etapa de registro, no asegura que el proceso de mosaico sea siempre exitoso, dado que existe la segunda etapa en el algoritmo de mosaico, que proyecta las imágenes de entrada en la superficie de interés. Lo que implica, que el análisis cuantitativo para la selección del conjunto detector-descriptor no buscó tener una relación

directa con el análisis cualitativo de los mapas finales presentados en la sección 5.2.

A partir de las imágenes de mosaicos resultantes presentadas en la sección 5.2, puede verse que la homografía obtenida utilizando la información sensorial de la plataforma robótica, presenta una mejor estimación en los escenarios de simulación. En contraste con la homografía estimada a partir de las imágenes capturadas. Esto se debe a que el FEK, trabaja mucho mejor con el ruido de simulación, y también debido a la imposibilidad de generar texturas más complejas en el simulador, necesarias en el proceso de obtención de puntos en la etapa de registro. Caso contrario, los mapas generados en entornos reales presentan características similares. Sin embargo, en escenarios en los cuales existan problemas de iluminación o escenarios con carencia de texturas complejas, puede obtenerse un mejor mapa a partir de la homografía generada con los datos de los sensores de la plataforma.

Como segunda etapa del proceso de mosaico, se tiene la composición. Esta etapa es la de mayor desviación en el consumo de tiempo y procesamiento como se ve en la Figura 5.2, y es la encargada de proyectar las imágenes de entrada en la superficie en la cual se desea la imagen de mosaico.

Es posible emplear otros procesos de compensación y suavizado en la imagen de salida que son empleados usualmente en los procesos de obtención de imágenes panorámicas. Sin embargo, para el presente proyecto estas últimas correcciones hacen que la imagen de salida del mosaico, pierda información que es requerida en la primera fase del proceso de mosaico, ya que un proceso de filtrado sobre las imágenes, tiene como resultado la pérdida de información, que visualmente se traduce en perdida de la nitidez de las imágenes que fueron unidas previamente en el proceso de mosaico. Al retirar estos procesos de compensación y filtrado en la imagen, se reduce el tiempo de ejecución de la etapa de composición y de la totalidad del algoritmo, permitiendo que los tiempos de cada una de las etapas (registro y composición) tengan una duración semejante en promedio, como se ve en la Figura 5.2, además por el tiempo total de ejecución se puede emplear el algoritmo de mosaico en simultáneo con el desplazamiento de la plataforma aérea.

6.2. Trabajos futuros complementarios

Como trabajos futuros que se pueden desarrollar al presente proyecto se proponen dos:

1. Después de implementar el sistema de mosaico, se detectó que la etapa de composición es la que más tiempo requiere en el proceso de mosaico. Y el tiempo de ejecución, es una limitación de la velocidad de desplazamiento de la plataforma. Es por esto, que surge la necesidad de reducir el tiempo del proceso de composición aun más de lo que se logro reducir en el presente proyecto. Para ello, se sugiere implementar la etapa de composición como un proceso multitarea. Lo cual, es posible dado que algunos de los procesos de mayor tiempo de ejecución tienen las mismas variables de entrada y diferentes variables de salida.

2. En la Figura 6.1, se presenta el diagrama de bloques del presente proyecto modificado, con un bloque nuevo indicado en color verde, el cual permitiría que el sistema fuera

autónomo en la construcción del mapa de la zona deseada. Ese bloque denominado *TrayectoriaROS*, tiene como objetivo generar los puntos de desplazamiento de la plataforma a partir de la información de posición inicial, la región deseada y la altitud de desplazamiento. Aplicando las ecuaciones planteadas en el presente proyecto en la sección 2.4. Además, el bloque *TrayectoriaROS* tendría la tarea de controlar el seguimiento de la trayectoria y recalcular en caso de ser necesario. Como entrada interna del sistema, está la información sensorial previamente filtrada. A diferencia de la aplicación del proyecto actual, sería necesario incluir la información de *GPS* para el funcionamiento en entornos exteriores, esta información es necesaria para realimentar el control de posición requerido para el seguimiento de trayectoria, y para el caso de las aplicaciones en entornos interiores, se requiere instalar una cámara de bajo peso en la parte inferior del ARDrone, con el propósito de emplear la aplicación de odometría visual implementada previamente en ROS [37], la cual funciona con la cámara frontal de la plataforma.

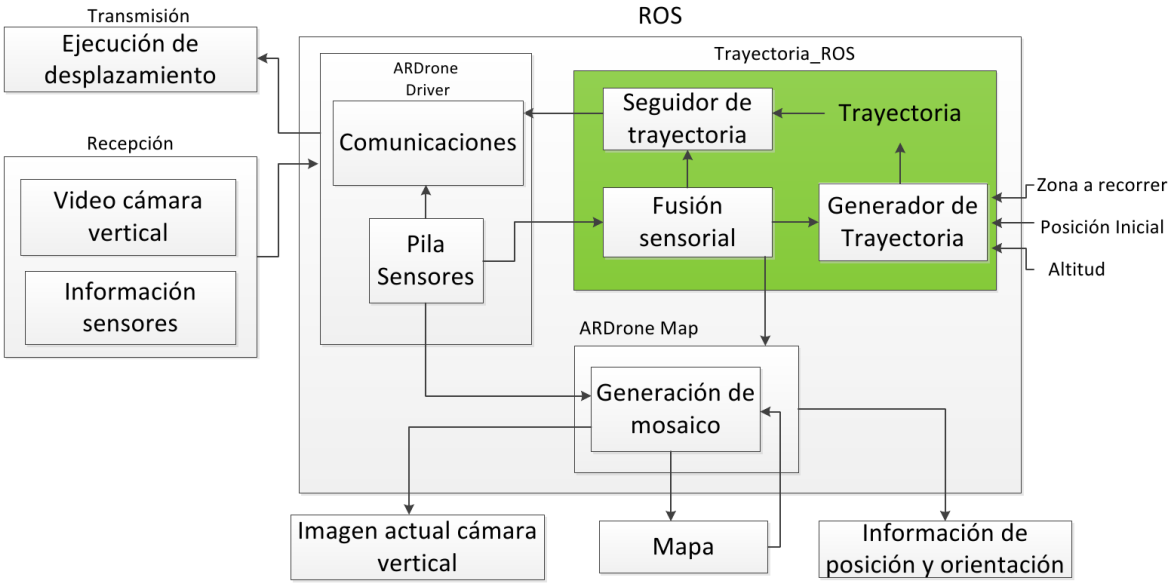


Figura 6.1: Diagrama de bloques de trabajo complementario para trabajo futuro. En color verde el nuevo bloque de trayectoria autónoma.

Bibliografía

[1] P.-J. Bristeau, F. Callou, D. Vissière, and N. Petit, "The Navigation and Control technology inside the AR.Drone micro UAV," 2011.

[2] A. Barrientos Cruz, J. Colorado Montaño, J. d. Cerro Giner, A. Martinez, W. Coral Cuellar, and J. R. Pereira Valente, "Aerial Remote Sensing in Agriculture: A Practical Approach to Area Coverage and Path Planning for Fleets of Mini Aerial Robots," *Journal of Field Robotics*, vol. 28, pp. 667–689, jun 2011.

[3] Y. Wang, R. Fevig, and R. Schultz, "Super-resolution mosaicking of UAV surveillance video," in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, 2008.

[4] I. Mondragon, *On-board visual control algorithms for unmannaed aerial vehicles*. PhD thesis, Escuela Tecnica Superior de Ingenieros Industriales, 2011.

[5] N. Dijkshoorn and A. Visser, "Simultaneous localization and mapping with the AR.Drone," Master's thesis, Universiteit Van Amsterdam, July 2012.

[6] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust Wide Baseline Stereo from Maximally Stable Extremal Regions," in *Procedings of the British Machine Vision Conference 2002*, pp. 36.1–36.10, British Machine Vision Association, 2002.

[7] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision–ECCV 2006*, pp. 430–443, 2006.

[8] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*, pp. 2564–2571, IEEE, Nov. 2011.

[9] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary Robust invariant scalable keypoints," in *2011 International Conference on Computer Vision*, pp. 2548–2555, IEEE, Nov. 2011.

[10] A. Alahi, R. Ortiz, and P. Vandergheynst, "FREAK: Fast Retina Keypoint," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, (Providence, RI), pp. 510–517, IEEE, June 2012.

[11] C. Pohl and J. L. Van Genderen, "Review article Multisensor image fusion in remote sensing: Concepts, methods and applications," *International Journal of Remote Sensing*, vol. 19, no. 5, pp. 823–854, 1998.

[12] F. Caballero, L. Merino, J. Ferruz, and A. Ollero, "Unmanned Aerial Vehicle Localization Based on Monocular Vision and Online Mosaicking," *Journal of Intelligent and Robotic Systems*, vol. 55, pp. 323–343, 2009.

[13] J. Engel, D. Cremers, and J. Sturm, "Autonomous Camera-Based Navigation of a Quadrocopter," Master's thesis, Der technischen Universitat Munchen, dic 2011.

[14] G. Nützi, S. Weiss, D. Scaramuzza, and R. Siegwart, "Fusion of IMU and Vision for Absolute Scale Estimation in Monocular SLAM.," *Journal of Intelligent and Robotic Systems*, vol. 61, no. 1-4, pp. 287–299, 2011.

[15] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments.," *J. Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011.

[16] A. Camargo, R. Schultz, Y. Wang, R. Fevig, and Q. He, "GPU-CPU implementation for super-resolution mosaicking of Unmanned Aircraft System (UAS) surveillance video," in *Image Analysis Interpretation (SSIAI), 2010 IEEE Southwest Symposium on*, pp. 25–28, may 2010.

[17] Y. Wang, R. Schultz, and R. Fevig, "Sensor fusion method using GPS/IMU data for fast UAV surveillance video frame registration," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, 2009.

[18] V. G. Juan, L. A. Angel, and J. T. Montserrat, "Generación de Trayectorias de Curvatura Continua para el Seguimiento de Líneas basado en Visión Artificial ," Master's thesis, Universidad Politecnica Valencia, dec 2010.

[19] H. E. Cuchango and o. I. S. Esmeral, "Propuesta de un algoritmo para la planeación de trayectorias de robots moviles empleando campos potenciales y enjambres de particulas activas Brownianas," Master's thesis, Universidad Nacional de Colombia, 2011.

[20] H. Ergezer and K. Leblebicioglu, "Path Planning for UAVs for Maximum Information Collection," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 49, pp. 502–520, jan. 2013.

[21] M. Brown and D. Lowe, "Automatic Panoramic Image Stitching using Invariant Features," *International Journal of Computer Vision*, vol. 74, pp. 59–73, 2007.

[22] R. Szeliski, "Image alignment and stitching: a tutorial," tech. rep., One Microsoft Way Redmond, WA 98052, Dec. 2006.

[23] D.-H. Kim, Y.-I. Yoon, and J.-S. Choi, "An efficient method to build panoramic image mosaics," *Pattern Recognition Letters*, vol. 24, no. 14, pp. 2421–2429, 2003.

[24] Y. Wang, R. Schultz, and R. Fevig, "Panorama recovery from noisy UAV surveillance video," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pp. 1285–1288, Apr. 2009.

[25] V. Cani, E. Pastor, and C. Barrado, "Image Stitching for UAV remote sensing application," Master's thesis, Castelldefels of Universitat Politécnica de Catalunya, Jan. 2011.

[26] H. Bülow and A. B. 0002, "Fast and robust photomapping with an Unmanned Aerial Vehicle (UAV).," in *IROS*, pp. 3368–3373, IEEE, 2009.

[27] C. Harris and M. Stephens, "A combined corner and edge detector," in *In Proc. of Fourth Alvey Vision Conference*, pp. 147–151, 1988.

[28] D. Lowe, "Object recognition from local scale-invariant features," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2, pp. 1150–1157 vol.2, 1999.

[29] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008. Similarity Matching in Computer Vision and Multimedia.

[30] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," in *Computer Vision – ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), vol. 6314 of *Lecture Notes in Computer Science*, pp. 778–792, Springer Berlin Heidelberg, 2010.

[31] T. Lindeberg, "Scale-space theory: A basic tool for analysing structures at different scales," *Journal of Applied Statistics*, pp. 224–270, 1994.

[32] J. L. Crowley and A. C. Parker, "A Representation for Shape Based on Peaks and Ridges in the Difference of Low-Pass Transform," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 2, pp. 156–170, 1984.

[33] T. Lindeberg, "Detecting Salient Blob-Like Image Structures and Their Scales with a Scale-Space Primal Sketch: A Method for Focus-of-Attention," *International Journal of Computer Vision*, vol. 11, pp. 283–318, 1993.

[34] P. L. Rosin, "Measuring Corner Properties.," *Computer Vision and Image Understanding*, vol. 73, no. 2, pp. 291–307, 1999.

[35] D. Scaramuzza and F. Fraundorfer, "Visual Odometry, Part I: The first 30 years and fundamentals," *IEEE Robotics & Automation Magazine*, pp. 80–92, Dec. 2011.

[36] D. Scaramuzza and F. Fraundorfer, "Visual Odometry, Part II: Matching, Robustness, Optimization and applications," *IEEE Robotics & Automation Magazine*, pp. 78–90, June 2012.

[37] J. Engel, J. Sturm, and D. Cremers, "Camera-Based Navigation of a Low-Cost Quadrocopter," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[38] J. Engel, J. Sturm, and D. Cremers, "Accurate Figure Flying with a Quadrocopter Using Onboard Visual and Inertial Sensing," in *Proc. of the Workshop on Visual Control of Mobile Robots (ViCoMoR) at the IEEE/RJS International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[39] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2008.

[40] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME-Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[41] J. Florez, F. Calderon, and C. Parra, "Video stabilization taken with a snake robot," in *Image, Signal Processing, and Artificial Vision (STSIVA), 2013 XVIII Symposium of*, pp. 1–5, Sept 2013.

[42] M. Muja and D. G. Lowe, "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration," in *International Conference on Computer Vision Theory and Application VISSAPP'09)*, 2009.

[43] M. Muja and D. G. Lowe, "Fast Matching of Binary Features," in *Computer and Robot Vision (CRV)*, pp. 404–410, 2012.

[44] S. Zhao, F. Lin, K. Peng, B. Chen, and T. Lee, *Homography-based Vision-aided Inertial Navigation of UAVs in Unknown Environments*. Guidance, Navigation, and Control and Co-located Conferences, American Institute of Aeronautics and Astronautics, aug 2012. doi:10.2514/6.2012-5033.

[45] B. M. M. de Oliveira, "Quadrotor Stabilization using Visual Servoing ," Master's thesis, may 2013.

[46] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. 27, pp. 1615–1630, Oct. 2005.

[47] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 2003.

[48] A. Canclini, M. Cesana, A. Redondi, M. Tagliasacchi, J. Ascenso, and R. Cilla, "Evaluation of low-complexity visual feature detectors and descriptors," in *Digital Signal Processing (DSP), 2013 18th International Conference on*, pp. 1–7, July 2013.

[49] R. E. Walpole, R. H.Myers, S. L. Myers, and K. Ye, *Probabilidad & Estadística para ingeniería y ciencias* , vol. 82. Pearson, 2007.

[50] J. Kelly, S. Saripalli, and G. Sukhatme, "Combined Visual and Inertial Navigation for an Unmanned Aerial Vehicle," in *Field and Service Robotics* (C. Laugier and R. Siegwart, eds.), vol. 42 of *Springer Tracts in Advanced Robotics*, pp. 255–264, Springer Berlin Heidelberg, 2008.

[51] G. Welch and G. Bishop, *An introduction to the Kalman Filter*. ACM, 2001.

Capítulo 7

Anexos

7.1. Anexo: Instalación.

Script 1 Instalación de ROS Fuerte

ROS Fuerte

– Setup your sources.list Ubuntu 12.04 (Precise)

sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu precise main" > /etc/apt/sources.list.d/ros-latest.list'

– Set up your keys

wget http://packages.ros.org/ros.key -O - | sudo apt-key add -

– Installation (Punto de instalación)

sudo apt-get update

sudo apt-get install ros-fuerte-desktop-full

– Environment setup (Incluir al entorno la variable de ejecución de ROS)

echo "source /opt/ros/fuerte/setup.bash" >> ~/.bashrc

. ~/.bashrc

– Getting rosinstall and rosdep (Otros elementos de ROS, para el correcto funcionamiento de Python)

sudo apt-get install python-rosinstall python-rosdep

Script 2 Instalación paquetes para ARDrone de ROS

Herramientas para el ARDrone

– Ir a la carpeta de paquetes de ejecución de ROS Fuerte

cd /opt/ros/fuerte/stacks

– Drivers para el ARDrone:

– Se requiere verificar que se tiene instalada la librería liblapack-dev y libblas-dev para correcto funcionamiento

– Paquete: ardrone_autonomy

sudo git clone git://github.com/AutonomyLab/ardrone_autonomy.git

– Se requiere dar permisos a las carpetas por su localización

sudo chmod -R 777 ardrone_autonomy

export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:'pwd'/ardrone_autonomy

cd ardrone_autonomy

sudo ./build_sdk.sh

rosmake

– Paquete: tum_ardrone

cd ..

sudo git clone git://github.com/tum-vision/tum_ardrone.git tum_ardrone

export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:'pwd'/tum_ardrone

cd tum_ardrone

rosmake tum_ardrone

– Drivers para funcionamiento de ARDrone con control de PlayStation 3 - JoyStick.

cd ..

svn checkout https://svncvpr.informatik.tu-muenchen.de/cvpr-ros-pkg/trunk/ardrone_helpers

export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:‘pwd‘/ardrone_helpers rosmake ardrone_joystick rosmake joy

– Herramienta de simulación que emplean Gazebo

– Paquete: tum_simulator

git clone https://github.com/tum-vision/tum_simulator.git

export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:‘pwd‘/tum_simulator

rosmake cvg_sim_gazebo_plugins

rosmake message_to_tf

Script 3 Instalación previa a OpenCV

Previo a la instalación del paquete de OpenCV

– Verificar que se tienen instalados los pre-requisitos de OpenCV

sudo apt-get install build-essential libgtk2.0-dev cmake cmake-gui libtbb2 libtbb-dev libavformat-dev libavcodec-dev libavfilter-dev libswscale-dev x264 libx264-dev subversion

– Verificar que se tienen las dependencias de OpenCV

sudo apt-get install build-essential cmake pkg-config libpng12-0 libpng12-dev libpng++-dev libpng3 libpnglite-dev zlib1g-dbg zlib1g zlib1g-dev libjasper-dev libjasper-runtime libjasper1 pngtools libtiff4-dev libtiff4 libtiffxx0c2 libtiff-tools libjpeg8 libjpeg8-dev libjpeg8-dbg ffmpeg libavcodec-dev libavformat-dev libgstreamer0.10-0-dbg libgstreamer0.10-0 libgstreamer0.10-dev libxine1-ffmpeg libxine-dev libxine1-bin libunicap2 libunicap2-dev libdc1394-22-dev libdc1394-22 libdc1394-utils swig libv4l-0 libv4l-dev python-numpy

Script 4 Instalación de OpenCV

Instalación OpenCV
cd opencv-2.4.8
mkdir build
cd build
– Abriendo con cmake-gui use el compilador UNIX por defecto, de click en configure, seleccione lo que requiere luego en configure por último generate, preferiblemente dos veces también, cuando termine, cierre la ventana
cmake-gui ..
– Empleando 4 -j4 cores para compilar, puede durar un rato
make -j4
– Fin de la compilación, instalar
sudo make install

7.2. Anexo 2: Fusión Sensorial

Para realizar la fusión sensorial se opta por aplicar un filtro de Kalman [40], ya que este algoritmo aplica un análisis de datos óptimo recursivo. Óptimo porque minimiza un criterio determinado y porque incorpora toda la información que se le suministra para determinar el filtrado. Recursivo porque no precisa mantener los datos previos, lo que facilita su implementación en sistemas de procesado en tiempo real. En el estado del arte se pueden encontrar diferentes aplicaciones de fusión sensorial en los cuales se aplica un filtro de Kalman, en especial el caso del filtro extendido [13, 14, 50], que incluye la información no lineal del sistema. A continuación se presenta el desarrollo teórico del filtro de Kalman discreto y de su versión extendida [51].

7.2.1. Filtro de Kalman discreto

El filtro de Kalman se enfoca en el problema general de tratar de estimar el estado de $x \in \Re^n$ de un proceso de control en tiempo discreto gobernado por una ecuación lineal de diferencias estocástica.

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \tag{7.1}$$

con un observador $z \in \Re^m$ que es:

$$z_k = Hx_k + v_k \tag{7.2}$$

Las variables aleatorias w_k y v_k representan el ruido del proceso y de la observación respectivamente. Se asume que son independientes entre ellas y con procesos blancos con función de distribución normal.

$$\begin{aligned} p(w) &\sim N(0,Q) \\ p(v) &\sim N(0,R) \end{aligned} \tag{7.3}$$

En la práctica, la matriz de covarianza del ruido del proceso Q y la matriz de covarianza del ruido de la observación R pueden cambiar con el paso del tiempo o medida, sin embargo aquí se asumen constantes.

La matriz A_{nxn} relaciona el estado en el tiempo k con el estado en el tiempo $k+1$. La matriz B_{nx1} relaciona la entrada de control u con el estado x. Y la matriz H_{nxm} relaciona el estado con la media z_k.

Se define como estimador del estado a posteriori $\hat{x_k}$ y al estimador a priori $\hat{x}_k^-$, luego se incluyen además los errores a posteriori y a priori con sus respectivas covarianzas.

$$\begin{aligned} e_k &= x_k - \hat{x}_k \quad , \quad P_k = E\left[e_k e_k^T\right] \\ e_k^- &= x_k - \hat{x}_k^- \quad , \quad P_k^- = E\left[e_k^- e_k^{-T}\right] \end{aligned} \tag{7.4}$$

Partiendo de los anteriores y con la ecuación que permita calcular el estado a posteriori como una combinación lineal del estado a priori, Ecuación 7.5. Se obtienen las ecuaciones del filtro de Kalman.

$$\hat{x_k} = \hat{x}_k^- + K\left(z_k - H_k \hat{x}_k^-\right) \tag{7.5}$$

A la diferencia en la Ecuación 7.5, se le denomina usualmente innovación de la medida o simplemente residuo y refleja la discrepancia entre la predicción de la medida y la observación actual. La matriz K_{nxm} llamada ganancia de Kalman o factor de mezcla establece la cantidad de influencia del error entre nuestra estimación y la medida.

$$K_k = P_k^- H^T \left(H P_k^- H^T + R\right)^{-1} \tag{7.6}$$

Siendo P_k^- el estimador de la covarianza del error a priori y R_k la covarianza del error medido. Se encuentra que si R_k se aproxima a 0, la ganancia ponderará el residuo con mayor peso. Por el contrario, cuando P_k^- se aproxima a 0, la ganancia ponderará menos el residuo.

Otra forma más intuitiva de ver la ponderación de K, es que cuando la covarianza del error de medida R_k se aproxime a 0, tendremos más confianza en la observación actual z_k, mientras que la medida predicha $H_k \hat{x}_k^-$ perderá confianza en la misma medida. Por otra parte, cuando el estimador de la covarianza del error a priori P_k^- se aproxime a 0 se perderá confianza en la medida z_k y la de la medida predicha $H_k \hat{x}_k^-$ se incrementará.

El filtro de Kalman estima variables de estado en un proceso con realimentación (ver Figura 7.1[1]). Calcula el estado del proceso en algún instante y entonces obtiene información de la medida. Por tanto, las ecuaciones del filtro se pueden clasificar en dos tipos: actualización del tiempo y actualización de las medidas. Las primeras son responsables de proyectar hacia el futuro los estimadores del estado actual y de la covarianza del error, para obtener los estimadores a priori del siguiente estado.

[1]Imagen tomada de [51]

Las ecuaciones de actualización de las medidas son responsables de la realimentación, incorporando una nueva medida a los estimadores a priori para obtener unos estimadores a posteriori mejorados.

$$\begin{aligned} \hat{x}_k^- &= A\hat{x}_{k-1} + Bu_k \\ P_k^- &= AP_{k-1}A^T + Q \end{aligned} \tag{7.7}$$

Y las ecuaciones de actualización del tiempo pueden ser interpretadas como ecuaciones de predicción, mientras que las de actualización de la medida pueden pensarse como ecuaciones de corrección.

$$\begin{aligned} K_k &= P_k^- H^T \left(HP_k^- H^T + R_k\right)^{-1} \\ \hat{x_k} &= \hat{x}_k^- + K\left(z_k - H_k\hat{x}_k^-\right) \\ P_k &= \left(I - K_k H_k\right) P_k^- \end{aligned} \tag{7.8}$$

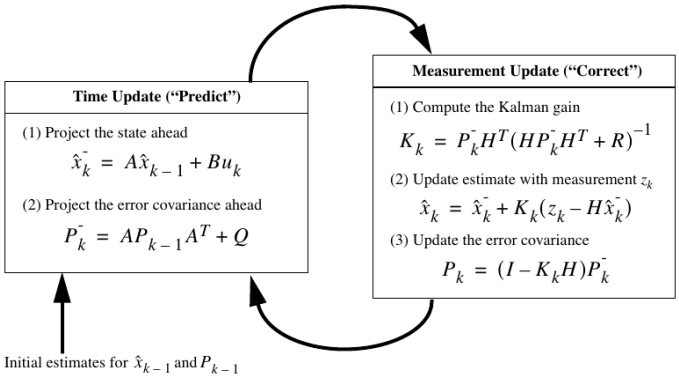


Figura 7.1: Proceso del filtro de Kalman

7.2.2. Filtro Extendido de Kalman (FEK)

Según lo explicado en la sección anterior el filtro de Kalman se ocupa del problema general de intentar estimar el estado $x \in \Re^n$ de un proceso controlado en tiempo discreto que es gobernado por una ecuación lineal de diferencias estocástica. Pero que sucede cuando el proceso a ser estimado y/o la relación de medida para el proceso es no lineal. Un filtro Kalman que aplica linealización en la media y en la covarianza actuales se denomina filtro extendido de Kalman.

Mediante algo semejante a una serie de Taylor se puede linealizar la estimación alrededor del estimador actual usando las derivadas parciales de las funciones de proceso y medida para calcular estimaciones aún en el lado de las relaciones no lineales. De esta forma se modifican las ecuaciones presentadas en la sección previa. Se asume nuevamente que el proceso tiene un vector de estado $x \in \Re^n$, pero que este proceso es ahora gobernado por una ecuación en diferencias estocástica no lineal.

$$x_k = f(x_{k-1}, u_k, w_{k-1}) \tag{7.9}$$

Con un observador:

$$z_k = h(x_k, v_k) \tag{7.10}$$

Las variables aleatorias w_k y v_k siguen representando el ruido del proceso y de la medida. Sin embargo en este caso ya se cuentan con las funciones no lineales f y h que relacionan el estado actual con los estados previos.

En la práctica se desconocen los valores individuales del ruido para cada paso del tiempo. Sin embargo se puede aproximar el vector de estado y del observador como se indica en 7.11.

$$\tilde{x}_k = f(\hat{x}_{k-1}, u_k, 0) \qquad \& \qquad \tilde{z}_k = h(\tilde{x}_k, v_k) \tag{7.11}$$

Es importante mencionar que un defecto fundamental del FEK es que las distribuciones (o las densidades en el caso continuo) de las variables aleatorias ya no son normales después de sufrir la transformación no lineal. El FEK es un simplemente un estimador de estado que solo aproxima la regla de Bayes óptimamente por linealización.

Para estimar el proceso sin no linealidades, se comienza re-escribir las ecuaciones 7.11.

$$x_k \approx \tilde{x}_k + A(x_{k-1} - \hat{x}_{k-1}) + W w_{k-1} \tag{7.12}$$

$$z_k \approx \tilde{z}_k + H(x_k - \tilde{x}_k) + V v_k \tag{7.13}$$

donde:

- x_k y z_k son los vectores actuales del estado y la medida.
- $\tilde{x}_k$ y $\tilde{z}_k$ son los vectores aproximados de los estados y la medida de la Ecuación 7.11.
- $\hat{x}_k$ es el estado estimado a posteriori en el tiempo k.
- Las variables aleatorias w_k y v_k representan el ruido como en la Ecuación 7.3.
- A es la matriz Jacobiana de las derivadas parciales de f con respecto a x, lo cual es: $A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_{k-1}, u_k, 0)$.
- W es la matriz Jacobiana de las derivadas parciales de f con respecto a w, lo cual es: $W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}(\hat{x}_{k-1}, u_k, 0)$.
- H es la matriz Jacobiana de las derivadas parciales de h con respecto a x, lo cual es: $H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\tilde{x}_k, 0)$.
- V es la matriz Jacobiana de las derivadas parciales de h con respecto a w, lo cual es: $V_{[i,j]} = \frac{\partial h_{[i]}}{\partial w_{[j]}}(\tilde{x}_k, 0)$.

Observe que en las notaciones de las ecuaciones anteriores no se utiliza el subíndice del paso en el tiempo k con los Jacobianos A, W, H, V, aunque ellos, en efecto, son diferentes en cada paso del tiempo.

Ahora se define una nueva notación para el error de predicción.

$$\tilde{e}_{x_k} \equiv x_k - \tilde{x}_k \tag{7.14}$$

y para el error de medida residual,

$$\tilde{e}_{z_k} \equiv z_k - \tilde{z}_k \tag{7.15}$$

Es necesario recordar que en la práctica no se tiene acceso a x_k en la Ecuación 7.14, el cual es el actual vector de estado, es decir la cantidad que se intenta estimar. Por otra parte, se tiene acceso a z_k en la Ecuación 7.15, esta es la medida actual que se usa para estimar x_k. Usando la Ecuación 7.14 y la Ecuación 7.15 se pueden escribir las ecuaciones que gobiernan los procesos de error como:

$$\tilde{e}_{x_k} \approx A\left(x_{k-1} - \hat{x}_{k-1}\right) + \varepsilon_k \qquad \& \qquad \tilde{e}_{z_k} \approx H\tilde{e}_{x_k} + \eta_k \tag{7.16}$$

donde ε_k y η_k representan nuevas variables aleatorias independientes con media cero y matrices de covarianza WQW^T y VRV^T, con Q y R como en 7.3.

Note que las ecuaciones en 7.16 son lineales y esto se asemeja a las ecuaciones del proceso y la medición, ecuaciones 7.1 y 7.2 del filtro Kalman discreto. Esto motiva a utilizar el residuo de la medida actual $\tilde{e}_{z_k}$ en la Ecuación 7.15 y un segundo (hipotético) Filtro Kalman para estimar el error de la predicción $\tilde{e}_{x_k}$ dado por la primera ecuación de 7.16. Esta estimación llamada $\hat{e}_k$, luego se puede usar con la Ecuación 7.14 para obtener el estado estimado a posteriori estimado para el proceso no lineal original. De la siguiente forma:

$$\hat{x}_k = \tilde{x}_k + \hat{e}_k \tag{7.17}$$

Las variables aleatorias de la Ecuación 7.16, tienen aproximadamente las siguientes funciones de distribución.

$$\begin{array}{rcl} p\left(\tilde{e}_{x_k}\right) & \sim & N\left(0, E\left[\tilde{e}_{x_k}\tilde{e}_{x_k}^T\right]\right) \\ p\left(\varepsilon_k\right) & \sim & N\left(0, WQW^T\right) \\ p\left(\eta_k\right) & \sim & N\left(0, VRV^T\right) \end{array} \tag{7.18}$$

Dadas estas aproximaciones y dejando el valor previsto de $\hat{e}_k$ a cero, la ecuación del filtro de Kalman usada para estimar $\hat{e}_k$ es

$$\hat{e}_k = K_k\tilde{e}_{z_k} \tag{7.19}$$

Sustituyendo en la Ecuación 7.19 en 7.17 se ve que no es necesario el segundo (hipotético) filtro de Kalman.

$$\begin{array}{rcl} \hat{x}_k & = & \tilde{x}_k + K_k\tilde{e}_{z_k} \\ & = & \tilde{x}_k + K_k\left(z_k - \tilde{z}_k\right) \end{array} \tag{7.20}$$

Finalmente el conjunto completo de ecuaciones para el filtro de Kalman extendido se muestran en las ecuaciones 7.21 y 7.22. Note que se ha sustituido $\hat{x}_k^-$ por $\tilde{x}_k$ para ser consistentes con la notación anterior del super índice '-', para el estado a priori y que ahora se ha colocado un sub índice k a los Jacobianos A, W, H y V. Adicionalmente en la Figura 7.2[51] se presenta el ciclo de actualización.

Las ecuaciones de actualización en el tiempo son:

$$\begin{aligned} \hat{x}_k^- &= f(\hat{x}_{k-1}, u_k, 0) \\ P_k^- &= A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T \end{aligned} \tag{7.21}$$

Las ecuaciones de actualización de medida son:

$$\begin{aligned} K_k &= P_k^- H_k^T \left(H_k P_k^- H_k^T + V_k R_k V_k^T\right)^{-1} \\ \hat{x}_k &= \hat{x}_k^- + K_k\left(z_k - h(\hat{x}_k^-, 0)\right) \\ P_k &= (I - K_k H_k) P_k^- \end{aligned} \tag{7.22}$$

Al igual que con el filtro de Kalman discreto la ecuación de actualización 7.22 corrige la estimación del estado y de la covarianza con la medida z_k.

Una característica importante del Filtro Extendido de Kalman es que el Jacobiano H_k en la ecuación para la ganancia K_k sirve para propagar o magnificar sólo el componente relevante de la información de la medida. Por ejemplo, si no existe un mapeo uno a uno entre la medida z_k y el estado a través de h, el Jacobiano H_k afecta la ganancia de Kalman de modo que sólo magnifica la porción del residuo $z_k - h(\hat{x}_k^-, 0)$ que afecta al estado. Por supuesto que si sobre todas las medidas no existe un mapeo uno a uno entre la medida z_k y el estado a través de h, entonces el filtro diverge rápidamente. En este caso el proceso es ***no observable.***

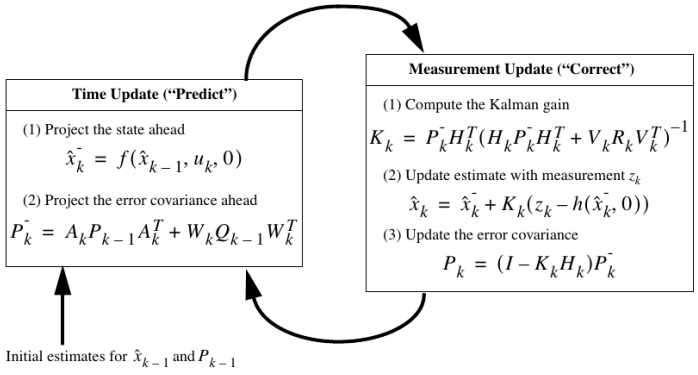


Figura 7.2: Proceso del Filtro Extendido de Kalman

7.3. Anexo 3: Información gráfica de la estadística de los resultados con los evaluadores de los conjuntos detector-descriptor.

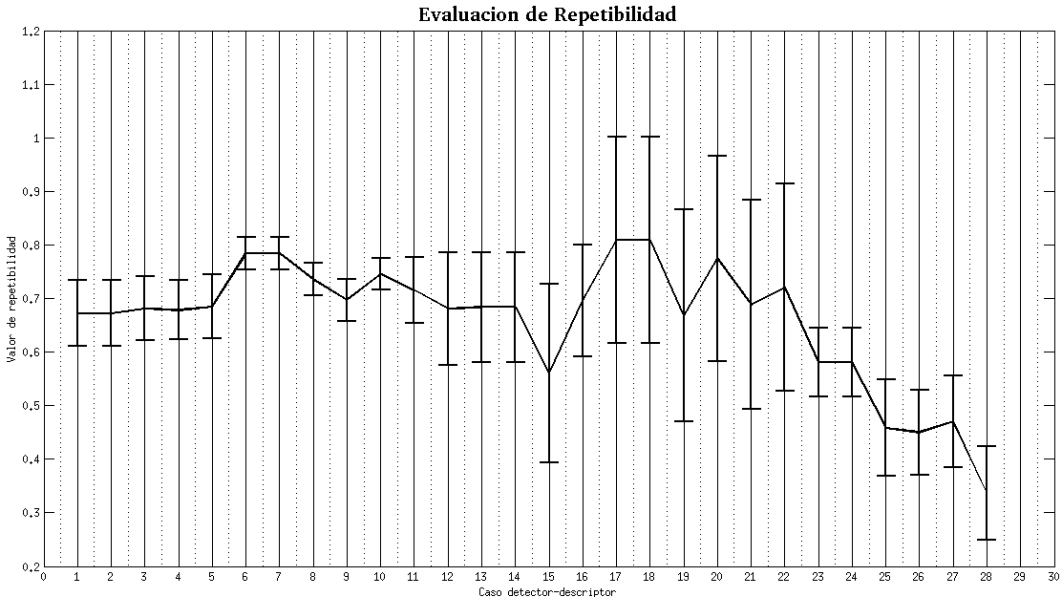


Figura 7.3: Parámetro de evaluación de $repetibilidad(Rep)$ diagrama de valor medio y desviación estándar

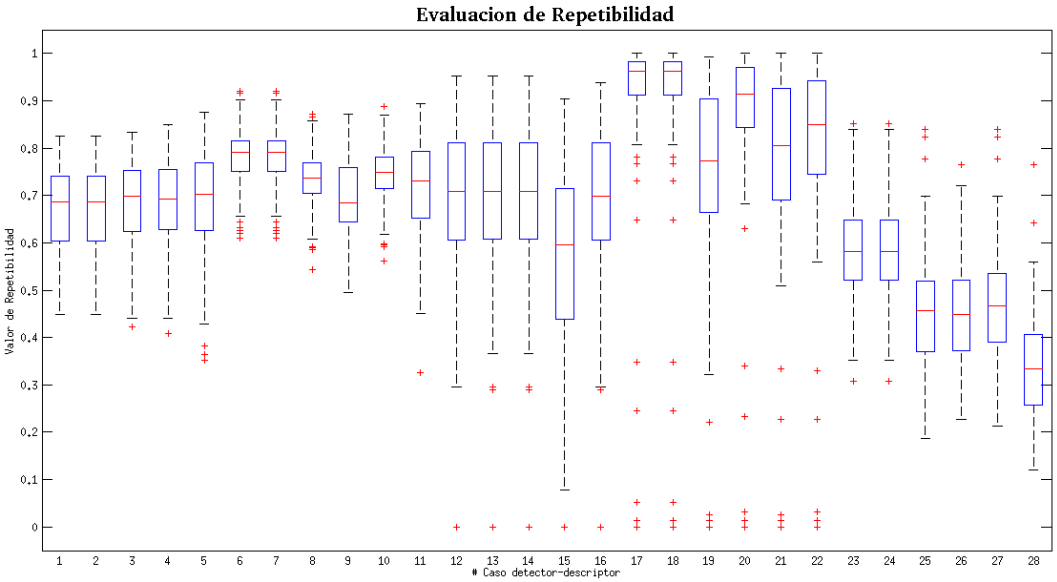


Figura 7.4: Parámetro de evaluación de $repetibilidad(Rep)$ diagrama de caja

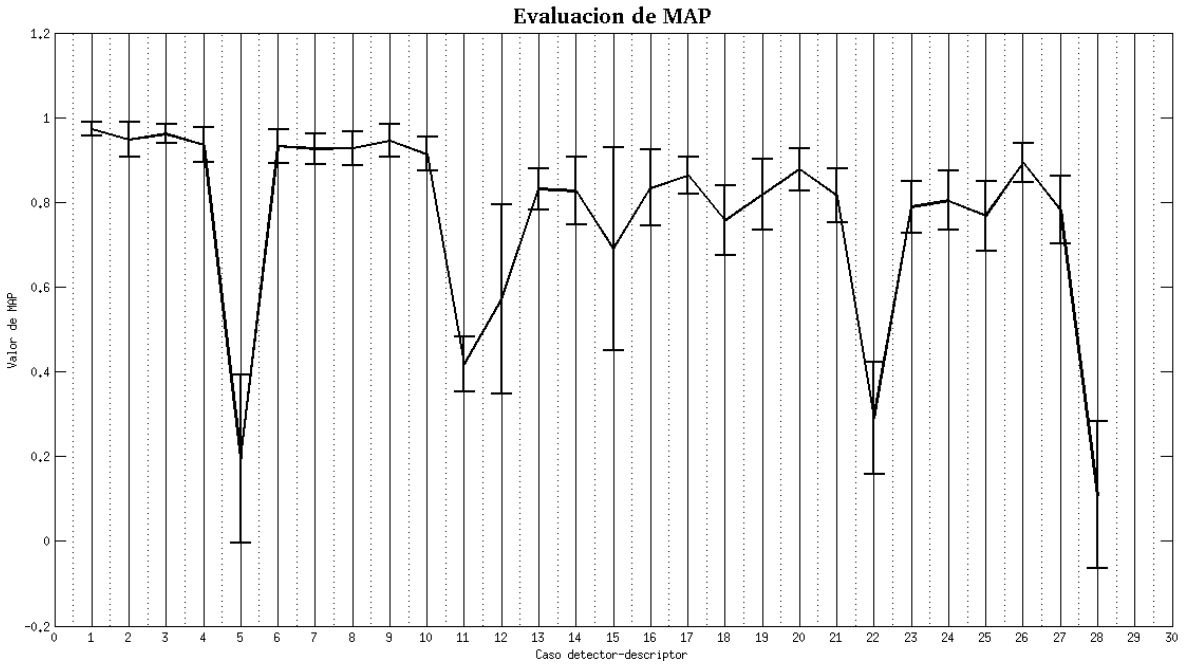


Figura 7.5: Parámetro de evaluación *MAP* diagrama de valor medio y desviación estándar

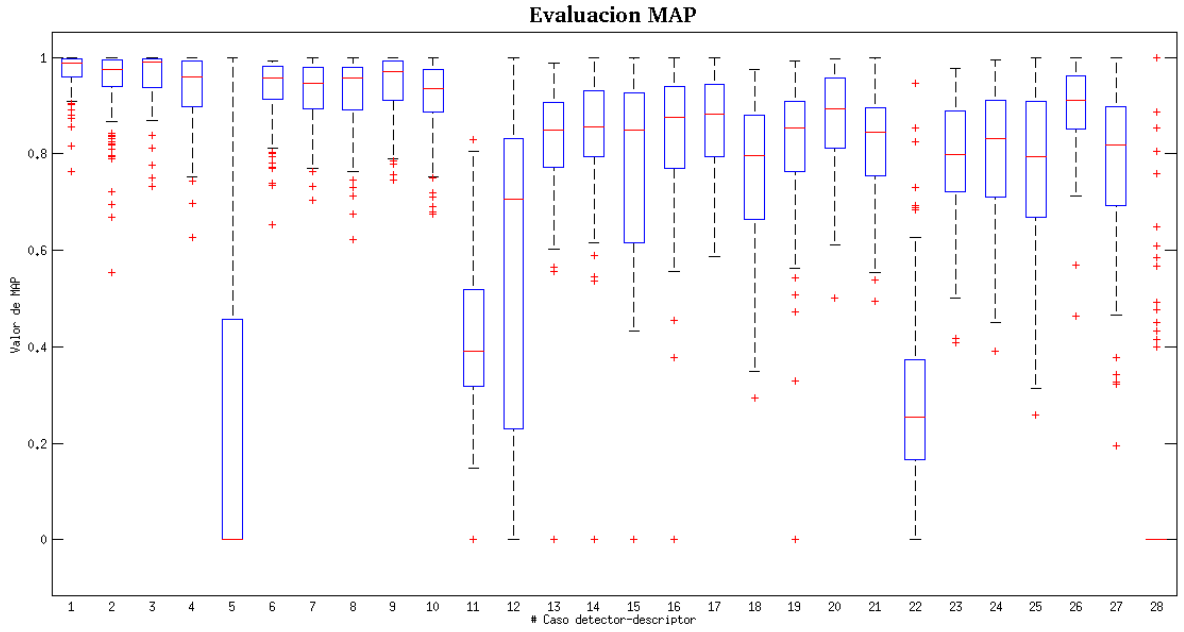


Figura 7.6: Parámetro de evaluación *MAP* diagrama de caja

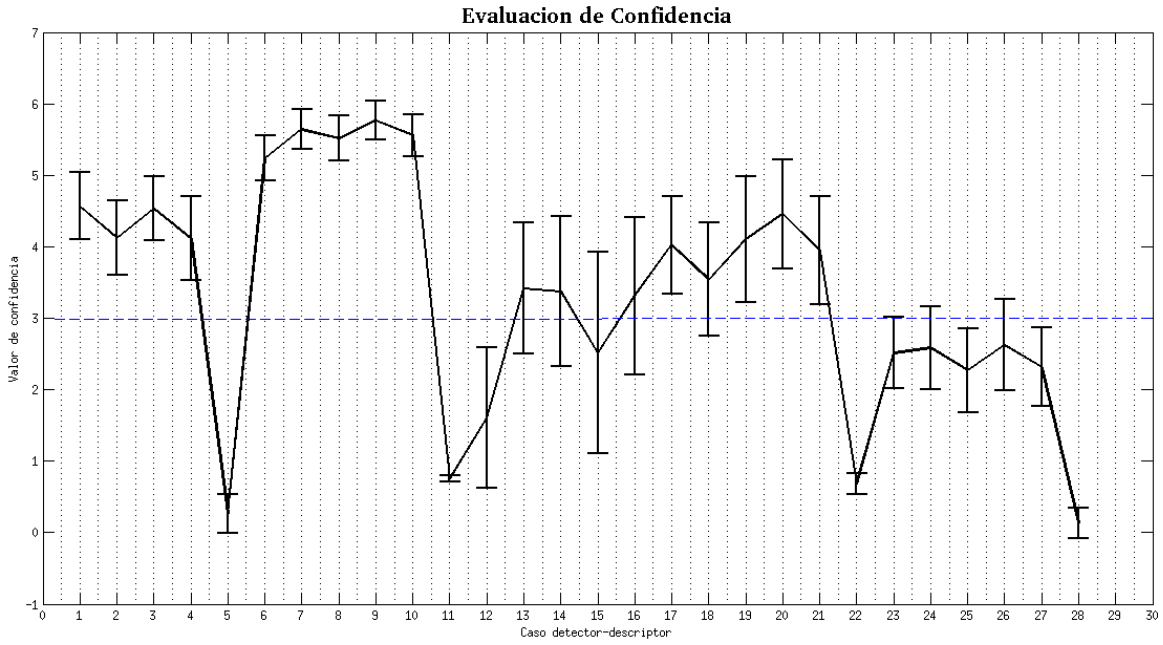


Figura 7.7: Parámetro de evaluación de $confidencia(C_{on})$ diagrama de valor medio y desviación estándar

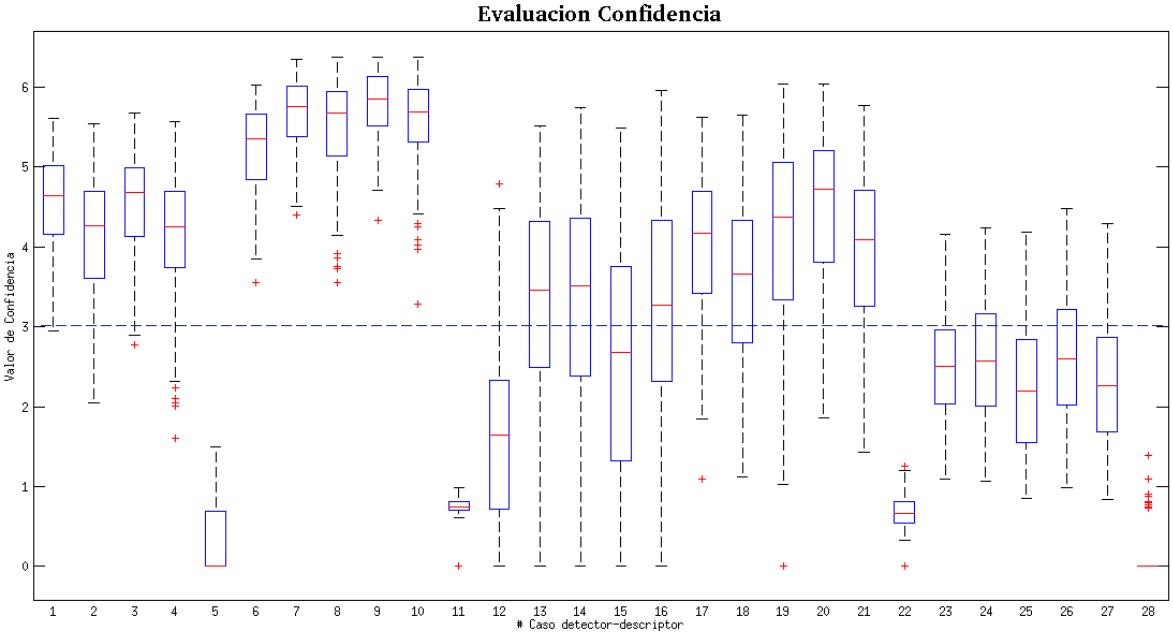


Figura 7.8: Parámetro de evaluación de $confidencia(C_{on})$ diagrama de caja

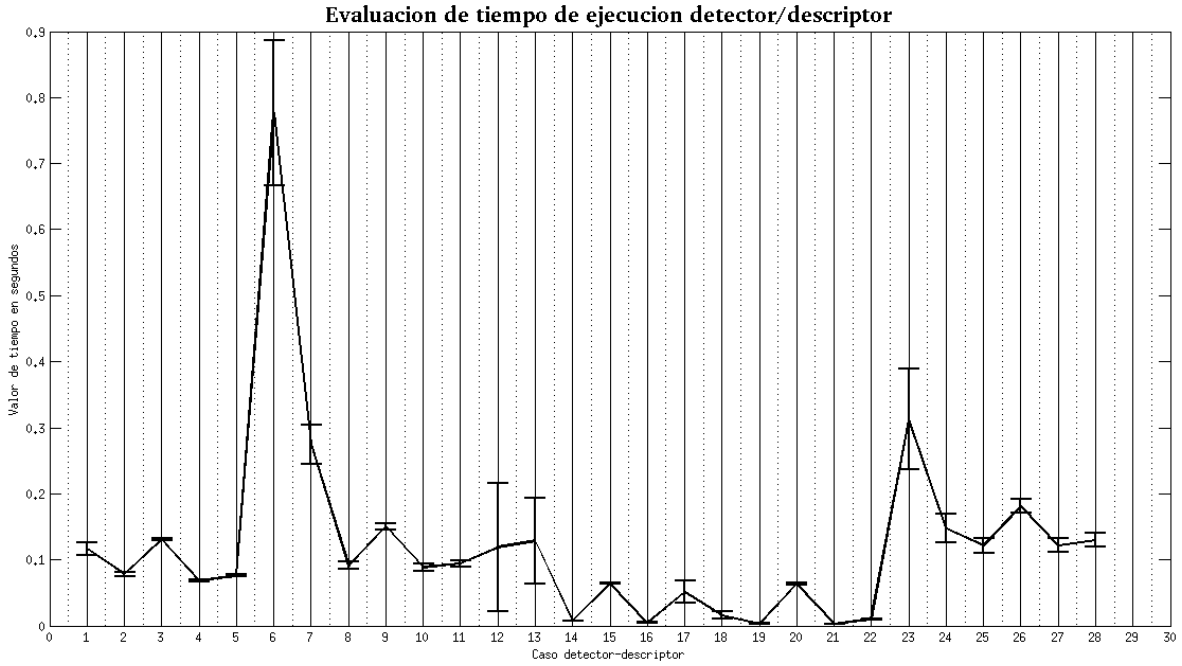


Figura 7.9: Parámetro de evaluación correspondiente al tiempo de ejecución detector-descriptor t_{dd} diagrama de valor medio y desviación estándar

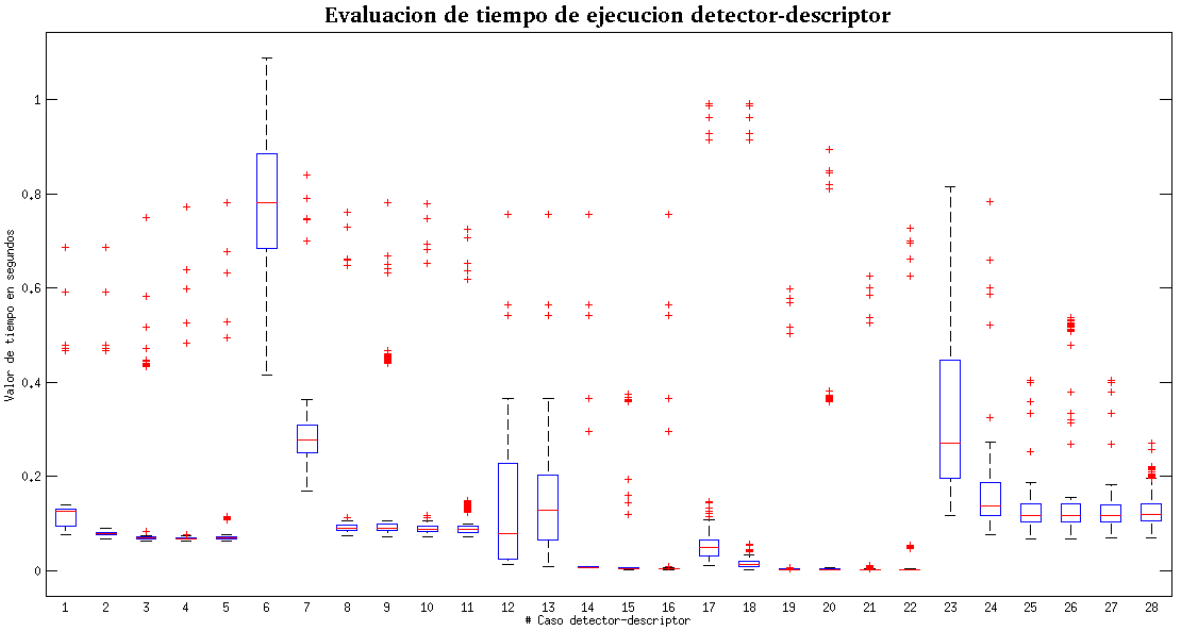


Figura 7.10: Parámetro de evaluación correspondiente al tiempo de ejecución detector-descriptor t_{dd} diagrama de caja

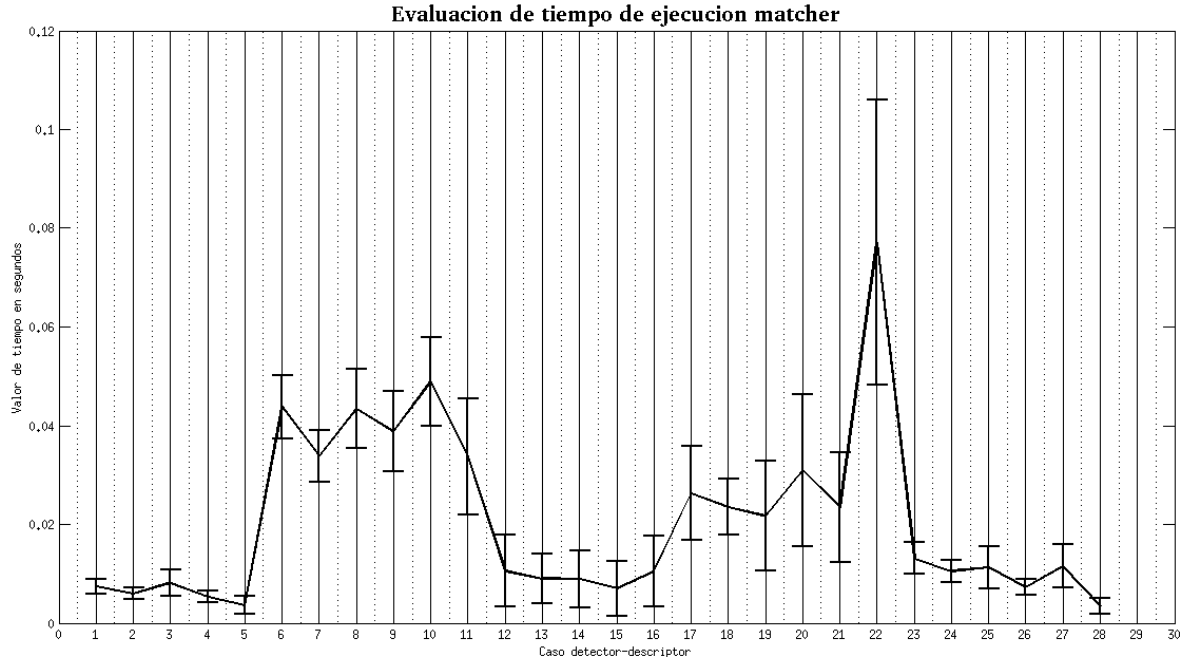


Figura 7.11: Parámetro de evaluación correspondiente al tiempo de ejecución *matcher* t_m diagrama de valor medio y desviación estándar

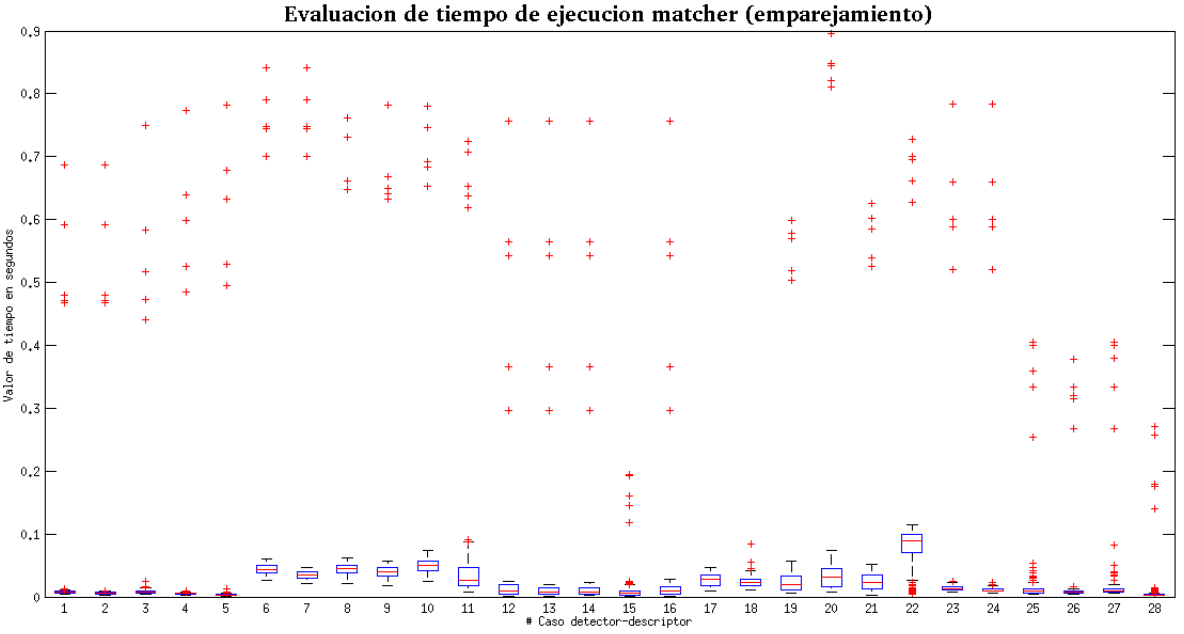


Figura 7.12: Parámetro de evaluación correspondiente al tiempo de ejecución *matcher* t_m diagrama de caja