Stitching automático de imágenes panorámicas con drones para la digitalización de zonas rurales aisladas usando MATLAB

Brayan Romero Bulla

Universidad Antonio Nariño

Facultad de Ingeniería Mecánica, Electrónica y Biomédica

Bogotá DC, Colombia

2022

Stitching automático de imágenes panorámicas con drones para la digitalización de zonas rurales aisladas usando MATLAB

Brayan Romero Bulla

Proyecto de grado presentado como requisito parcial para optar al título de:
Ingeniero Electrónico

Director:
PhD. Christian Camilo Erazo Ordoñez

Universidad Antonio Nariño
Facultad de Ingeniería Mecánica, Electrónica y Biomédica
Bogotá DC, Colombia
2022

AGRADECIMIENTOS

A mi familia por los sacrificios que me permitieron lograr cada unos de los objetivos que me propuse durante estos años de formación, su apoyo fue fundamental para poder culminar exitosamente este proceso formativo. A las personas que conocí a lo largo de este proceso, de la cuales tengo el gusto y el privilegio de llamarlos amigos, me dieron experiencias con las que voy a sentirme afortunado de conservar en mis recuerdos durante el resto de mi vida. A los maestros que compartieron sus conocimientos para aportar en mi formación como ingeniero, entre ellos mi asesor de tesis Chritian Erazo y en especial a Leonardo Torres a quien admiro y respeto, ya que fue la persona que me enseño la importancia del sacrificio y el trabajar duro, persona que confió en mis capacidades cuando ni yo mismo lo hacía. Por último, le agradezco a mi yo del pasado que no se rindió en ningún momento a pesar de las dificultades que se encontró en el camino, voy a estar enteramente agradecido por no desistir aun cuando esa parecía ser la única opción.

Resumen

En el presente proyecto se desarrolla y simula un algoritmo de Stitching, el cual tiene la función de unir imágenes capturadas por la simulación de un dron en Matlab y Simulink de forma completamente automática. Este algoritmo se basa en la coincidencia de características robustas aceleradas (SURF). Estas características, siendo locales e invariantes, permiten encontrar coincidencias entre múltiples imágenes sin importar el orden en que se encuentren, la orientación que dispongan, el tamaño que inicialmente tengan, el cambio significativo en su iluminación o si se ven borrosas debido al movimiento del dron a lo largo de su recorrido. Se describe también el proceso de creación del escenario y control de vuelo del dron, así como la validación del algoritmo para comprobar su correcto funcionamiento.

Palabras clave: Algoritmo, Stitching, características, coincidencias, transformación geométrica, escenario, dron

Abstract

In the present project a Stitching algorithm is developed and simulated, which has the function of joining images captured by the simulation of a drone in Matlab and Simulink in a completely automatic way. This algorithm is based on accelerated robust feature matching (SURF). These features, being local and invariant, allow finding matches between multiple images regardless of the order in which they are found, their orientation, their initial size, the significant change in their illumination or if they are blurred due to the movement of the drone along its path. The process of creating the scenario and flight control of the drone is also described, as well as the validation of the algorithm to verify its correct operation

Keywords: Algorithm, Stitching, features, matching, geometrical transformation, scenario, drone

Índice general

Índice de figuras

Capítulo 1

Introducción

Llegando a mediados del siglo XIX, cuando se realizaron significativos avances en la tecnología, la fotografía se convierte en una las invenciones más importantes, una vez la sociedad notó esto, las cámaras fotografías se volvieron mucho más comunes y a su vez también se despertó el interés por cada vez mejorar la calidad de las fotografías. Esto ha dado pie al desarrollo tecnológico, ya sea que se hable de cámaras tipo GoPro, cámaras de fotografías especializadas o de cámaras de teléfonos inteligentes. Si bien, la fotografía en un principio estuvo centrada al arte vinculada esencialmente a la pintura, luego las imágenes fotografiadas se vieron ligadas a la documentación de sucesos, eso llevó a que los lugares fotografiados fueran sitios comunes, es decir, se dio el inicio a las cámaras de seguridad. Por supuesto que esto ha ido escalando al punto de que países y algunas compañías privadas cuentan con satélites, los cuales permiten tener imágenes aéreas actualizadas y de alta resolución en prácticamente cualquier parte del mundo [1].

La obtención de imágenes aéreas en zonas específica es fundamental cuando se quiere obtener información aérea de áreas específicas, ya sea en zonas urbanas o rurales, para obtener este tipo de datos es necesario recurrir a medios de costo elevado como los ya mencionados satélites de observación, torres de control o aviones pilotados. Sin embargo, los UAVs (Unmanned Aerial Vehicle) o vehículos no tripulados son una alternativa que se ha vuelto popular a lo largo de los últimos años, ya que involucran un coste económico y energético significativamente reducido. Algunos de los usos más comunes de estos dispositivos están centrados en proyectos de ingeniería como en la supervisión de obras civiles o el levantamiento topográfico, no

obstante, en casos de emergencia como zonas de desastres, específicamente deslizamientos de tierra, derrumbes, inundaciones o incendios también son de gran ayuda a causa de que es posible evitar que el personal de rescate se ponga en riesgo entrando en este tipo de entornos. De igual manera, los UAVs se están expandiendo cada vez con más frecuencia a zonas rurales, siendo de ayuda para agricultores, ya que facilitan la supervisión de cultivos, el suministro de insumos agrícolas, además de vigilancia y seguridad de terrenos alejados [2]. El uso de los UAVs se ha vuelto frecuente por parte de la comunidad en general debido a su uso trivial y a que no requieren de un permiso para volarlos a menos que sobrepasen un peso de 250 gramos según la RAC 91 (Reglamentos Aeronáuticos de Colombia) [3], de igual manera, una más de las ventajas que poseen los UAVs es que no requieren un consumo energético elevado a la hora realizar vuelos, sin embargo, esto implica cierto tipo de limitaciones a la hora del despliegue en zonas de trabajo, por lo que este tipo de aeronaves se ven supeditadas a tiempos de vuelo no mayores a 10 minutos y a velocidades de no más de 8.3 m/s (aproximadamente 30 km/h), estos tomando como ejemplo en el caso del Dron Parrot Mambo [4].

En el campo de la robótica en general, uno de los temas más importantes es el control autónomo de los sistemas electrónicos. La agricultura es un ejemplo de esta problemática, ya que se utilizan diferentes sistemas cooperando entre sí para llevar a cabo diferentes tareas. Mayormente, en zonas rurales existe la necesidad llevar a cabo actividades como plantar semillas, fumigar las tierras de cultivo, proteger las plantas de organismos externos a ellas, monitorear variables y en general agricultura de precisión. En varias de estas tareas se emplearon drones aéreos no tripulados en los últimos años, debido a su rapidez a la hora de cubrir grandes distancias, sin afectar los propios cultivos [5].

1.1. Estado del arte

En las últimas décadas, tanto la ingeniería como la tecnología han tenido avances significativos en el diseño de vehículos no tripulados o también llamados UAVs (Unmanned Aerial Vehicle), así como en diferentes técnicas de control de vuelo, las cuales han sido desarrolladas e implementadas en diferentes contextos, en un principio utilizados en el ámbito militar como dispositivos de reconocimiento y rastreo, sin embargo, la accesibilidad ha permitido que este tipo de dispositivos se usen también en campos como la ingeniería civil para detectar fallas en diferentes tipos de estructuras, el mapeo de terrenos o en diferentes entornos como la

búsqueda y rescate de personas en áreas de difícil acceso, agricultura de precisión, etc. Estos dispositivos cuentan con diferentes tipos de control, los cuales pueden ser seleccionados en función de las necesidades de los objetivos de cada proyecto, en particular para este proyecto se trabaja en la plantilla que proporciona MATLAB llamada *Quadcopter Flight Simulation Model - Mambo* [6],

En [7] es posible apreciar las dificultades en cuanto a la inseguridad con la que cuentan las personas que habitan en zonas rurales, por algunas de las primeras soluciones a las que recurren las familias de estas localidades es ubicar cámaras para tener un control de lo que está pasando en los terrenos donde viven, sin embargo, este tipo de soluciones resultan viables para las personas que cuentan recursos relativamente altos, ya que este tipo de soluciones requieren de una compañía que realice la instalación, el monitoreo y el mantenimiento de este tipo de tecnologías, el Stithing es otra de las soluciones que aparecen en la mesa, ya que con la captura de imágenes tomadas por un dron de bajo costo es posible unirlas para poder tener una vista de los terrenos, las cuales no solo pueden ser de ayuda para la seguridad y la vigilancia, sino también para tener un control en el tema de los cultivos o el ganado que se encuentra en terrenos alejados. Sin embargo, los algoritmos que existen de stitching están incorporados en cámaras profesionales de alto costo a las que no es posible acceder, por otro lado, estos algoritmos también están disponibles en internet y es complicado contar con internet en localidades donde apenas llegan servicios básicos como el agua o la luz.

1.2. Descripción del problema

El avance en la tecnología actual ha permitido que los drones tengan diferentes aplicaciones, tanto en el ámbito de la arquitectura, como en la agricultura, en la seguridad, además de muchas otras. En el campo de la seguridad activa, los drones permiten un ahorro considerable de recursos (económico, humano y energético) a la hora de realizar tareas comunes como rondas de vigilancia perimetral, visualización de zonas comprometidas, identificación de amenazas, etc. Sin embargo, a la hora de implementar estas soluciones en zonas rurales alejadas de la ciudad y de un área extensa en las que se vuelve indispensable tener información actualizada, ya sea para saber el estado de cercas, verificar que el ganado no se pase de un lote a otro, la supervisión de cultivos o simplemente asegurarse de que no hay intrusos en la propiedad, se presentan dos problemáticas diferentes. La primera es la baja calidad que presentan las fotografías capturadas por este tipo de aeronaves no tripuladas, ya sea por la iluminación de la zona, la velocidad de captura de las imágenes o simplemente factores ambientales indeseables, lo que dificulta el reconocimiento de personas, objetos o animales cuando las imágenes son tomadas desde una altura considerable. La segunda está relacionada con la fusión de las imágenes, ya que no solo se suele perder nitidez en el proceso de fusión, sino que también es complicado realizar la correcta combinación digital de estas fotografías, porque no se hace de forma automática. Para ello es necesario organizar las imágenes manualmente, puesto que es una limitación que suelen tener los algoritmos y también es común que no se pueda distinguir la posición y la rotación correcta de cada una de las imágenes. Algunos algoritmos de Stitching o mosaico de imágenes logran realizar la unión de cada una de las imágenes capturadas sin perder calidad en la resolución [8], pero tienen cierto tipo de inconvenientes, como escoger de forma manual las características a comparar en cada una de las fotografías, lo que además de llevar bastante tiempo, es complicado para usuarios que no estén familiarizados con este conjunto de datos relacionados con el procesamiento digital de imágenes [9]. El sistema empleado en [10] como ya se mencionó hace uso de características locales invariantes y un modelo probabilístico, logrando verificar coincidencias en conjuntos de imágenes desordenadas y realizando el cocido de forma automática, también logra una significativa robustez en cuanto a factores como el cambio de iluminación, del Zoom y la orientación entre las imágenes haciendo que sea prácticamente imperceptible el solapamiento de las mismas, sin embargo, este algoritmo está diseñado para trabajar con relación a un

punto central, es decir, que la cámara no se traslada paralelamente a la escena capturada, sino rota con respecto a un punto central. Esto se vuelve un problema en el momento de trabajar con captura de imágenes con drones, ya que precisamente el modo en que trabajan los UAVs consiste en desplazarse de a lo largo de la escena y no rotando con respecto a un punto fijo.

1.3. Objetivos

1.3.1. Objetivo general

Implementar un algoritmo de Stitching automático de imágenes con ayuda de un entorno completamente simulado.

1.3.2. Objetivos específicos

- Diseñar y programar tanto la el escenario como el plan de vuelo para la captura de imágenes.
- Investigar y desarrollar el algoritmo de Stitching para la unión de imágenes.
- Efectuar validación del algoritmo de Stitching.

1.4. Justificación

La seguridad en zonas rurales suele ser un problema común porque es limitado el acceso a tecnología de vigilancia como cámaras u otros tipos de sensores que permitan la supervisión de diferentes terrenos donde se localizan cultivos, ganado o en general zonas de difícil acceso. En zonas rurales de terreno extenso, muchas veces requieren de supervisión, ya que el ganado puede trasladarse a otras propiedades o pueden existir intrusos o simplemente se requiere tener información actualizada a cerca de los cultivos y de las zonas rurales en general. Los drones se vuelven una herramienta eficaz para realizar este tipo de trabajo porque no son tan costosos como los satélites de observación, las torres de control o los aviones pilotados. Unas de las dificultades que pueden llegar a tener los drones son la batería, ya que los drones de bajo costo tienen dificultades para recorrer largas distancias por sus limitaciones energéticas, pero este problema podría resolverse analizando los requerimientos de cada zona rural en particular para determinar si es posible realizar diferentes vuelos para abarcar completamente la zona o si definitivamente es necesario recurrir a drones de mayor tamaño, otra dificultad suele ser la calidad en la imagen, debido a que la estabilidad del dron a lo largo del recorrido no es constante y depende varios parámetros como la velocidad a la que se capturan las fotografías, la altura a la que se sobrevuela la zona, las condiciones ambientales que difícilmente se pueden predecir con exactitud, además muchos otros. Algunos otros inconvenientes se presentan en [11] donde se trata de implementar un sistema de vigilancia por medio de estas aeronaves, donde es evidente que es necesario un algoritmo o herramienta que permita mejorar la calidad de las imágenes y también sería ideal que en vez de tener videos con minutos o incluso horas de grabación se tuvieran imágenes panorámicas de alta calidad donde fuera asequible tener un control rápido y robusto en cuanto a la seguridad y la vigilancia de este tipo de lugares. Como aclaración, este proyecto no pretende resolver por completo problemáticas como la vigilancia o la supervisión de zonas rurales donde se emplee el uso de los UAVs, simplemente representa una herramienta para facilitar el trabajo que los encargados de la seguridad realizan en este tipo de espacios. En cuanto al método para llevar a cabo la unión de imágenes, se propone tomar como base el sistema descrito en [10], realizando ajustes al algoritmo para solucionar el problema del paralaje que se mencionó en la sección anterior.

Capítulo 2

Metodología

A continuación, se presentan las etapas y métodos para la elaboración de este proyecto, donde en su mayor parte serán desarrollados en *Matlab/Simulink*, teniendo así las siguientes fases:

2.1. Creación de un ambiente virtual y obtención de datos

En primera instancia se requiere un grupo de imágenes capturadas desde una altura determinada con ayuda del mini dron, para esto se debe diseñar un entorno virtual con ayuda de *3D World Editor*, de igual manera es necesario definir la trayectoria del mini dron para que realiza la captura de imágenes durante el trayecto. Una vez hecha la toma de imágenes se almacenan en una carpeta especifica, la cual debe tenerse en cuenta para posteriormente usar estos datos para el procesamiento digital del las imágenes, y de esta manera unirlas en una sola imagen panorámica usando el algoritmo de stitching.

2.2. Desarrollo e implementación del Algoritmo de Stitching

Para el desarrollo del algoritmo de Stitching se tienen en cuenta cuatro pasos fundamentales, el primero en el que se leen las imágenes y se extraen las características invariantes (SURF) de cada una de ellas, en el segundo se comparan las características de cada una de las imágenes entre sí para encontrar coincidencias (RANSAC) y se determina un modelo probabilístico que nos diga con cual de las otras imágenes es mas probable que coincida, en el tercero se sobreponen las imágenes en su orden correcto y por último en el cuarto paso se realiza un

enderezamiento de la imagen.

2.3. Validación del Algoritmo de Stitching

En el proceso de validación se toma como base la imagen panorámica obtenida por el algoritmo de Stitching y una imagen capturada desde la parte superior de la pista, de esta forma se tienen dos imágenes las cuales pueden ser comparadas, para comprobar que las imágenes corresponden una con la otra se realiza correlación cruzada, lo que permite encontrar similitudes entre las dos imágenes y así gráficamente para comprobar que contienen la misma información.

Capítulo 3

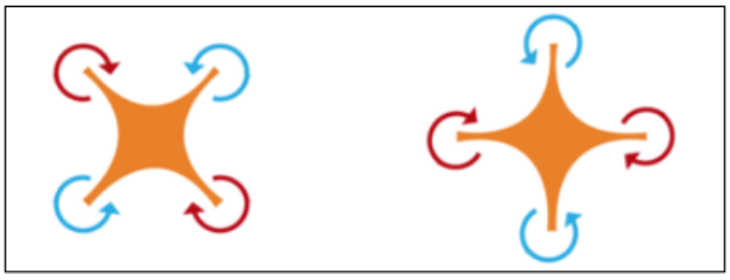
Marco Teórico

3.1. Modelo del Dron

3.1.1. Funcionamiento

Los drones, más específicamente los de tipo cuadricóptero, son aeronaves que funcionan mediante cuatro hélices que rotan a gran velocidad, las cuales generan una presión proporcional en cada una de los rotores, esto permite que se genere una fuerza en sentido contrario a la gravedad para así elevarse. Los rotores deben moverse sincrónicamente, donde dos de ellos giran en el sentido de las agujas de reloj y los otros dos en sentido contrario, esto para garantizar un aterrizaje seguro.

Figura 3.1: Sentidos de los rotores cuadricóptero



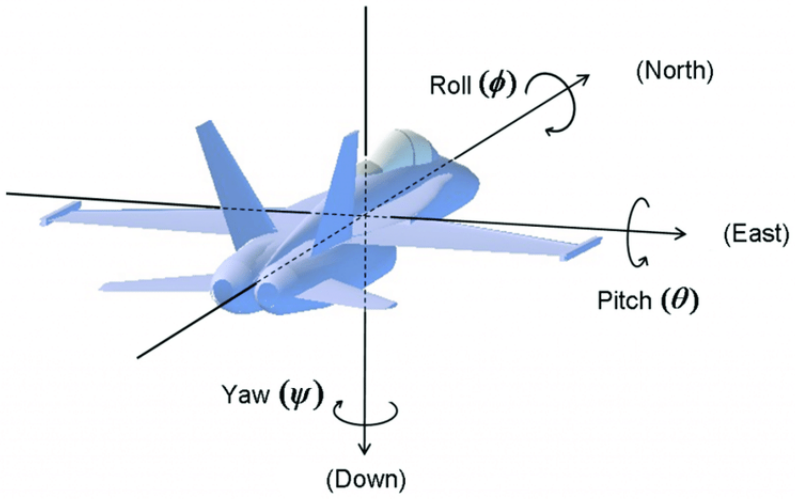
Fuente: [12]

La fuerza que se crea para realizar el desplazamiento vertical del cuadricóptero también es posible variarla, el empuje que ejerce cada rotor debe hacerse de manera sincrónica junto con

los otros tres para conseguir la estabilidad completa de la nave. Un cuadricóptero dispone de tres tipos de movimientos con respecto a sus ejes [13, 14]:

- Yaw: Rotación alrededor del eje vertical.
- Pitch: Rotación alrededor del eje de adelante hacia atrás.
- Roll: Rotación alrededor del eje de lado a lado.

Figura 3.2: Tipos de movimientos de un dron



Fuente: [14]

3.1.2. Minidron Parrot Mambo

El UAV ***Parrot Mambo*** es un cuadricóptero pequeño y ligero que es fácil de interconectar con Simulink: Matlab ofrece paquetes oficiales que permiten su fácil programación y en ordenadores con capacidades de procesamiento no muy elevadas, este dron cuenta con sensores de estabilización, los cuales permiten tener un control tanto de su posición como de su rotación en cada uno de los 3 ejes, también cuenta con un sensor de flujo óptico (cámara) con el que es posible tener una vista aérea de la zona que sobrevuela el dispositivo [4].

Figura 3.3: Mini Drone Parrot Mambo

Fuente: [4]

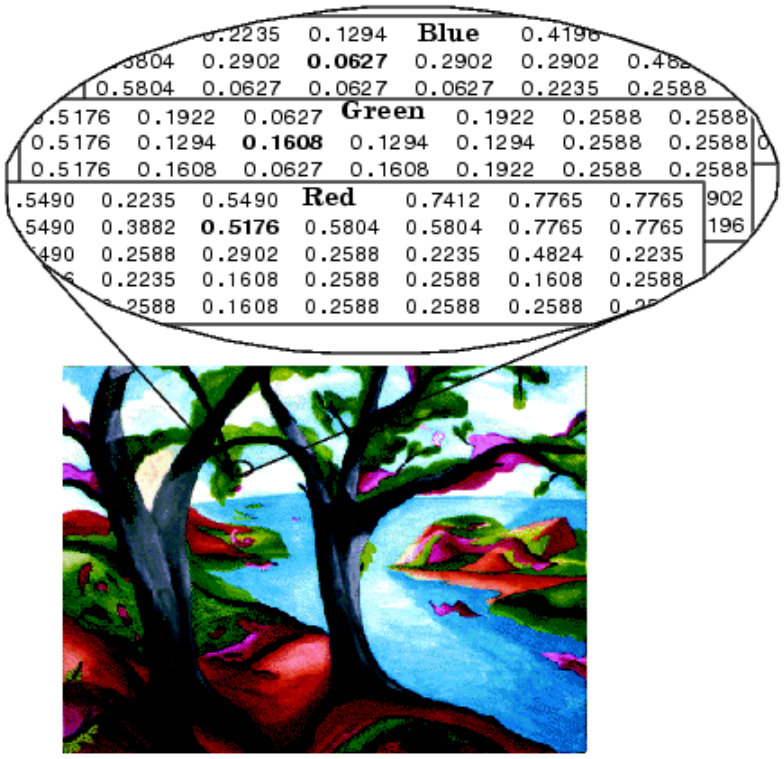
3.2. Tipos de Imagen

Existen varios tipos de imágenes con las que trabajan los ordenadores en este tipo de algoritmos, a continuación algunos ejemplos:

3.2.1. RGB

Una imagen RGB se almacena en MATLAB como una matriz de datos $mxnx3$ que define los elementos de color rojo, verde y azul para cada píxel personal, el color de cada píxel está definido por la unión de las intensidades de rojo, verde y azul almacenadas en cada plano de color en la localización del píxel. Los formatos de documento de gráficos almacenan imágenes RGB como imágenes de 24 bits, siendo los componentes rojo, verde y azul de 8 bits. Una matriz RGB MATLAB puede ser de clase double, uint8 o uint16.

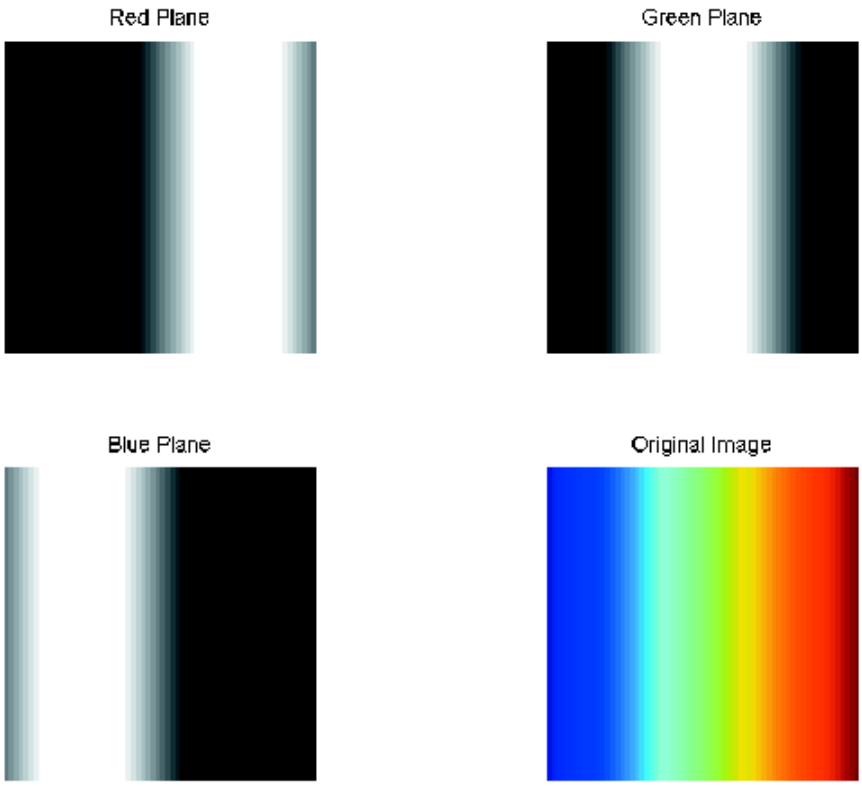
Figura 3.4: Los planos de color de una imagen RGB



Fuente: [15]

En la figura 3.5 se logra ver una imagen originar acompañada de tres otras tres con los planos separados, donde el blanco corresponde a los valores más altos (los tonos más puros) de cada color individual; el blanco representa la mayor concentración de valores de rojo, verde y azul puro, a diferencia de esto la zona negra de cada imagen muestra la escasez de estos colores, es decir que $R, G, B == 0$ [15].

Figura 3.5: Los planos de color separados de una imagen RGB



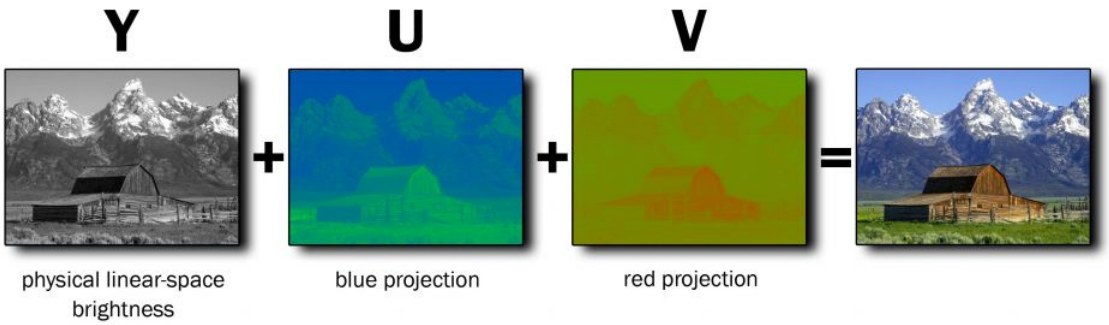
Fuente: [15]

3.2.2. YUV

YUV es un sistema de codificación de colores comúnmente utilizado como parte de la codificación de imágenes a color. Codifica imágenes o videos en color de acuerdo con la percepción humana, lo que permite una reducción en el ancho de banda de los componentes de crominancia, lo que generalmente permite que los errores de transmisión o las herramientas de compresión se enmascaren de manera más efectiva por la percepción humana de las personas en lugar de usar una representación RGB. El alcance de los términos Y'UV, YUV, YCbCr, YPbPr, etc. El modelo Y'UV define el espacio de color en términos de luminancia (Y') y dos componentes de croma (UV). El modelo de color Y'UV se utiliza en el estándar de vídeo en color compuesto PAL (excepto PAL-N). Los sistemas en blanco y negro más antiguos solo usan información de luminancia (Y'). La información de color (U y V) se agrega por separado

a través de la subportadora para que el receptor en blanco y negro pueda recibir y transmitir una imagen en color en el formato original en blanco y negro del receptor. Y' representa el componente de luminancia (brillo) y U y V son el componente de luminancia (color); La luminancia está representada por Y y la luma por Y': los símbolos iniciales (') representan la presión gamma, "luminancia"significa la luminancia del espacio físico lineal, mientras que "luma.[es] el brillo perceptible [16, 17].

Figura 3.6: Imagen descrita en escala de colore YUV

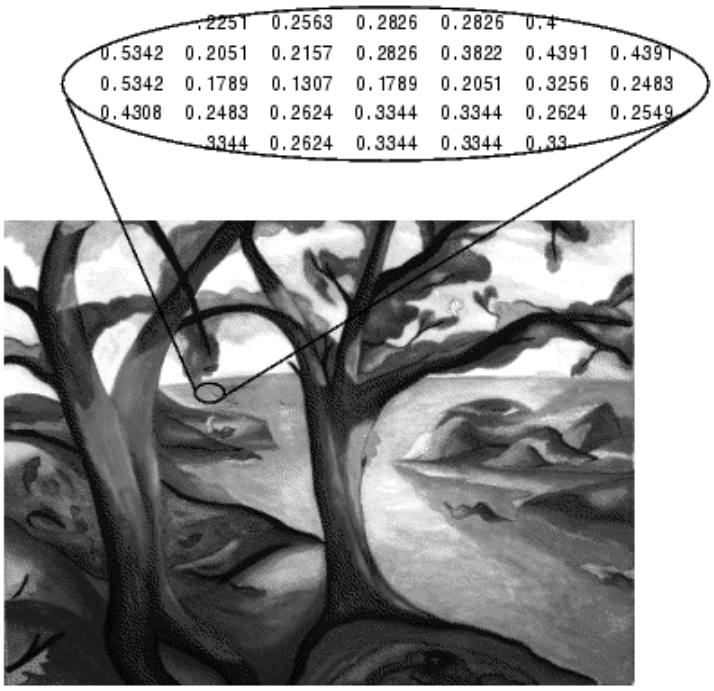


Fuente: [17]

3.2.3. Intensidad

Una imagen de intensidad es un conjunto de datos I, cuyos valores representan intensidades en un rango dado. Las matrices pueden ser de double, uint8 o uint16, aunque las imágenes de densidad rara vez se guardan con mapas de colores, MATLAB utiliza mapas de colores para mostrarlas. Los recursos de la matriz de magnitud representan diferentes intensidades, o niveles de gris, donde la magnitud 0 principalmente representa el negro y la magnitud 1, 255 o 65535 principalmente representa la magnitud total, o blanco, en la figura 3.7 es posible apreciar una imagen en escala de grises, donde la matriz está en un rango de 0 a 1 [15].

Figura 3.7: Los valores de píxeles en una imagen de intensidad definen la escala de gris

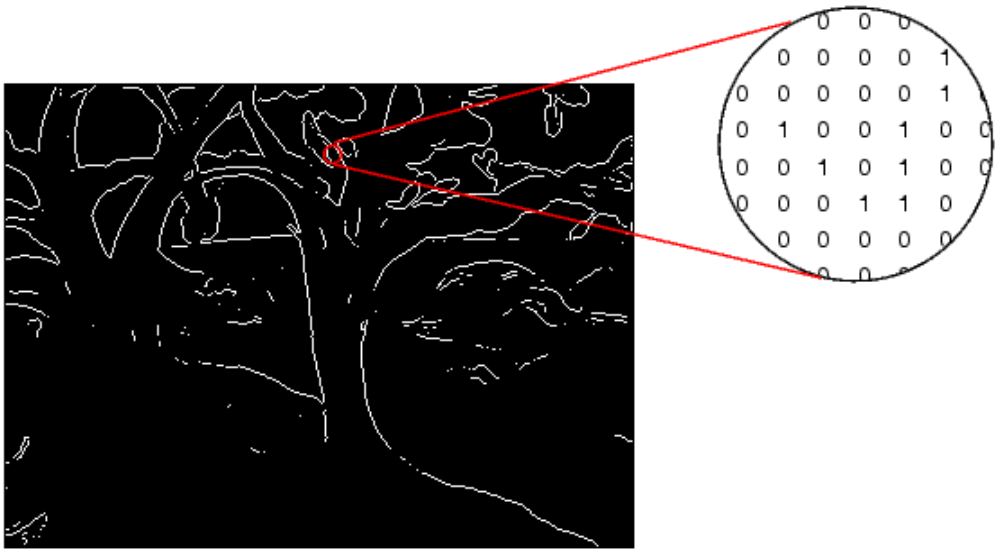


Fuente: [15]

3.2.4. Binaria

En una imagen binaria, cada píxel toma solo uno de dos colores separados, banco o negro, donde básicamente estos dos valores corresponden a on/off. La imagen binaria se almacena como una matriz lógica de 0 (offpíxeles) y 1 (onpíxeles). En la figura es logra apreciar la imagen anterior binarizada, donde el valor que toma cada pixel es el valor al que esté más cercano, ya sea 0 o 1 [15].

Figura 3.8: Los píxeles en una imagen binaria



Fuente: [15]

3.3. Stitching

Stitching o mosaico de imágenes hace referencia al proceso de unir múltiples imágenes fotográficas con rangos de visión superpuestos para producir una imagen panorama de alta resolución, dicho en otras palabras, el Stitching puede describir como el solapamiento exacto entre cada una de las imágenes originales, las cuales forman una sola panorámica [18]. El método de stitchig que se toma como referencia es el que desarrollan Brown y Lowe en [10], que se divide en 5 fases respectivamente:

1. Comparación de características
2. Correspondencia de imágenes
3. Ajuste de paquetes
4. Enderezamiento automático del panorama
5. Compensación de ganancia
6. Mezcla multibanda

3.3.1. Comparación de Características

En la literatura relacionada con la detección de características en imágenes se encuentran principalmente divididas en dos categorías, una es por métodos directos [19, 20], y basados en características [21], este documento toma como ejemplo el método basado en características, ya que no requiere de inicializaciones, como el orden de las imágenes. Los métodos basados en características toman como referencia la invariancia geométrica (traslación, rotación y escala) o fotométrica (brillo, contraste y color) o combinaciones de los dos, algunos de estos métodos son las siguientes:

Figura 3.9: Imagen original

3.3.1.1. HARRIS

El detector HARRIS mide la similitud de un parche de imagen centrado en un punto con parches cercanos superpuestos utilizando la suma de diferencias al cuadrado de las intensidades de los píxeles. La medida puede expresarse utilizando los valores propios de la matriz de suma de diferencias al cuadrado obtenida por la expansión de Taylor de la medida inicial [22] [23], en la figura 3.10 se puede ver la imagen con las características que detecta la

función propia de Matlab.

Figura 3.10: Detector HARRIS

.

3.3.1.2. SIFT

SIFT [24] realiza la convolución de la imagen con filtros gaussianos a varias escalas y el detector encuentra puntos claves invariantes de la escala seleccionando los extremos locales tanto en la escala como en el espacio. El descriptor SIFT es un histograma de las direcciones locales del gradiente de la imagen alrededor del punto de interés. RootSIFT [25] se diferencia de SIFT en que utiliza una distancia Hellinger en lugar de la distancia euclidiana estándar para medir la similitud entre sus descriptores [23], en la figura 3.11 se puede ver la imagen con las características que detecta la función propia de Matlab.

Figura 3.11: Detector SIFT

.

3.3.1.3. FAST

Con el objetivo de acelerar la recuperación de puntos clave, FAST [26] comprueba un número de píxeles más brillantes u oscuros en un anillo alrededor de un punto determinado, tras lo cual utiliza un clasificador de árbol de decisión aprendido previamente en un conjunto de imágenes similares para mejorar la eficiencia [23], en la figura 3.12 se puede ver la imagen con las características que detecta la función propia de Matlab.

Figura 3.12: Detector FAST

.

3.3.1.4. SURF

SURF [27] pretende acelerar SIFT convolucionando la imagen con una aproximación de la derivada de segundo orden del filtro Gaussiano utilizando imágenes integrales. El descriptor SURF asigna la orientación a un punto clave calculando la respuesta Haar wavelet en las direcciones x y y de una vecindad circular [23], en la figura 3.13 se puede ver la imagen con las características que detecta la función propia de Matlab.

Figura 3.13: Detector SURF

.

3.3.1.5. BRISK

BRISK [28] identifica los puntos de interés en la pirámide de imágenes utilizando el Criterio de Saliencia. Para cada punto clave, la orientación se obtiene aplicando un patrón de muestreo a su vecindad, y recuperando los valores de gris [23], en la figura 3.14 se puede ver la imagen con las características que detecta la función propia de Matlab.

Figura 3.14: Detector BRISK

.

3.3.1.6. ORB

ORB [29] (FAST orientado y BRIEF girado) intenta acelerar aún más SIFT y SURF. El detector ORB utiliza FAST para calcular los puntos clave y añade invariabilidad de rotación asignándoles una orientación por el centroide ponderado de intensidad. El descriptor ORB se basa en BRIEF [30] y añade la invariabilidad de la rotación girando los pares de puntos con la orientación de los puntos clave encontrados previamente [23], en la figura 3.15 se puede ver la imagen con las características que detecta la función propia de Matlab.

Figura 3.15: Detector ORB

.

3.3.1.7. KAZE

Entre los algoritmos clásicos más recientes, distinguimos KAZE [31]. El detector KAZE se basa en un determinante de la matriz Hessiana normalizada que se calcula en múltiples niveles de escala. Los máximos de las respuestas del detector se recogen como puntos clave utilizando una ventana móvil. El descriptor KAZE invariante de la rotación se obtiene encontrando la orientación dominante en una vecindad circular alrededor de cada punto clave[23], en la figura 3.16 se puede ver la imagen con las características que detecta la función propia de Matlab.

Figura 3.16: Detector KAZE

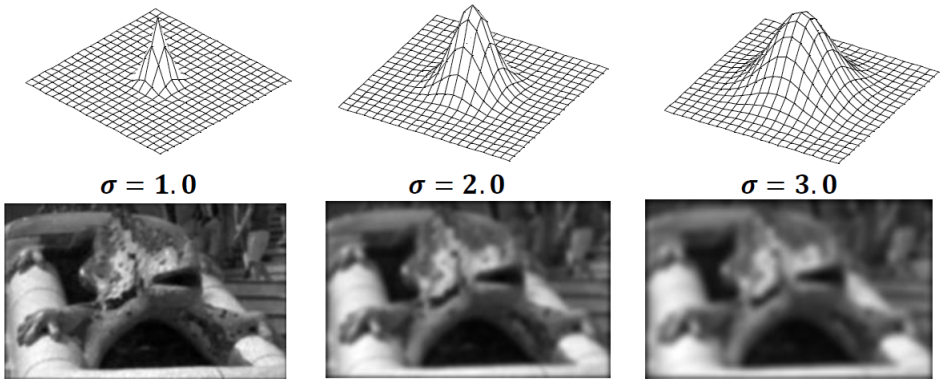
3.3.2. Filtro Gaussiano

Un filtro gaussiano es un filtro cuya respuesta de impulso es una función gaussiana, la función del filtro Gaussiano es preservar los bordes, disminuir el ruido y suavizar la imagen; sin embargo, la imagen resultante no es una imagen nítida.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.3.1}$$

x y y corresponden a las coordenadas de la región de la imagen y σ corresponde a valor del suavizado, si el valor de sigma es elevando, esto implican mayor suavizado de la imagen. En la figura 3.18 se observa el suavizado de una imagen, donde el valor de σ toma tres valores diferentes y se puede notar el nivel de suavizado en cada una de ellas [32].

Figura 3.17: Función Gaussiana



Fuente: [32]

3.3.3. Correspondencia de imagenes

En esta sección se desarrolla la correspondencia de características, la cual se lleva a cabo una vez se obtienen los puntos de interés de cada una de las imágenes, el método de comparación abordado en este caso es RANSAC y para la verificación de la coincidencia se usa un modelo probabilístico.

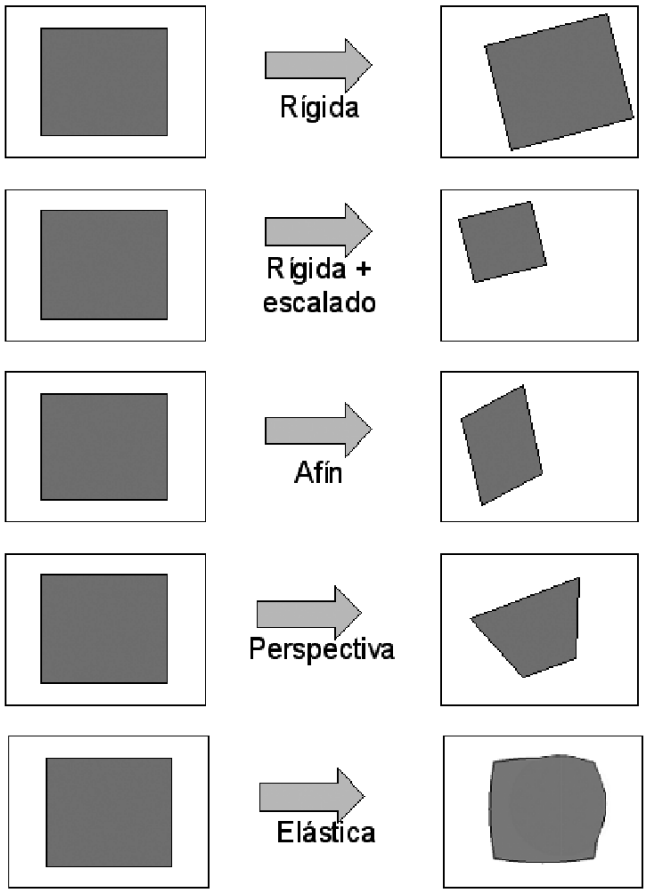
3.3.3.1. Transformación Geométrica

Al aplicar una transformación geométrica a una imagen, se modifican las coordenadas de sus píxeles y la imagen original queda modificada. La deformación de la imagen será mayor o menor, según el tipo de transformación que sea aplicada, de igual manera, el tipo de transformación a usar dependerá del tipo del caso en particular que se esté tratando, a continuación se presentan los diferentes tipos de transformaciones geométricas [33]:

- Transformación rígida: Es aquella en las que se traslada y rota la imagen original.
- Transformación rígida con escalado: Similar al anterior, pero agregando *zoom* a la imagen.
- Transformación afín: la imagen transformada cumplirá con que las líneas paralelas sigan siéndolo una vez transformadas.
- Transformación de perspectiva: Representa un cambio de perspectiva en la imagen, se cumple que las líneas rectas lo siguen siendo tras la transformación.

- Transformación elástica: Este tipo de transformación no es lineal, permiten deformar elásticamente una imagen para que se parezca a la imagen de referencia.

Figura 3.18: Tipos de transformaciones geométricas



Fuente: [33]

3.3.3.2. RANSAC

RANSAC (RANdom SAmple Consensus) es un algoritmo iterativo que estima de modelos matemáticos lineales a partir de un conjunto de datos que contienen valores anómalos. [34].

Estos valores anómalos se presentan en los conjuntos de datos por diferentes razones, como los errores de medición, ruido, incorrecta inserción de los datos o por inconvenientes relacionados a los datos con los que se está trabajando. Este algoritmo se desarrolla en 4 simples pasos:

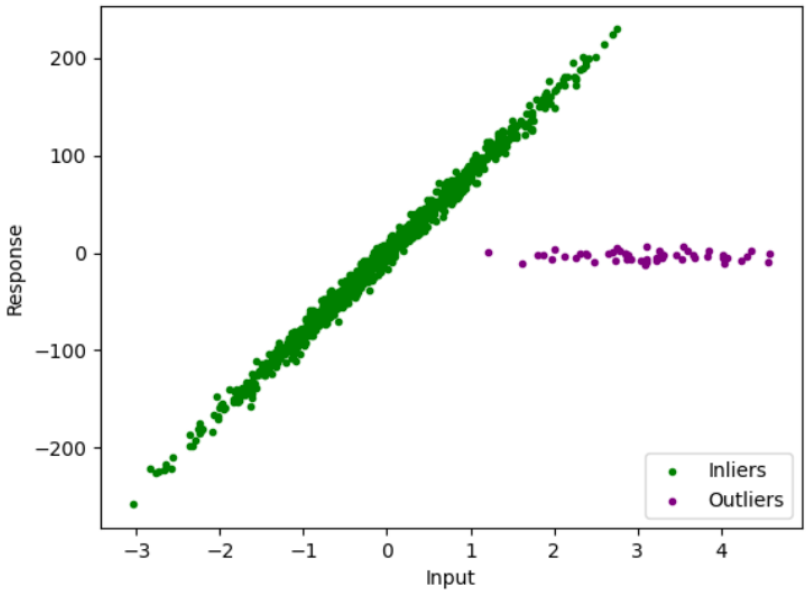
1. Selección aleatoria de un subconjunto del conjunto de datos
2. Ajuste de un modelo al subconjunto seleccionado

3. Determinación del número de valores anómalos

4. Repetir los pasos 1-3 para un número prescrito de iteraciones

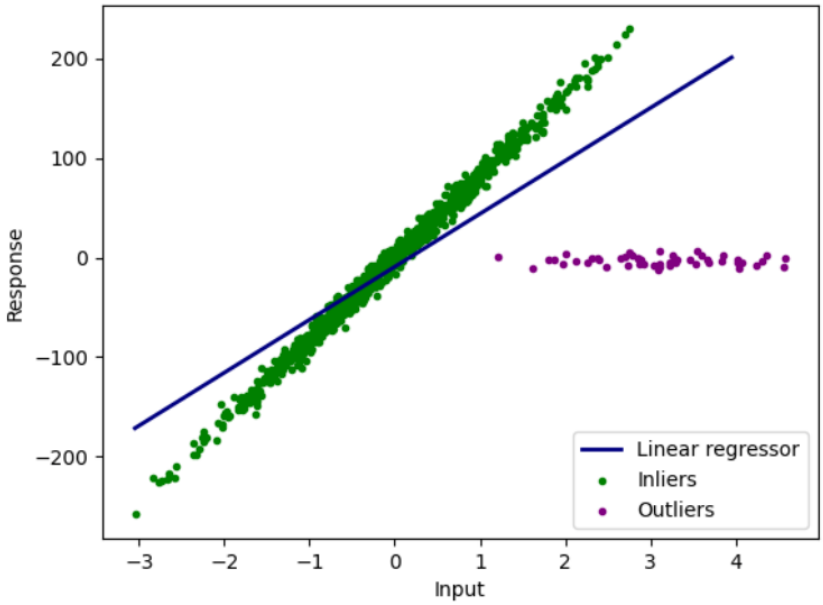
Como ejemplo se tiene este conjunto de datos, donde en la figura 3.19 se aprecia que existe un grupo de datos que responden a un comportamiento lineal, a los cuales se le va a dar el nombre de Inliers, sin embargo, también existe un grupo de datos anómalos marcados con color púrpura que van a denominarse Outliers.

Figura 3.19: Ejemplo algoritmo RANSAC



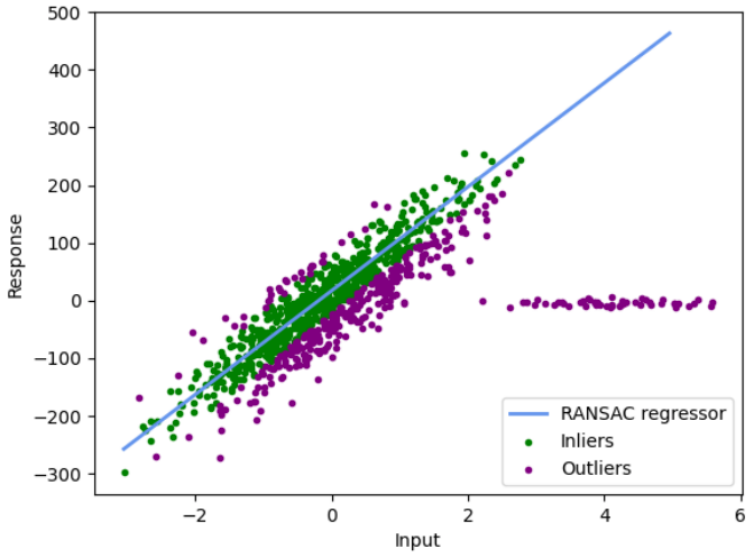
En este ejemplo es fácilmente apreciar cuáles corresponden a las muestras anómalas, de igual manera, se intenta trazar la recta que describa el conjunto de datos usando un método como el de regresión lineal, sin embargo, este es un método que no logra describir correctamente el comportamiento de los datos porque no distingue entre Inliers y Outliers.

Figura 3.20: Ejemplo Regresión lineal



Se usa t como el parámetro que define el umbral para determinar qué tan lejana se encuentra la línea de tendencia ajustada a los puntos escogidos, de esta manera se determina si son candidatos para considerarlos Inliers, como pasa en el siguiente caso.

Figura 3.21: Ejemplo algoritmo RANSAC



Este umbral (t) de ser escogido incorrectamente puede dar lugar a un modelo erróneo, ya que percibe los Inliers como Outliers.

El número de iteraciones es directamente proporcional a la probabilidad de que se detecte un subconjunto sin Outliers, es decir, cuan mayor sea el número de iteraciones, será más probable que se detecte un subconjunto sin datos anómalos en él. ω es la manera de expresar estadísticamente la relación entre el conjunto de datos anómalos y el total de puntos.

$$\omega = \frac{Inliers}{puntos\_totales} \tag{3.3.2}$$

Para determinar el número correcto de iteraciones se tiene la siguiente expresión:

$$k = \frac{\log(1-p)}{\log(1-\omega^n)} \tag{3.3.3}$$

Donde p es la probabilidad de encontrar un subconjunto sin Outliers y n es el número de datos con el que cuenta el conjunto original. Por poner un ejemplo, se selecciona un conjunto de datos que contiene 10 valores ($n = 10$), donde el 20 % de los datos pertenecen a valores anómalos ($\omega = 0{,}8$) y se desea que la probabilidad de que se escoja un correcto modelo, es decir, sin valores anómalos sea del 99 %($p = 0{,}99$). Realizando el cálculo, se determina que realizando 41 iteraciones, el algoritmo RANSAC tiene la probabilidad de 1 % de fallar [35]. En la figura 3.22e se aprecia un ejemplo de dos imágenes donde se desarrolla la detección de Inliers y se realiza la unión de imágenes basándose en el algoritmo RANSAC.

(a) Imagen 1

(b) Imagen 2

(c) Características SURF 1

(d) Características SURF 2

(e) RANSAC inliers

(f) Imágenes alineadas

3.3.3.3. Modelo probabilístico para la verificación de coincidencias de imágenes

El modelo probabilístico para la verificación de coincidencias de imágenes tiene como fin comparar las probabilidades de que este conjunto de Inliers/Outliers haya sido denominado como una coincidencia correcta o incorrecta [36, 10].

$$p(f^{1:n_f} \,|\, m=1) = B(n_i; n_f, p_1) \tag{3.3.4}$$

$$p(f^{1:n_f} \,|\, m=0) = B(n_i; n_f, p_1) \tag{3.3.5}$$

Donde p_1 es la probabilidad de que una característica sea un Inlier dada una coincidencia de imagen correcta, y p_0 es la probabilidad de que una característica sea un Inlier dada una coincidencia de imagen falsa. El conjunto de variables de coincidencia de características $\{f^{(i)}, i = 1, 2, 3..., n_f\}$ se denomina $f^{(1:n_f)}$. El número de Inliers es $n_i = \sum_{i=1}^{n_f} f^{(i)}$ y $B(...)$ corresponde a la distribución binomial.

$$B(x; n, p) = \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x} \tag{3.3.6}$$

Se escoge $p1 = 0{,}6$ y $p0 = 0{,}1$, de esta manera es posible evaluar la probabilidad de que una imagen coincida correcta utilizando la regla de Bayes, la cual se expresa enseguida.

$$p\left(m=1|\, f^{(1:n_f)}\right) = \frac{p\left(f^{(1:n_f)} \big|\, m=1\right) p\,(m=1)}{p\left(f^{(1:n_f)}\right)} = \frac{1}{1 + \frac{p\left(f^{(1:n_f)} \big| m=0\right) p(m=0)}{p\left(f^{(1:n_f)} \big| m=1\right) p(m=1)}} \tag{3.3.7}$$

Se acepta una coincidencia entre dos imágenes si se cumple que $p\left(m=1|\, f^{(1:n_f)}\right) > p_{min}$

$$\frac{B(n_i; n_f, p_1) p\,(m=1)}{B(n_i; n_f, p_0) p\,(m=0)} \underset{reject}{\overset{accept}{\gtrless}} \frac{1}{\frac{1}{p_{\text{mín}}} - 1} \tag{3.3.8}$$

La elección de los valores $p\,(m=1) = 10^{-6}$ y $p_{\text{mín}} = 0{,}999$, se la siguente condición para que pueda darse una correcta coincidencia de imágenes.

$$n_i > \alpha + \beta n_f \tag{3.3.9}$$

Donde $\alpha = 8$ y $\beta = 0{,}3$. Una vez establecidas las coincidencias entre imágenes, se logra encontrar secuencias panorámicas como conjuntos conectados de imágenes coincidentes. Esto nos permite reconocer múltiples panorámicas en un conjunto de imágenes [10].

3.3.4. Ajuste de paquete

En un conjunto de coincidencias geométricas de ciertas las imágenes, se utiliza el ajuste de paquetes para resolver los parámetros de la cámara conjuntamente. Sin el ajuste de paquetes, la unión de las homografías provocaría errores y no tomaría diferentes consideraciones o restricciones entre las imágenes, como que los extremos de una o varias de ellas se uniesen. Una por una las imágenes ingresan al ajustador de paquetes, incluyendo en cada ciclo la imagen que más se ajuste, basándose en el número máximo de coincidencias coherentes [37]. Cada vez que se analiza una nueva imagen, se inicializa con la misma distancia focal y rotación en comparación a la imagen con la que mejor corresponda [10].

3.4. Correlación cruzada

La correlación cruzada es una relación de similitud entre dos funciones, señales o imágenes para este caso, frecuentemente usada para encontrar características relevantes en una función desconocida por medio de la comparación con otra que ya conocida [38].

La correlacion cruxada utiliza el siguiente procedimiento:

- Calcula la correlación cruzada, ya sea en el dominio del espacio o de la frecuencia, eso depende del tamaño de las imágenes.
- Calcula las sumas locales precontabilizando las sumas en curso.
- Utiliza sumas locales para normalizar la correlación cruzada y conseguir los coeficientes de correlación.

Se expresa de la siguiente manera:

$$\gamma(u,v) = \frac{\sum_{x,y}\left[f(x,y) - \overline{f}_{u,v}\right]\left[t(x-u,y-v) - \overline{t}\right]}{\left\{\sum_{x,y}\left[f(x,y) - \overline{f}_{u,v}\right]^2 \sum_{x,y}\left[t(x-u,y-v) - \overline{t}\right]^2\right\}^{0,5}} \tag{3.4.1}$$

donde f es la imagen, $\overline{t}$ es la medida de la plantilla y $\overline{f}_{u,v}$ es la medida de $f(x,y)$ debajo de la plantilla [39, 40].

3.5. Parallel Computing Toolbox

Parallel Computing Toolbox permite la solución de problemas informáticos y de datos con uso intensivo de datos con procesadores multinúcleo. Las construcciones de alto nivel, como bucles paralelos, tipos de matrices especiales y algoritmos numéricos paralelos, permiten equilibrar las aplicaciones de MATLAB sin programación CUDA o MPI. Este Toolbox permite usar funciones que están habilitadas en paralelo en MATLAB. Este Toolbox también permite trabajar con Simulink para ejecutar varias simulaciones de modelos en paralelo. Este toolbox hacer uso de toda la potencia de procesamiento de los equipos multinúcleo gracias a la ejecución de aplicaciones en workers (motores de cálculo de MATLAB) [41, 42]. Existen diferentes formar de abrir el Parallel Pool, la forma que este algoritmo utiliza es: `parpool(4)`, donde el 4 corresponde al número de nucleos que manera el equipo con el que se está trabajando, esto puede variar dependiendo el dispositivo. Otra forma en la que se activa el Parallel Poool es con bucles, el que se usa en diferentes partes del codigo es `parfor`, pero los siguientes también se pueden utilizar [42]:

- `spmd`
- `distributed`
- `Composite`
- `parfeval`
- `parfevalOnAll`
- `gcp`
- `mapreduce`
- `mapreducer`

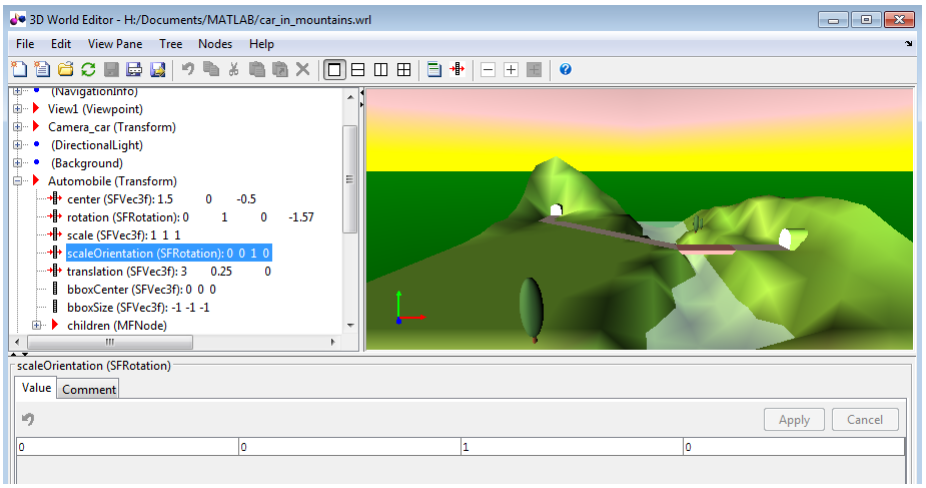
- `tall`
- `ticBytes` and `tocBites`

3.6. 3D World Editor

La aplicación 3D World Editor crea universos virtuales para visualizar y comprobar la conducta dinámica del sistema usando Simulink 3D Animation. En la figura 3.23 se puede observar la interfaz de esta aplicación. Este puede ser utilizado para:

- Crear objetos de mundo virtual desde cero usando tipos de nodos X3D o VRML
- Diseñar objetos utilizando plantillas disponibles en la biblioteca de objetos de 3D World Editor
- Importar objetos exportados desde herramientas CAD
- Simplificar la forma de los objetos importados
- Generar o cambiar la jerarquía de objetos en la escena
- Asignar nombres únicos a los objetos del mundo virtual para que se pueda acceder a ellos desde MATLAB y Simulink
- Configurar el fondo, la iluminación y las propiedades de navegación de la escena
- Identificar el punto de vista apropiado que tenga sentido para trabajar con el mundo virtual

Figura 3.23: 3D World Editor



Fuente: [43]

Capítulo 4

Algoritmo de control de trayectoria

4.1. Diseño de Entorno Virtual

Para diseñar y desarrollar el entorno virtual de simulación se trabaja en entorno gráfico de simulación Simulink el cual proporciona Matlab.

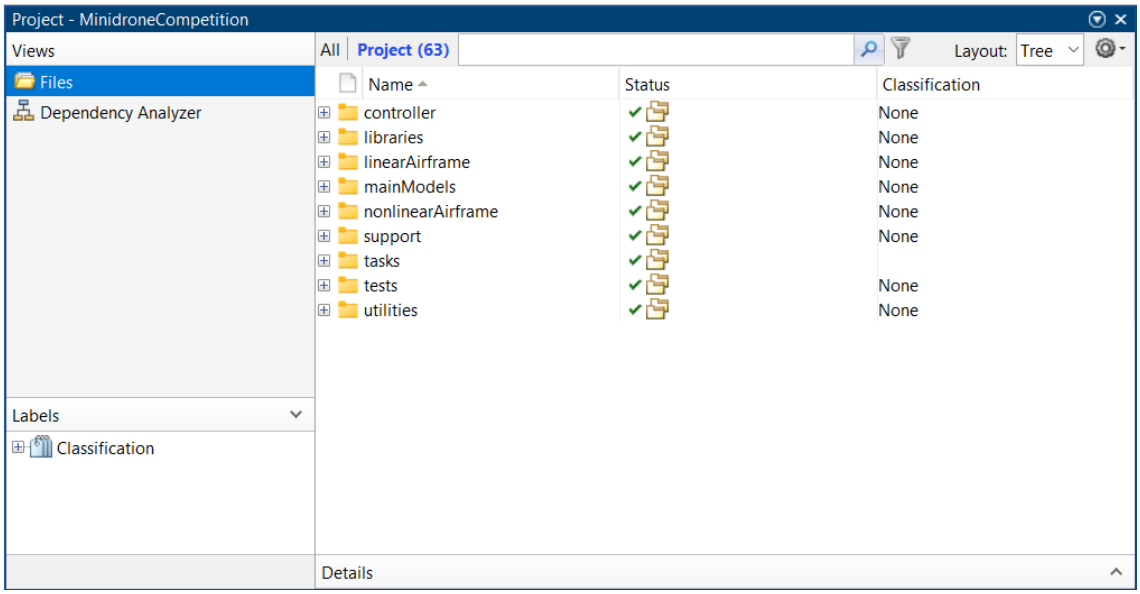
4.1.1. Quadcopter Flight Simulation Model - Mambo

UAV toolbox es una herramienta que permite diseñar, construir e implementar algoritmos de control de vuelo para drones. Donde es posible encontrar una amplia variedad en cuanto a sensores que los drones tienen integrados por defecto. En este caso, ya que se trabaja con el mini dron parrot Mambo es posible acceder a sensores como acelerómetro, giroscopio y sensores ultrasónicos, así como de presión de aire, adicionalmente es posible acceder de las imágenes capturadas por su cámara.

El *Quadcopter Flight Simulation Model - Mambo* es un modelo en Simulink que permite navegar con el Mambo por un entorno el cual puede ser modificado o rediseñado, lo principal a tener en cuenta para correr este modelo es contar con ***Parrot Drone Support Package from MATLAB*** el cual proporciona las interfaces necesarias para controlar el mini dron así como medir cada uno de sus sensores en tiempo real. Una vez instalado este Toolbox es posible abrir el modelo introduciendo la siguiente función en el Command Window "parrotMinidroneCompetitionStart", al ejecutarse, se abre el proyecto llamado *MinidroneCompetition* el cual contiene diferentes carpetas necesarias para que puedan

funcionar correctamente los sensores y actuadores del dron, así como su visualización en e entorno, estas carpetas se logran ver a continuación 4.1.

Figura 4.1: Project - MinidroneCompetition

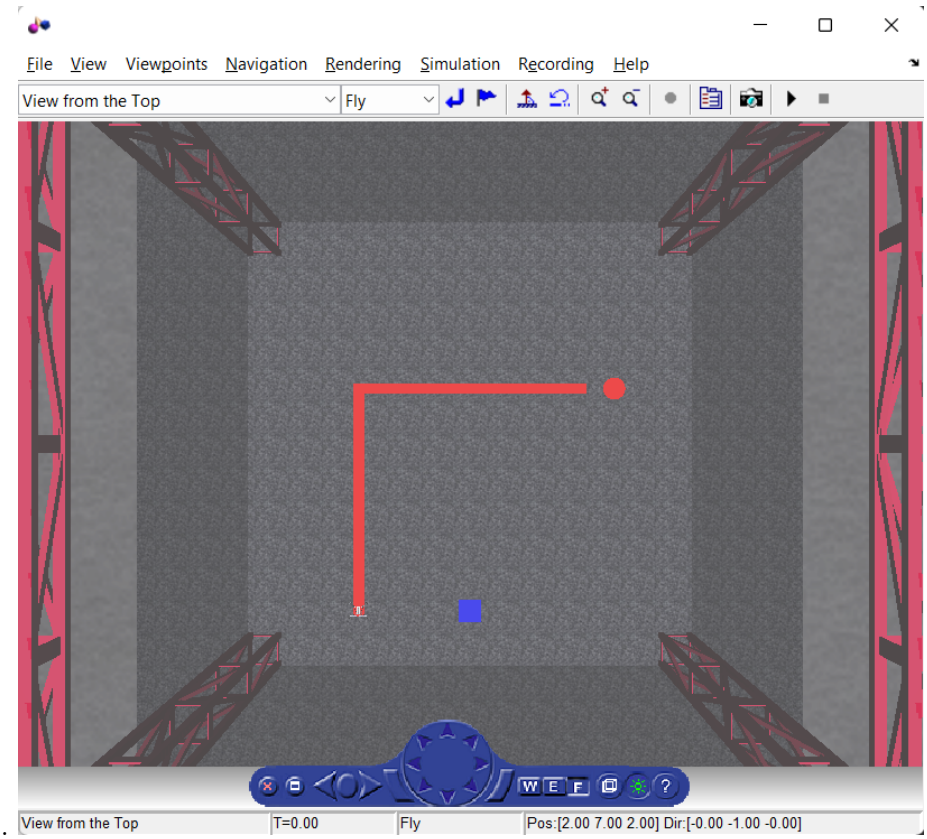


Este template permite implementar el control de vuelo para el minidron modelo Mambo de la marca Parrot [4]. Los métodos de control para este dispositivo que proporciona esta plantilla son 4, por joystick, por datos predefinidos en formato .mat, por datos predefinidos de una hoja de cálculo y por señales, este último es el usado en la implementación del siguiente proyecto, debido a que las otras opciones se descartaron. Primero que todo el joystick, ya que hay que tener cierto tipo de habilidades que permitan maniobrar el dron tanto en los tres ejes de translación como en los tres de rotación durante la trayectoria y además mantenerlo estable para poder hacer la correcta captura de imágenes, también se descarta el control por inserción de datos (ya sea en formato .mat o en hoja de cálculo) porque lo que se busca es diseñar un sistema de control de vuelo que se pueda adaptar a diferentes escenarios sin la necesidad de diseñar un archivo de datos para cada escenario diferente, sin embargo, de una forma fácil es posible alternar entre estos diferentes métodos para realizar pruebas en una posible implementación en campo.

Una vez termina de ejecutarse el comando, dos ventanas adiciones son abiertas, en primer lugar el modelo *Quadcopter Flight Simulation Model - Mambo* 4.4, y también la interfaz gráfica que permite ver la pista y el dron en la parte inferior como es posible apreciar en la

figura 4.2.

Figura 4.2: Interfaz Simulación



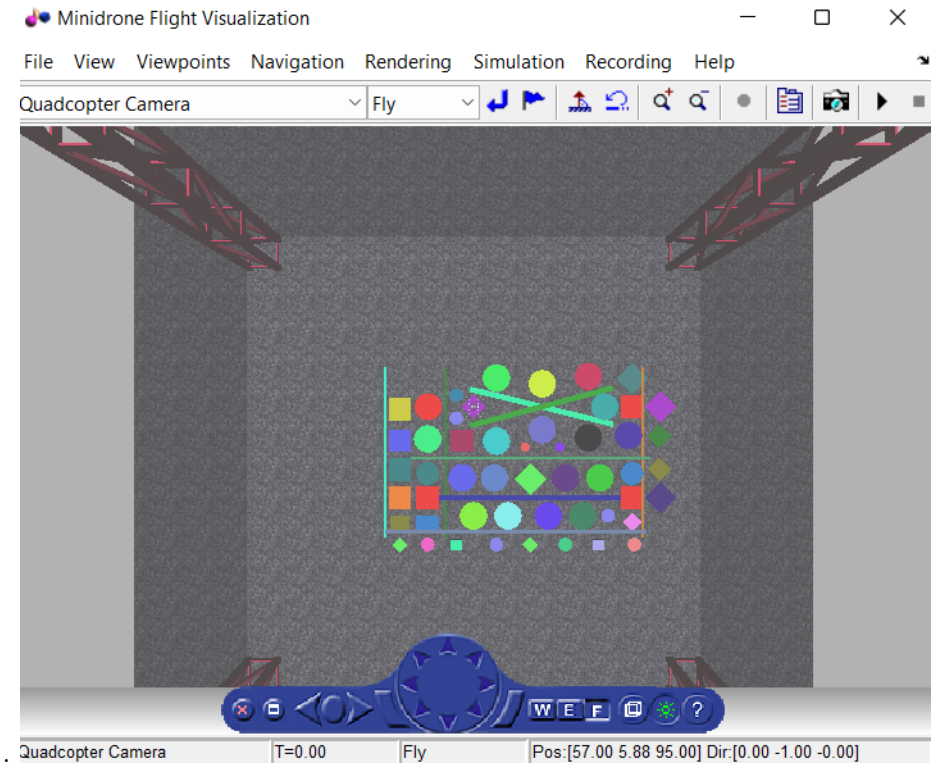
4.1.2. 3D World Editor

Es necesario modificar esta pista para implementar el algoritmo de unión de imágenes, ya que el suelo cuenta con patrones repetitivos en la mayor parte de visualización que es capturada por la cámara, es decir, en la forma en que se ve la pista por ahora cuenta con demasiadas características en común, las mismas que pueden confundir al algoritmo de stitching al momento de unir las imágenes. Teniendo claro esto, el escenario es modificado de manera que tiene ubicados diferentes tipos de formas geométricas a lo largo de la pista, la cuales ayudan a identificar diferencias significativas entre cada una de las imágenes capturadas por el dron.

El desarrollo del nuevo escenario se realiza con ayuda del editor *3D World Editor*, una vez es ajustado con las suficientes figuras para reconocer correctamente las características en cada una de las imágenes, se obtiene la escena de la figura 4.3, estas figuras son de dos dimensiones,

de manera que no cambian su forma no cambia desde ángulos diferentes de captura del dron, para no dificultar el reconocimiento de coincidencias a la hora de unir las imágenes, es posible apreciar que la posición del dron es modificada para tenerlo ubicado en una parte mucho más central del escenario.

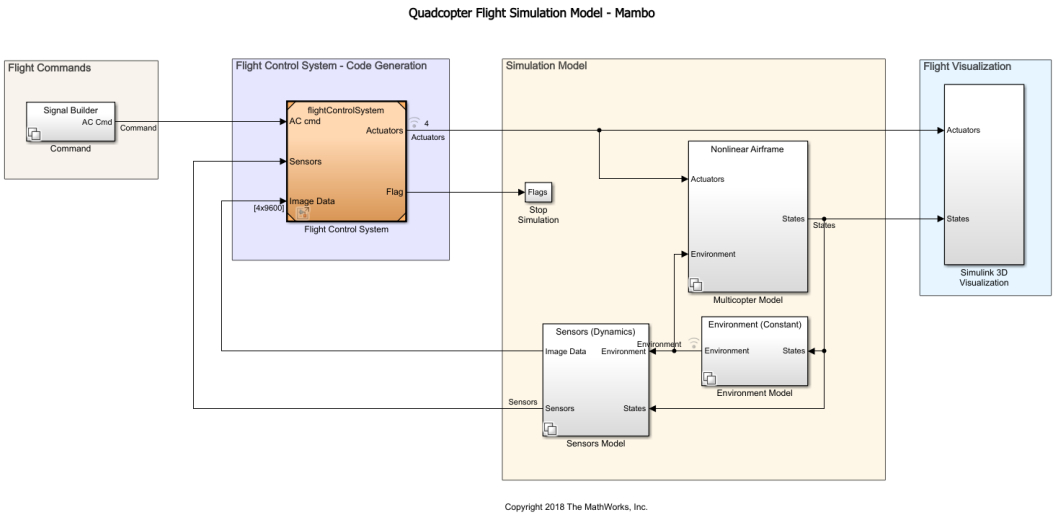
Figura 4.3: Escenario Virtual



4.2. Trayectoria de Vuelo

En el *Quadcopter Flight Simulation Model - Mambo* de la figura 4.4 se logran apreciar 6 subsistemas que conforman el sistema subactuado de seis grados de libertad donde cada uno tiene una función especifica; el bloque que determina la trayectoria del dron (Command), el bloque encargado del control de vuelo del dron (Flight Control System), el bloque que permite detectar magnitudes físicas necesarias para implementar el sistema de control (Sensors Model), el bloque de modelamiento dinámico no lineal del dron (Multicopter Model), un bloque más es el encargado de modelar el ambiente por el cual se va a trasladar el dron (Environment) y por último el bloque a cargo de la visualización 3D tanto del ambiente como del vuelo del dron (Flight Visualization).

Figura 4.4: Quadcopter Flight Simulation Model - Mambo

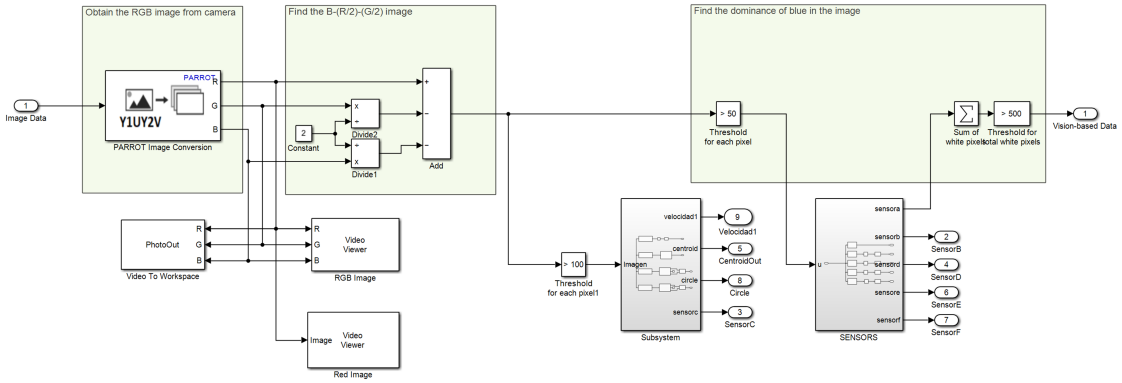


4.2.1. Visualización de imágenes

En el subsistema *Flight Control System* se aprecia que lo constituyen dos diferentes subbloques.

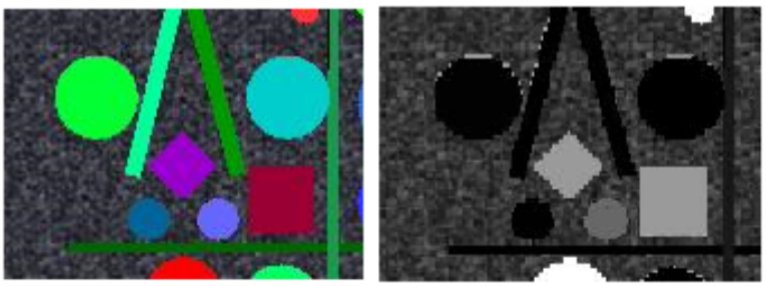
En primer lugar, se encuentra el bloque de procesamiento de imágenes capturadas por la cámara que se localiza en el dron. Estos datos suministrados al sistema están dados en el espacio de color YUV y gracias al bloque *PARROT Image Conversion* es posible realizar la conversión a RGB ya que de esta manera es mas fácil tanto para trabajar en el sistema de control y procesamiento en el algoritmo de stitching como para visualizar los los datos obtenidos. Con el bloque **Video To Workspace** es posible almacenar periódicamente las imágenes en el Workspace de MATLAB, específicamente en la variable `PhotosOut` a la que es posible acceder fácilmente desde el Command Window o directamente desde un Script.

Figura 4.5: Bloque de Procesamiento de Imágenes



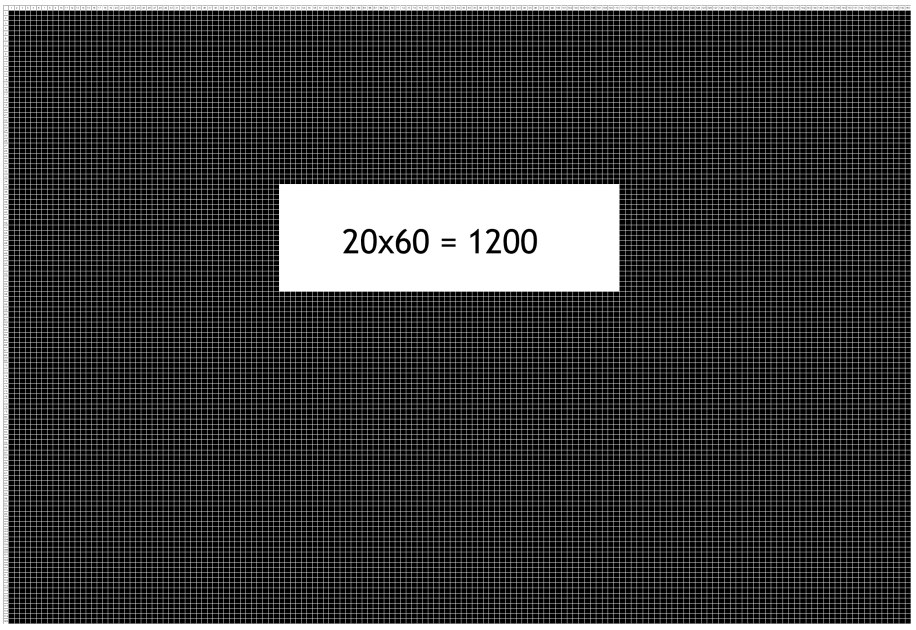
En la figura 4.8 se aprecian las dos imágenes, la primera es la que proporciona la cámara del dron después de la conversión a formato RGB, esta matriz MATLAB la percibe como una matriz de 120x160x3 y en segunda posición es la imagen que se va a tomar como referencia para realización del control de la trayectoria del dron, donde se desprecian los pixeles de color G (verde) y B (azul) y es posible observar la intensidad de R (rojo) en la imagen original, esta imagen MATLAB la percibe como una matriz de 120x160, lo que intrínsecamente revela la resolución de la cámara, es decir, una resolución de 120x160 píxeles.

Figura 4.6: RGB vs Intensidad



La sección de esta matriz que se toma como punto de referencia para realizar el control de vuelo es la que se ve en color blanco de la siguiente figura, la cual es llamada sensorA en la etapa de procesamiento de imágenes y RedDetector en la etapa de control de vuelo,

Figura 4.7: Visualización cámara del dron



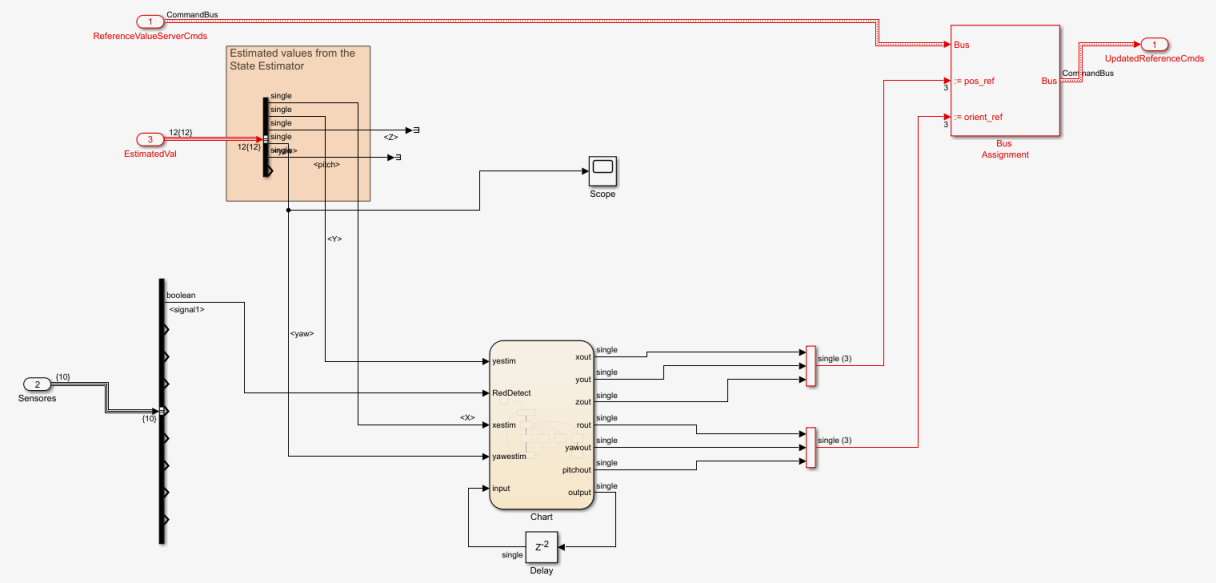
en la figura siguiente también es posible observar una función, en la cual se aprecia que rango de píxeles se selecciona como zona de interés para el control de vuelo, la cual va desde la fila 35 hasta 55 y el rango de columnas de la 50 a la 110.

Figura 4.8: Función sensorA/RedDetect

```
Function sensorA = fcn(u)
   sensorA = u(35:55,50:110);
end
```

En segundo lugar, al mirar con más detalle el otro bloque *Control System* que compone al *Sistema de control de vuelo* se observa el *Planificador de trayectorias* donde realiza ajustes en la trayectoria del dron en los 6 grados de libertad, tanto en traslación, como en inclinación, con base en los datos obtenidos del bloque de procesamiento de imagen (ver figura 4.9).

Figura 4.9: Planeador de trayectorias



Como se puede ver en la figura 4.5, la sección del centro tiene las tres entradas que hacen relación a la señal de la cámara en formato RGB, en este bloque se separa el color rojo, dando como salida una imagen en escala de grises que permite identificar cuanta cantidad de intensidad del color rojo se encuentra en cada uno de los píxeles. Esto se hace realizar el control de vuelo por medio de detección de color, en este caso el rojo; sin embargo, es posible realizarlo con cualquiera de los tres colores, posterior a este filtrado se procede realizar una sumatoria para saber el número de píxeles que identifica como unos para utilizar esa información posteriormente en el planificador de trayectorias.

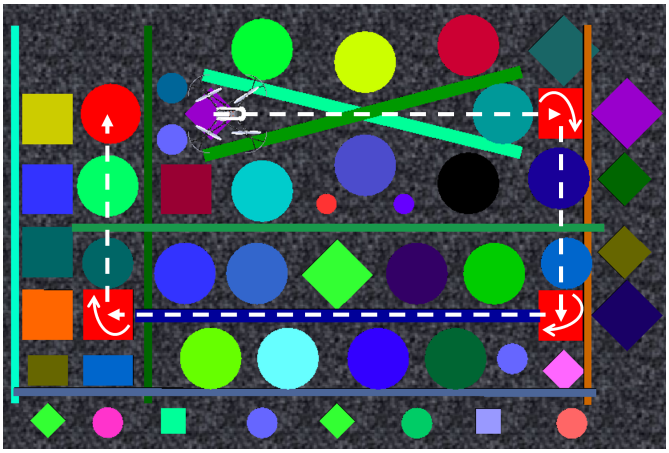
Una vez la señal es leída por el planificador de trayectorias (figura 4.9) y el algoritmo conoce el número de píxeles que están en rojo (o en 1 después de la filtración de color) esta información es utilizada para implementar la máquina de estados o stateflow como lo denomina MATLAB.

4.2.2. Sistema de Control de vuelo

La forma en que se desarrolla el sistema de control de trayectoria consiste en que el dron debe rodear un área de forma rectangular. Para realizar esto, se seleccionan 5 puntos de interés en la pista de vuelo, el primero es el punto inicial donde el dron se eleva e inicia el recorrido en línea recta, a medida que avanza el dispositivo se encuentra con tres puntos ubicados

estratégicamente los cuales van a tomarse como referencia para que gire 90 grados hacia la derecha y para posteriormente seguir con su recorrido, estos tres puntos se diferencian de las otras figuras porque son los cuadrados rojos y ubican en los extremos de la pista y por último, el sitio de aterrizaje también se diferencia por ser un círculo rojo ubicado cerca al punto de salida; esto se puede percibir en la siguiente imagen, donde también se hace un esquema de la trayectoria del dron.

Figura 4.10: Esquema trayectoria de vuelo



En la figura 4.9 es posible apreciar las entradas y salidas que tiene la máquina de estados;

ENTRADAS

- xestim: posición actual estimada del dron en el eje x.
- yestim: posición actual estimada del dron en el eje y.
- RedDetect: detección del color rojo en la zona seleccionada.
- yawestim: ángulo de rotación estimado del dron en yaw.
- input: variable de cambio de ángulo de rotación cada vez que el dron debe girar.

SALIDAS

- xout: desplazamiento en el eje x.
- yout: desplazamiento en el eje y.
- zout: desplazamiento en el eje z.

- pitchout: rotación con respecto al eje x.
- yawout: rotación con respecto al eje y.
- rout: rotación con respecto al eje z.
- output: variable de cambio de ángulo de rotación cada vez que el dron debe girar.

La lógica de la máquina de estados tiene como base 8 estados, los cuales se puede observar en la primera tabla de la figura 4.11 donde como se menciona anteriormente, nos encontramos con el despegue, avance, aterrizaje y cada uno de los giros para completar el recorrido. En cuanto a las transiciones, se evalúan dos parametros principalmente, en primer lugar, si la cámara detecta la precedencia de color rojo en la sección definida anteriormente, si esto se cumple la entrada `RedDetect` tomará el valor de uno, de lo contrario será cero, la segunda entrada que se evalúa en las transiciones es `input` la cual empezará en cero y una vez realiza los giros irá incrementando sucesivamente, hasta tomar el valor de 3, ya que el dron realiza tres giros a lo largo del trayecto, las transiciones no solo manejan condiciones, hay otras que se utilizan para crear retardos entre un movimiento y otro, para evitar que el dron se mueva de manera brusca y pierda estabilidad. Esta lógica se puede ver de manera detallada en la figura 4.12 y la implementación en MATLAB es posible verla en la figura 4.13.

Figura 4.11: Estados y transiciones

ESTADO	DESCRIPCION
E1	DESPEGUE
E2	AVANCE
E3	PAUSA
E4	GIRO DE 90°
E5	GIRO DE 180°
E6	GIRO DE 270°
E7	ATERRIZAJE
E8	AVANCE

TRANSICION	PROCESO
T1	si(RedDetect == 0)
T2	si(RedDetect == 1)
T3	si(input == 0)
T4	si(input == 1)
T5	si(input == 2)
T6	si(input == 3)
T7	delay 0.5 seg
T8	delay 0.5 seg
T9	delay 0.5 seg
T10	delay 1 seg & si(RedDetect == 0)

Figura 4.12: Esquema máquina de estados

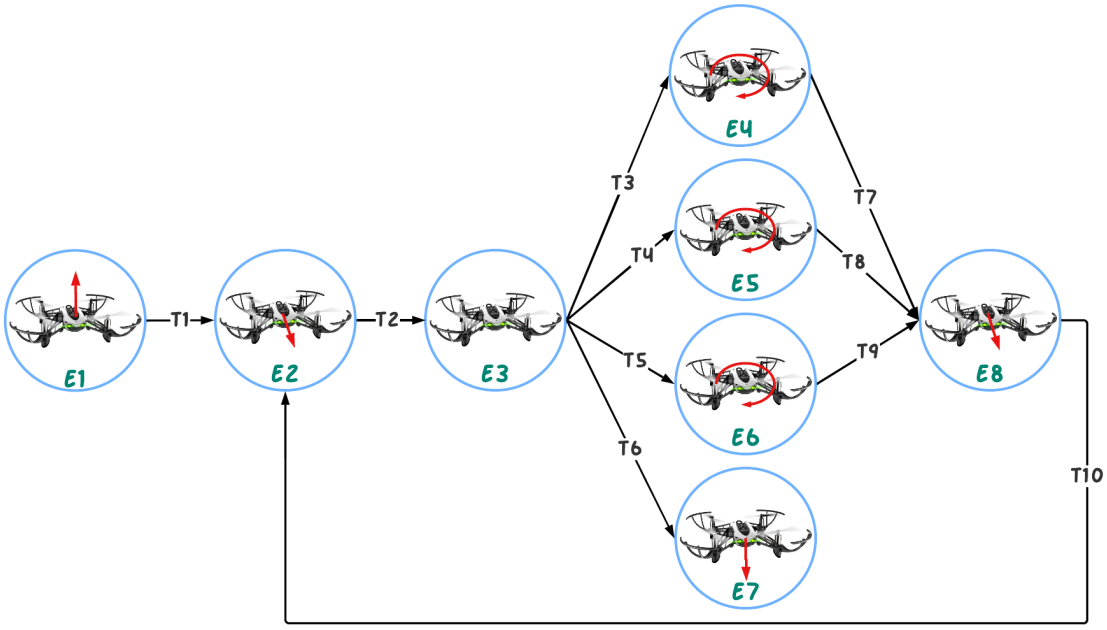
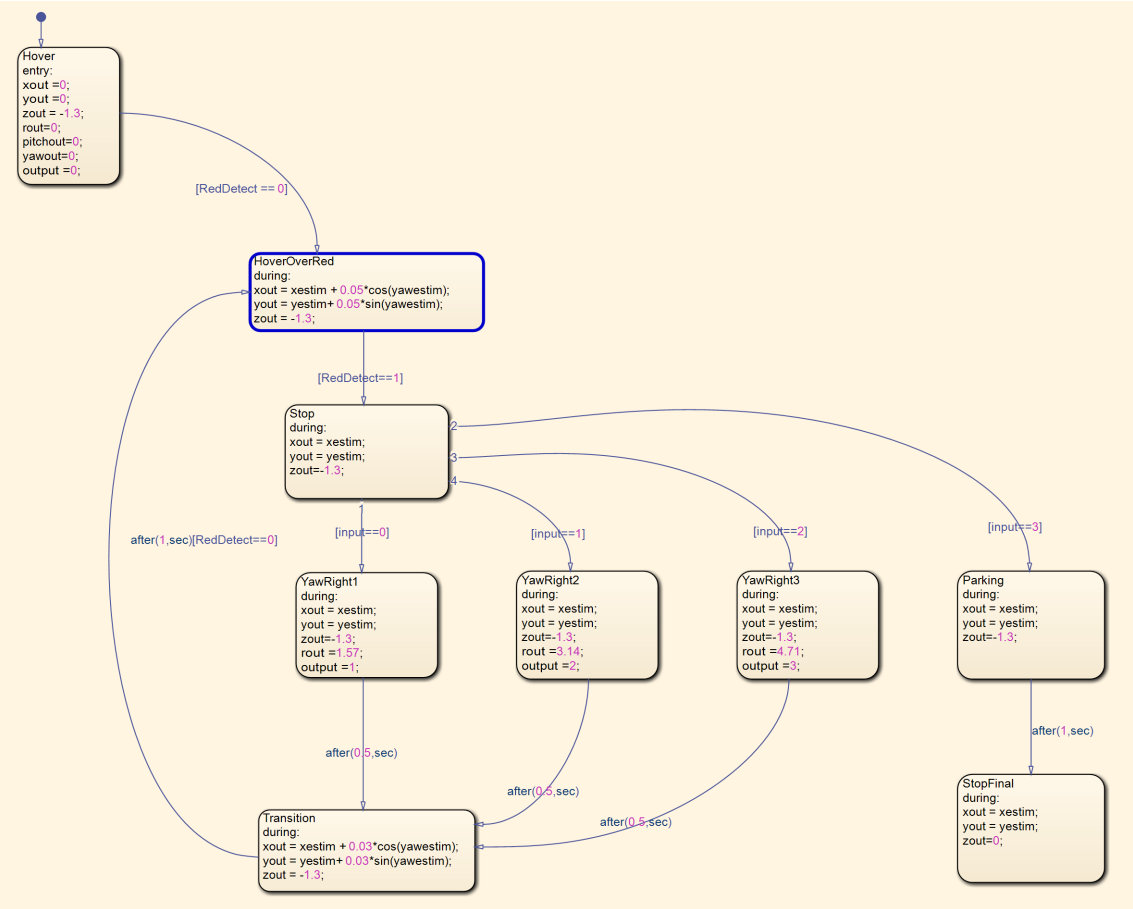


Figura 4.13: Máquina de estados

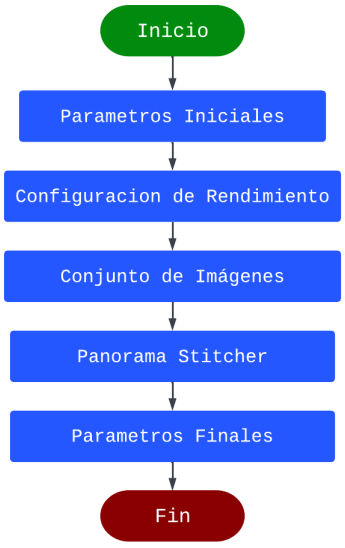


Capítulo 5

Algoritmo de Stitching

En este capítulo se describe una a una las fases para el diseño, desarrollo e implementación del algoritmo de Stitching Automático, el cual se distribuye en cinco secciones principales; Parámetros Iniciales, Configuración de Rendimiento, Conjunto de Imágenes, Panorama Stitcher y Parámetros Finales. A continuación se describen detalladamente las secciones:

Figura 5.1: Algoritmo General Stitching

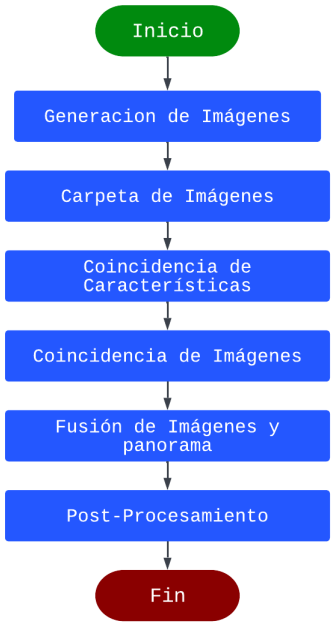


.

5.1. Parámetros Iniciales

En la primera sección *Parámetros Iniciales*, se llama al script `InitialParameters.m` con el fin de realizar dos principales tareas. La primera consiste en exportar los datos tomados por la cámara del dron en formato de imágenes, ya que como se mencionó anteriormente, estas se encuentran en formato de matrices en el Workspace con el nombre de PhotoOut, el número de imágenes capturadas a lo largo del trayecto es de 121; sin embargo, este número puede ser cambiado. Y la segunda tarea de `InitialParameters.m` consiste en crear variables auxiliares que van a ser de ayuda en futuras secciones, por ejemplo, en la primera variable se almacena la localización de la carpeta de donde serán extraídas las imágenes capturadas por el dron durante la trayectoria del dron por la pista, otras variables almacenan el método de coincidencia de características. En la siguiente figura se muestra un esquema donde se observan la forma en que se divide este script y cada una de estos fragmentos contienen variables que serán útiles en secciones posteriores este algoritmo.

Figura 5.2: Parámetros Iniciales



.

5.2. Configuración de Rendimiento

En esta sección, con ayuda de la herramienta **Parallel Computing Toolbox** se mejora el rendimiento del algoritmo, haciendo que algunos partes de él trabajen simultáneamente, en la sección 3.5 se detalla el funcionamiento de esta herramienta. Una vez nos aseguramos de que el Parallel Pool esté activado, se inicializa un cronómetro con el fin de contabilizar el tiempo en que el algoritmo genera la imagen panorámica.

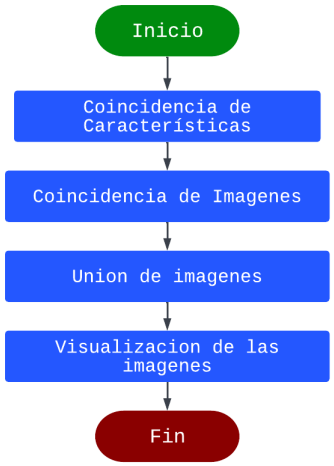
5.3. Conjunto de Imágenes

En esta sección se toma como referencia la carpeta en la que se almacenaron las imágenes capturadas y se crea un vector con tres propiedades, el nombre de la carpeta donde está almacenado el conjunto de fotografías, la dirección de almacenamiento de cada y también el número de imágenes encontradas en esa carpeta.

5.4. Panorama Stitcher

En esta sección se realiza la unión de imágenes y se divide en 4 etapas, la extracción y coincidencia de características en cada una de las imágenes, la coincidencia de entre imágenes, la unión de imágenes y por último la visualización de la imagen final, como se describe en la siguiente figura [24].

Figura 5.3: Esquema creación de panorama



.

5.4.1. Coincidencia de Características

La función `featureMatching`, encargada de detectar las características en cada una de las imágenes tiene como parámetros de entrada:

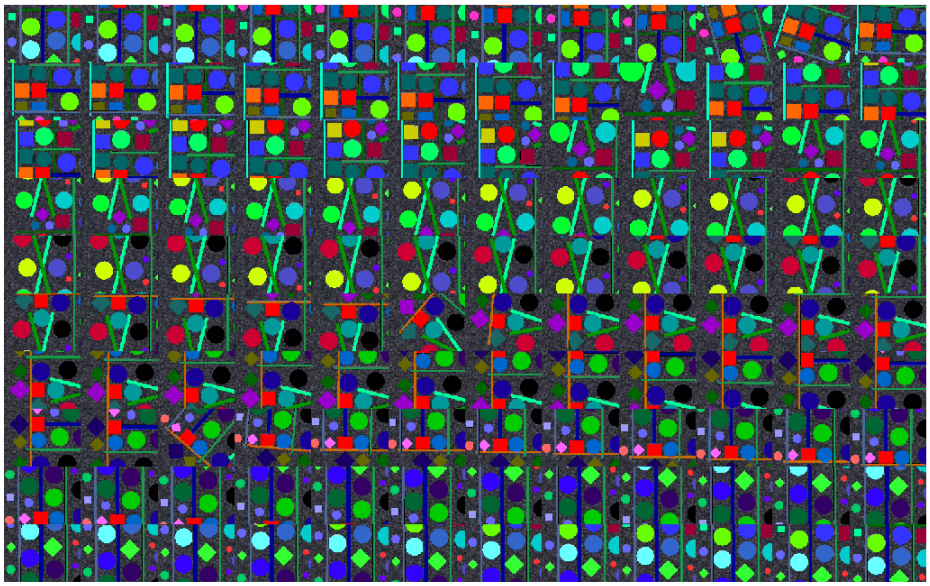
- `imgSetVector`:conjunto de ubicaciones de las imágenes

y como parámetros de salida:

- `keypoints`: Puntos clave de cada imagen
- `allDescriptors`: Características de cada imagen
- `images`: Conjunto de imágenes en RGB
- `imagesinfo`: Inicialización matriz celda de 1xn
- `imageFocals`: Inicialización matriz de ceros 1xn
- `n`: Número de Imágenes

El primer paso en esta parte es agrupar el total de las imágenes en una sola figura (ver figura 5.4). Por otro lado, se obtienen las características SURF (ver sección 3.3.1.2) del conjunto de imágenes, se selecciona este método de detección de características, ya que es robusto frente a transformaciones de imágenes (desenfoque, cambios de rotación, cambios de escala y cambios de iluminación).

Figura 5.4: Montaje imágenes originales



.

Lo que hace SURF es localizar los puntos de interés (x, σ) en la imagen, es decir, las áreas que presentan una variación y la forma en que funciona este detector, es basándose en la *Matriz Hessiana*:

$$\mathrm{H}(x,\sigma) = \begin{pmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{yx}(x,\sigma) & L_{yy}(x,\sigma) \end{pmatrix} \tag{5.4.1}$$

Donde $L_{xy}(x, \sigma)$ es el producto de la imagen filtrada por la segunda derivada de una gaussiana de tamaño σ (ver 3.3.2) [32].

$$L_{xy}(x,\sigma) = I(x) * \frac{\partial^2}{\partial xy} g(\sigma) \tag{5.4.2}$$

Una vez se halla el determinante de H, se obtiene el cambio local al rededor de un punto x, para este caso como queremos tener 20 características a comparar entre cada una de las imágenes, se escogen los primeros 20 valores que expresen un cambio local mayor con respecto al resto del contenido de la imagen y estos puntos serán denominados características SURF. Todo este proceso es necesario realizarlo de forma manual, MATLAB tiene herramientas especializadas

en procesamiento digital de imágenes y la función `detectSURFFeatures` retorna el objetivo `SURFPoints`, el cual contiene información sobre las características SURF detectadas en la imagen de entrada en escala de grises 2D [44].

5.4.2. Coincidencia de Imágenes

La función `imageMatching`, tiene como objetivo encontrar las imágenes coincidentes, los parámetros de entrada corresponden a:

- `parameters`: variables creadas en `InitialParameters`
- `n`: Número de Imágenes
- `images`: Conjunto de imágenes en RGB
- `keypoints`: Puntos clave de cada imagen
- `allDescriptors`: Características de cada imagen

y como parámetros de salida se tienen:

- `allMatches`: Celda de coincidencias válidas
- `numMatches`: Matriz de coincidencias válidas
- `initialTforms`: Matriz de transformación geométrica

Inicialmente, se calcula la homografía (H) usando el método de transformación geométrica, en este caso se realiza una transformación afín, sin embargo, de igual manera también es posible realizarlo con las otras transformaciones vistas en la sección 3.3.3.1, en este caso MATLAB permite realizar esta transformación con la función `affine2d`. Para la implementación de RANSAC se define un número de iteraciones igual a 500 ($k = 500$), se escoge el número de características a comprar como 4 ($n = 4$) y se estima la cantidad de Inliers en el 70 %($\omega = 0{,}7$), sin embargo, el número de iteraciones es suficientemente grande como para que el algoritmo RANSAC tenga una probabilidad del 99 % de encontrar un modelo apropiado.

Figura 5.5: Algoritmo RANSAC

Data: Number of inliers we use each iteration n, maximum number of iterations m, input data *data*, threshold for determining a good fit t, number of close points required for a good model fit k
Result: Model $bestFit$
$i \leftarrow 0$;
$bestFit \leftarrow null$;
$bestError \leftarrow infinite$;
while $i < m$ **do**
  $tempInliers \leftarrow$ selectRandom$(data, n)$;
  $tempInliersAdd \leftarrow empty$;
  $tempModel \leftarrow$ fit$(tempInliers)$;
  for *point in data* **do**
    if *point not in tempInliers* **then**
      if $distance(point, tempModel) < t$ **then**
        $tempInlierAdd$.add$(point)$;
      end
    end
  end
  if *length(tempInliersAdd)* $> k$ **then**
    $newModel \leftarrow$ fit$(maybeInliers$ and $tempInliers)$;
    if $newModel.error > bestError$ **then**
      $bestError \leftarrow error$;
      $bestFit \leftarrow betterModel$;
    end
  end
end

Fuente: [35]

El bucle for se selecciona una muestra de tamaño n de datos aleatorios, se ajusta el modelo basándonos en estos datos. Una vez ajustado este modelo, se comprueba qué tan alejados se encuentran estos puntos del conjunto de datos original con respecto al modelo en el segundo ciclo del bucle for. Para esta distancia se usa el parámetro t como umbral y partir de estos puntos, se crea una nueva lista llamada *tempInlierAdd*. Si esta lista tiene un tamaño mayor al de (k), se considera candidata para un modelo adecuado. Luego creamos un modelo a partir de *tempInlier* y *tempInlierAdd* y lo comparamos con el mejor modelo de nuestras iteraciones anteriores [35].

A continuación implementa el modelo probabilístico de verificación de coincidencias. Para cada par de imágenes potencialmente correspondientes tenemos un conjunto de características geométricamente consistentes (RANSAC Inliers) y un conjunto de características que están dentro del área de solapamiento, pero no son geométricamente consistentes (RANSAC Outliers). La forma en que implementar este modelo es la función auxiliar `refineMatch`.

En la matriz `numMatches` se encuentran almacenados los Inliers que relacionan a cada una de las imágenes, en la figura 5.6 es posible apreciar gráficamente el contenido de esta matriz, donde por simplicidad se muestra únicamente la relación de coincidencia que tienen las 7 primeras imágenes, lo primero que se puede observar es que la diagonal marcada de color verde es el orden correcto en que deberían ir ubicadas las imágenes, es decir, las posiciones donde deberían estar los valores más altos; sin embargo, no es así en todos las columnas, esto es a causa de que las imágenes están tomadas con un periodo de tiempo corto y es por eso que por ejemplo la imagen 1 tiene más coincidencias con la imagen 3 que con la imagen 2; no obstante, esto no es inconveniente porque esto se ajusta en la siguiente sección. Lo siguiente que se puede ver en la figura 5.6 es que los valores por debajo de la diagonal principal marcados de color rojo tienen el valor de cero, esto se hace de forma manual para evitar que haya inconsistencias en la unión de imágenes.

Figura 5.6: Matriz `numMatches`

	Im1	Im2	Im3	Im4	Im5	Im6	Im7
Im1	0	4	7	4	0	0	0
Im2	0	0	13	6	4	6	4
Im3	0	0	0	15	13	7	5
Im4	0	0	0	0	20	11	10
Im5	0	0	0	0	0	20	14
Im6	0	0	0	0	0	0	16
Im7	0	0	0	0	0	0	0

5.4.3. Ajuste de paquete

Esta sección se encarga realizar el ajuste en el orden las imágenes cuando estas tienen un número elevado de coincidencias con dos o más imágenes, la función `bundleAdjustmentLM` tiene como parámetros de entrada:

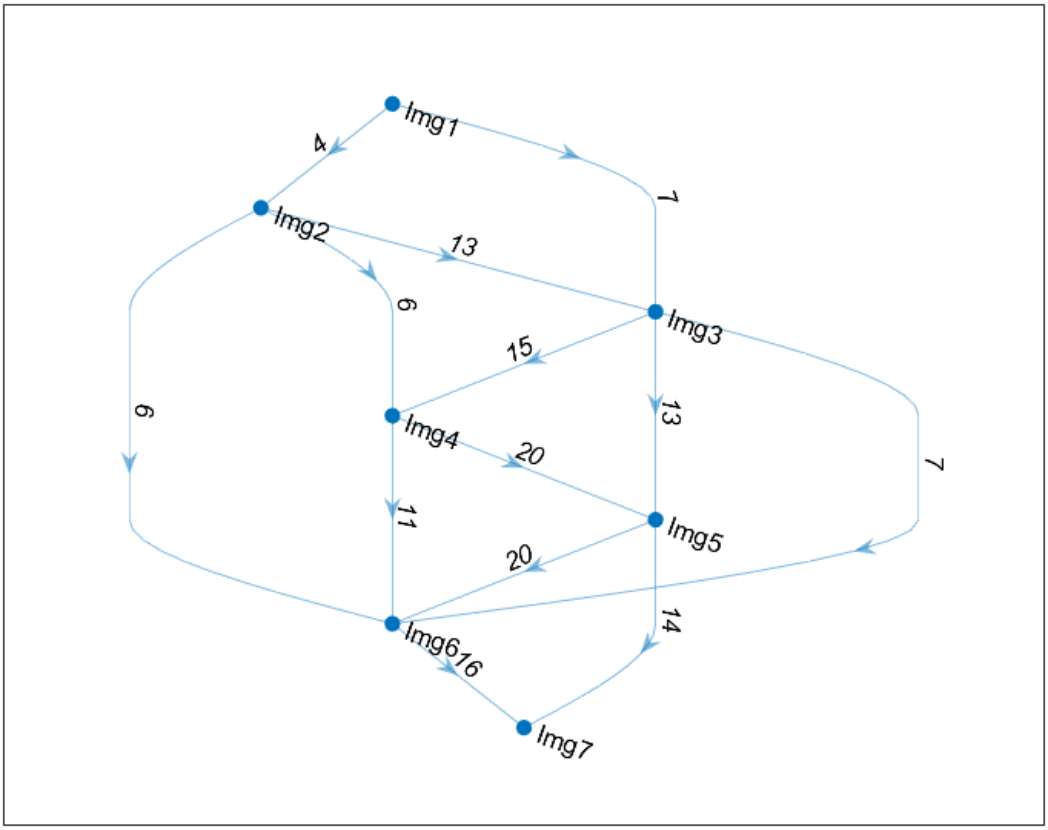
- `images`: Conjunto de imágenes en RGB
- `keypoints`: Puntos clave de cada imagen
- `allMatches`: Celda de coincidencias válidas
- `numMatches`: Matriz de coincidencias válidas
- `InitialTforms`: Matriz de transformación geométrica

y como parámetros de salida:

- `finalPanoramaTforms`: tamaño, ubicación, forma y orden de cada imagen
- `concomps`: Conexión entre cada una de las imágenes

Inicialmente, se obtiene la transformación geométrica de las características y las conexiones de las coincidencias por medio de grafos, en la figura 5.7 se logra apreciar cada uno de los nodos, que representan las imágenes y las aristas que representan las coincidencias con las otras imágenes. De esta manera, se procede encontrar el orden de las imágenes con base en las coincidencias encontradas con ayuda de un árbol de decisión. Inicialmente, con la función auxiliar `getMST` se obtiene una matriz de adyacencia con la mayor extensión del árbol de decisión, basándose en la matriz de adyacencia nombrada `numMatches`. Más adelante, con ayuda de la función `getTforms` se calculan las transformaciones proyectivas para todas imágenes en el componente conectado de la imagen `i` en el árbol de la matriz `tree`, `getPanoramaSize` es otra función auxiliar que permite obtener el tamaño de la imagen panorámica, con ayuda de las transformaciones proyectivas calculadas anteriormente. La función `getOrdering` retorna el orden de los índices especificados, en este caso como las imágenes estaban ordenadas a la hora de obtenerlas de la carpeta, entonces el orden no cambia, pero podría cambiarse el orden de entrada y la función seguiría ordenándolas correctamente.

Figura 5.7: Grafos



.

5.4.4. Visualización de las Imágenes

La función `displayPanorama` tiene como parámetros de entrada:

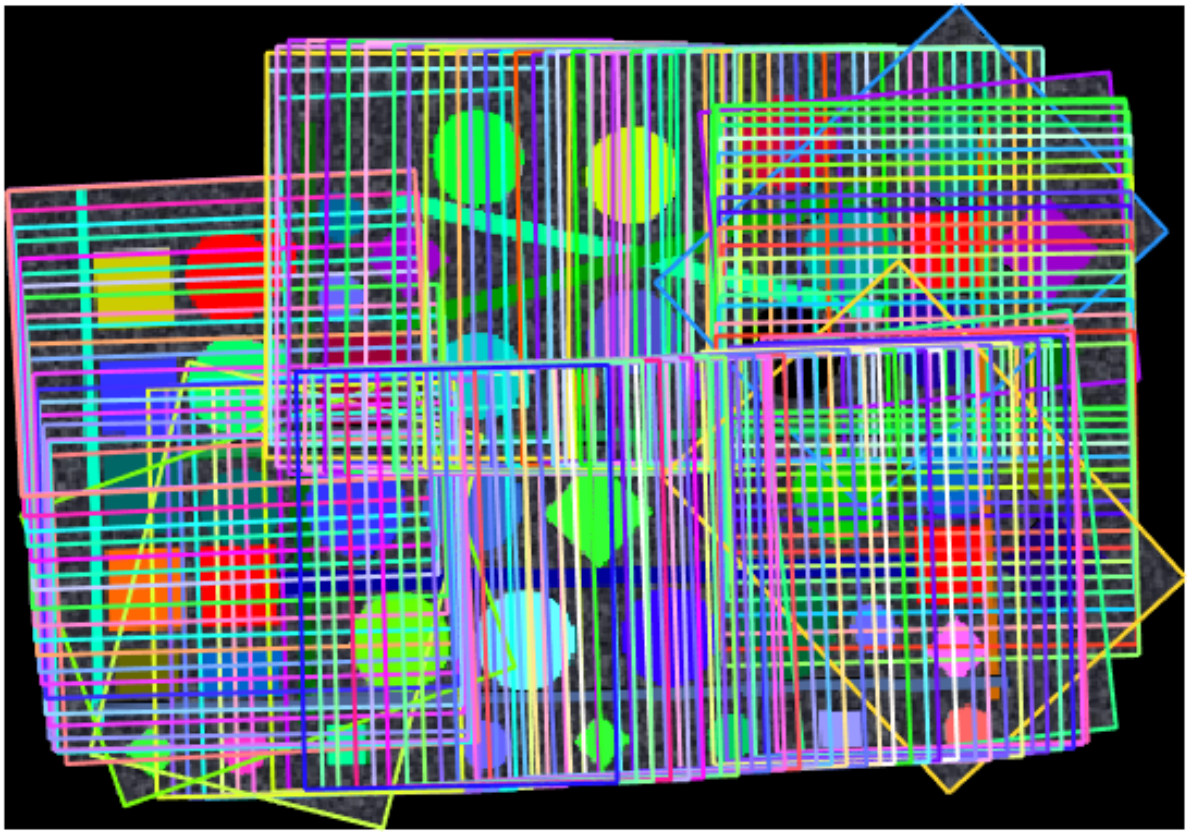
- `finalPanoramaTforms`: tamaño, ubicación, forma y orden de cada imagen
- `InitialParameters`
- `images`: Conjunto de imágenes en RGB
- `imageFocals`: Inicialización matriz de ceros 1xn
- `concomps`: Conexión entre cada una de las imágenes
- `datasetName`: Nombre de la carpeta que contiene las imágenes

y como parámetros de salida:

- `allPanoramas`:
- `croppedPanoramas`:

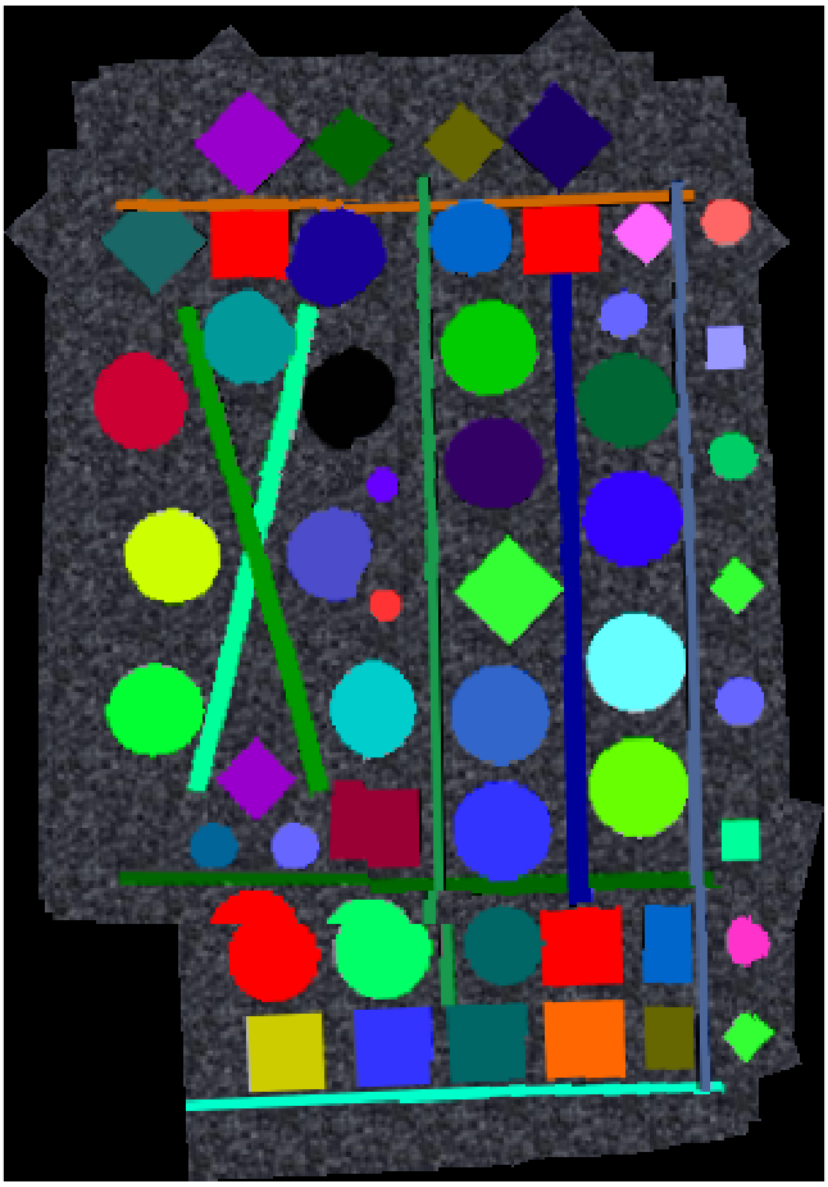
En esta sección se crea y se muestra el panorama con ayuda de la función renderPanorama, donde se crea una imagen vacía y con `outputLimits` se calculan los límites mínimos y máximos de salida de todas las transformaciones. Estos valores se utilizan para calcular automáticamente el tamaño del panorama. Con `imwarp` se realiza la concatenación de las imágenes a imagen vacía basándonos en el índice obtenido de la función anterior (`bundleAdjustmentLM`). Por último, cuando se obtiene panorama se muestra en pantalla el resultado y adicionalmente otra imagen que deja ver donde se ubican cada una de las imágenes originales.

Figura 5.8: Representación de las imágenes unidas



Por último se obtiene la imagen panorámica obtenida por el algoritmo de stirching.

Figura 5.9: Resultado algoritmo



5.5. Parámetros finales

Esta sección final del algoritmo correspondiente al manejo del temporizador para determinar cuanto tiempo tardó en compilar el algoritmo. En la variable `Runtime` se almacena el valor en que se detiene el cronómetro con la ayuda de la función `toc`, recordemos que empezó a contar el tiempo desde el principio del capítulo en la sección *Configuración de rendimiento*, de esta manera se imprime en pantalla el tiempo que le tomó al algoritmo realiza el proceso completo de unión de imágenes, es decir, desde que se tienen las imágenes individuales hasta

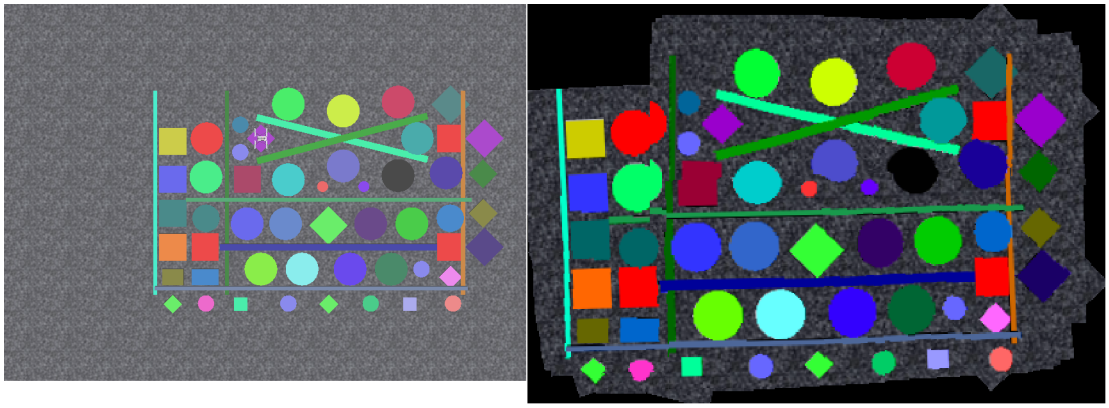
que se observa en pantalla el resultado de la imagen panorámica.

Capítulo 6

Validación

Una vez se genera la imagen con el algoritmo de Stitching, se realiza una captura de la imagen superior del escenario virtual, las dos imágenes se puede ver en la figura 6.1, donde a la izquierda se encuentra la imagen capturada desde el entorno de simulación y a la derecha el resultado de la imagen panorámica obtenida por el algoritmo de stitching.

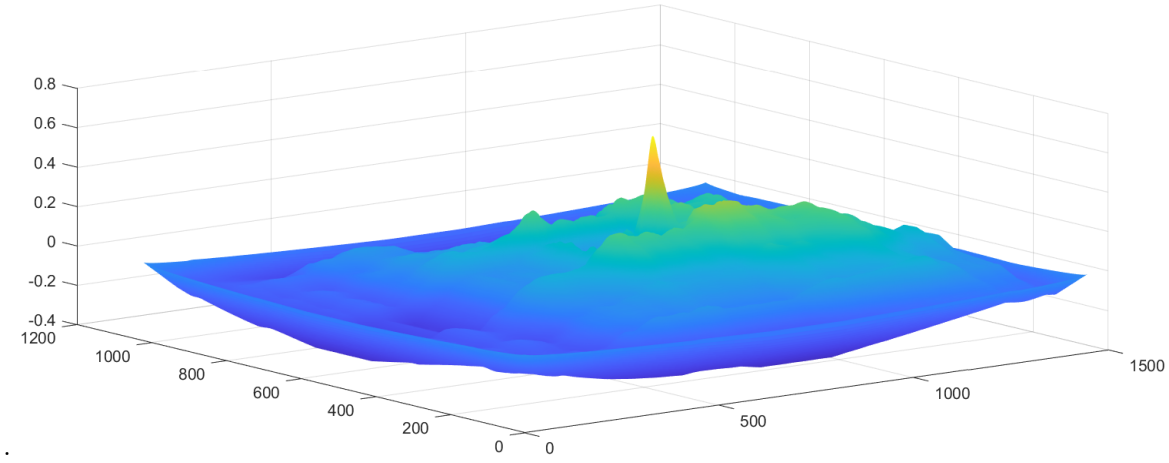
Figura 6.1: Imágenes de validación



Como se mencionó anteriormente, la validación se va a realizar con ayuda de la función que proporciona Matlab para realizar correlación cruzada, la opción que se ve a primera vista en función `xcorr2` ya que con esta es posible encontrar relación entre las dos imágenes y así poder realizar una comparación entre ellas, el problema con esta función es que en este caso en particular no es posible usarla, debido a que las imágenes no son del mismo tamaño y hay que realizar un proceso previo de normalizado a cada una de ellas. La función `normxcorr2`

es mejor para este caso porque realiza la normalización y la correlación cruzada de forma conjunta. La imagen de superficie que se puede extraer de esta función resultante es la que se logra ver en las figuras 6.2, esta es una gráfica que permite observar la intensidad en cada uno de los píxeles de las imágenes, donde se logra observar que existe picos de color amarillo en varios puntos de la imagen, estos corresponden a similitudes entre las dos imágenes.

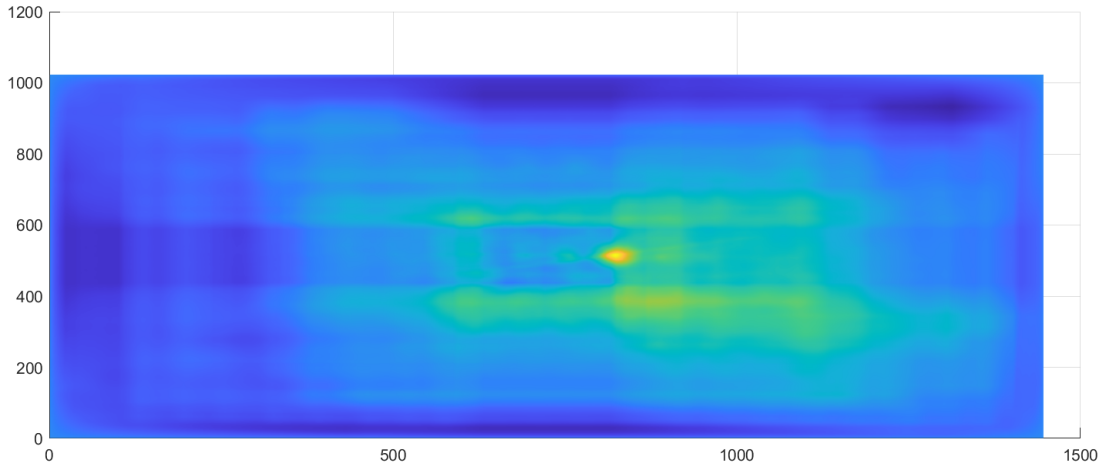
Figura 6.2: Validación cruzada



.

Visto desde la parte superior se aprecia mejor la similitud en cada una de las zonas de la imagen.

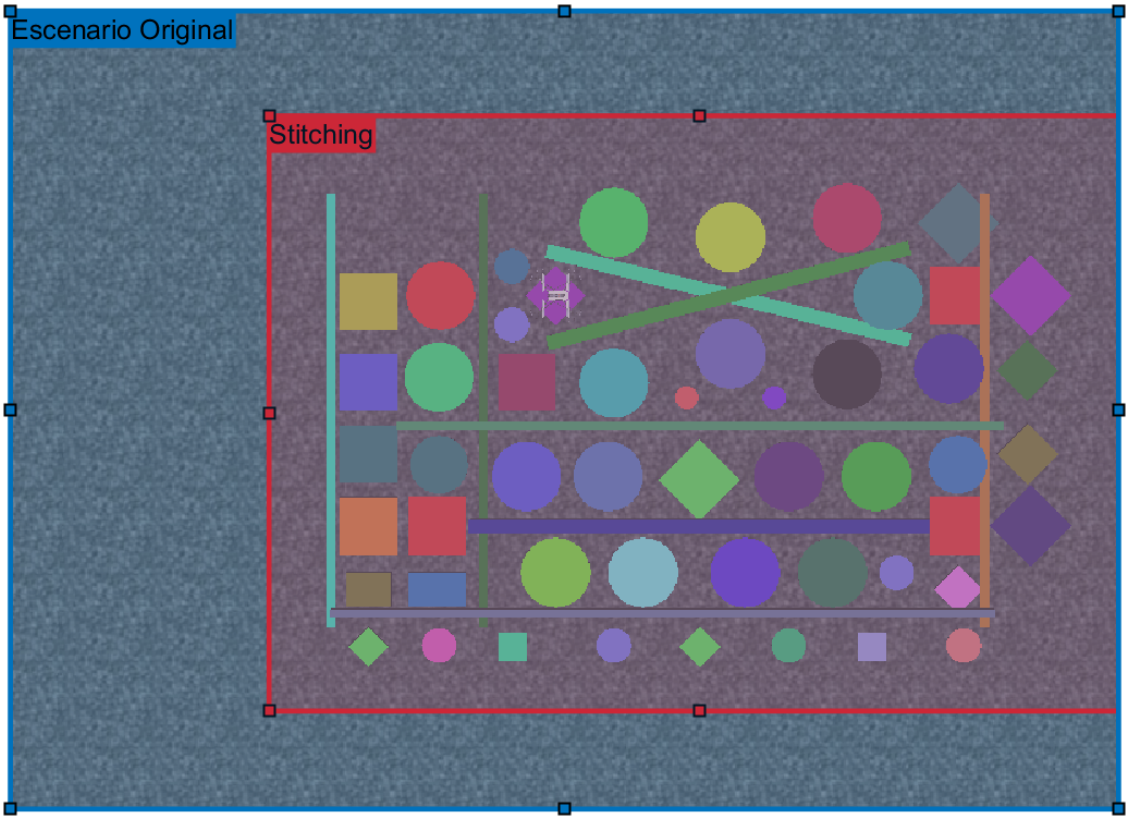
Figura 6.3: Validación cruzada vista superior



Por último, teniendo el resultado de la correlación cruzada, es posible determinar si la imagen del algoritmo pertenece a alguna sección concreta de la imagen captura de la pista, de esta

manera se marca un recuadro mostrando la relación que tienen entre ellas.

Figura 6.4: Validación cruzada comparación



Capítulo 7

Resultados

7.1. Escenario y el plan de vuelo

Gracias al modelo en Simulink *parrotMinidrone*, fue posible eludir ciertos parámetros importantes a la hora de la simulación del vuelo, como el diseño del propio dron, el diseño en la adquisición de datos por medio de los sensores incorporados y el modelamiento las leyes físicas a lo largo de la simulación. De esta manera, con ayuda del 3D Word Editor se realiza el diseño del escenario donde es posible apreciar la pista, por la que va a cruzar el dron:

Figura 7.1: Visualización del escenario

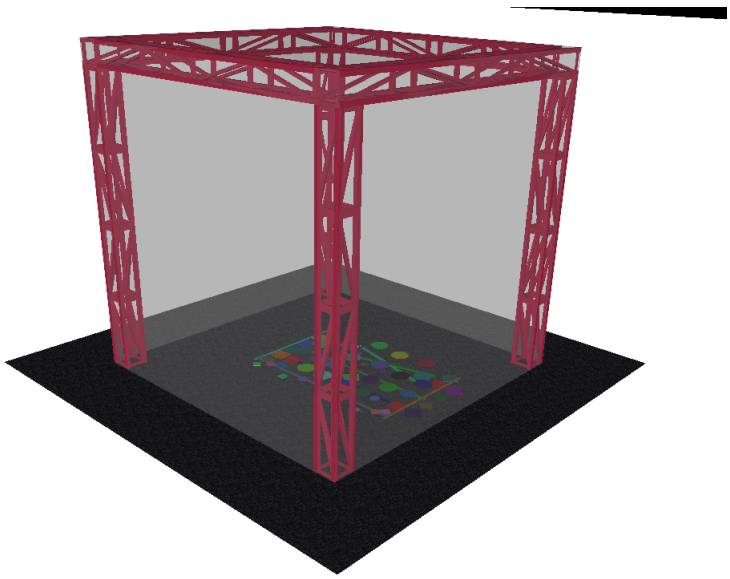


Figura 7.2: Visualización del escenario

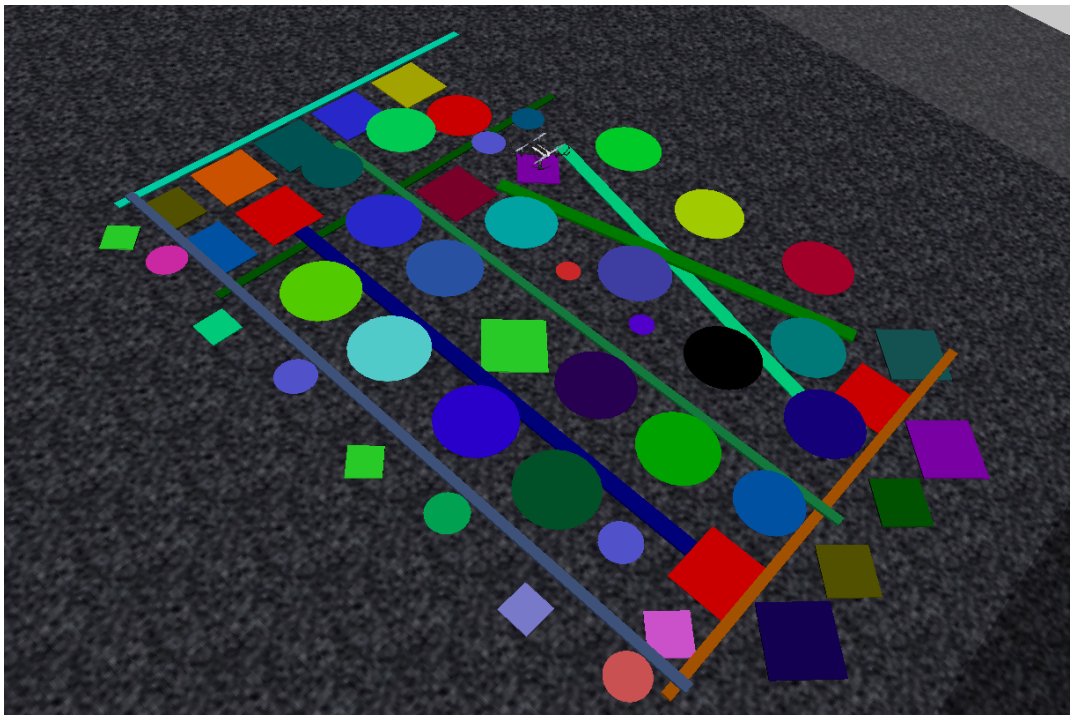
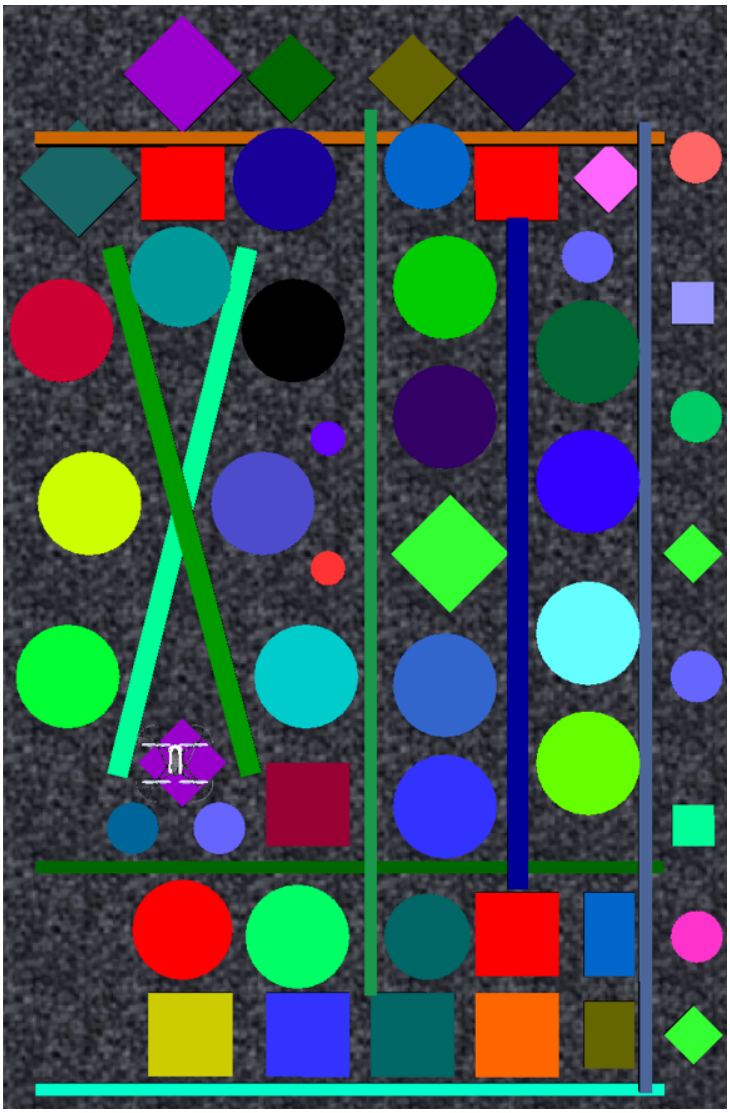


Figura 7.3: Visualización del escenario



El mini-dron realiza el despegue y se comporta a lo esperado en relación con el modelo del plan de vuelo definido inicialmente, es posible determinar que no hay inconvenientes tanto en la sección de detección de imágenes como en la de control de trayectoria.

Figura 7.4: Vuelo del dron

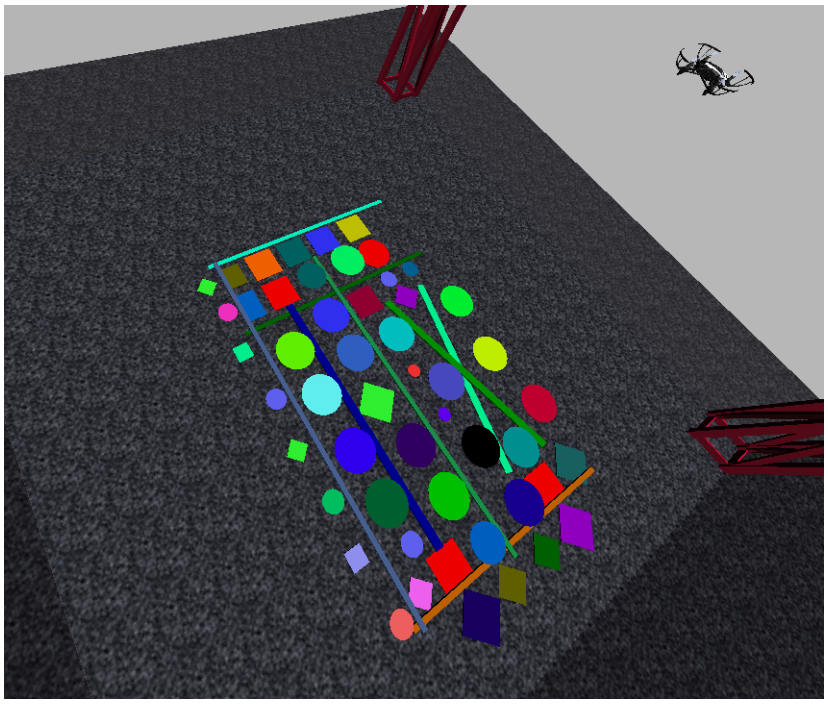


Figura 7.5: Vuelo del dron

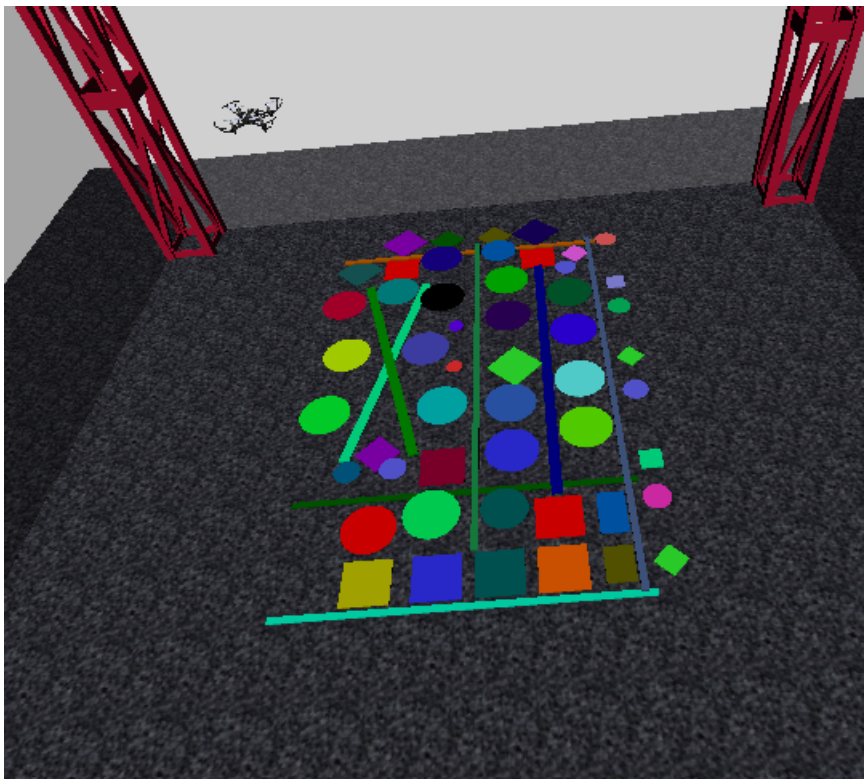
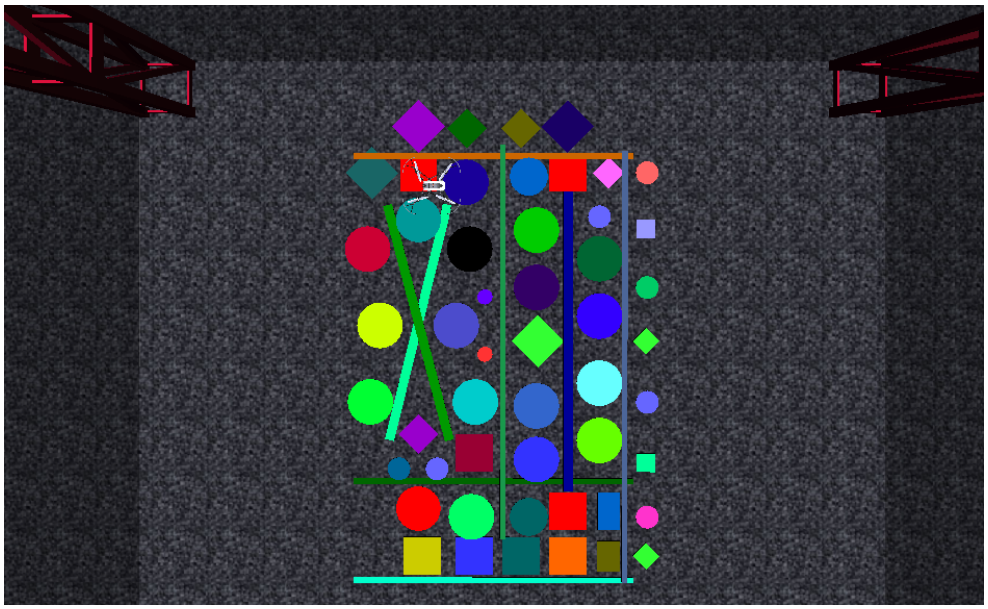


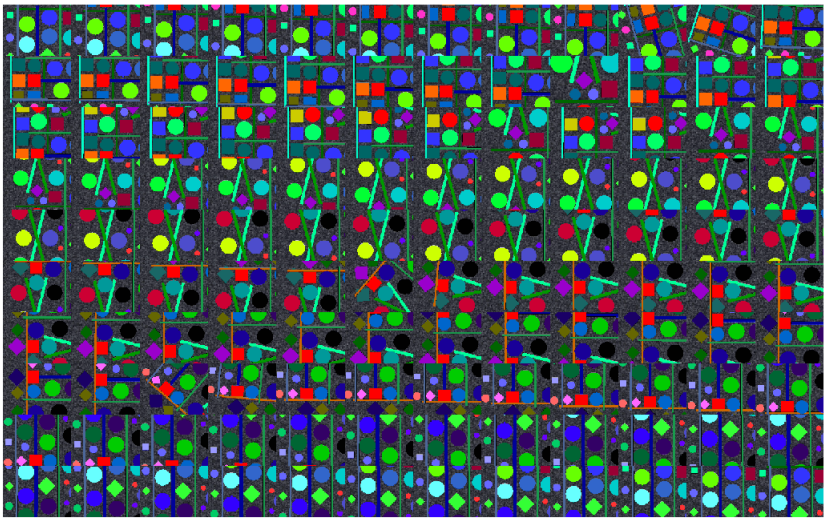
Figura 7.6: Vuelo del dron



7.2. Algoritmo de Stitching

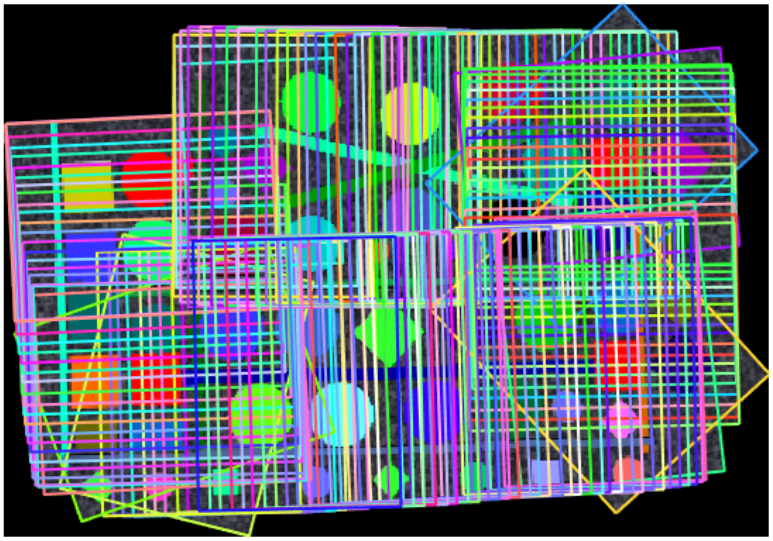
Una vez se exportan las imágenes, el algoritmo busca obtener el conjunto de imágenes en la carpeta especificada y una vez las adquiere, se desarrollan todos los pasos vistos anteriormente para llegar al resultado final, que corresponde al panorama, en las siguientes figuras, es posible observar en primer lugar las imágenes originales capturadas por el dron.

Figura 7.7: Imágenes originales



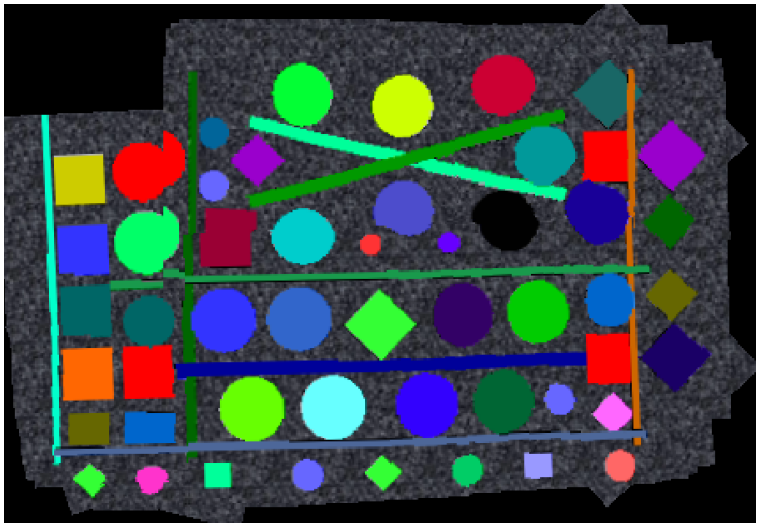
Acá se logra ver el panorama, que contiene una serie de cuadros de colores, estos representan la ubicación de cada una de las imágenes.

Figura 7.8: Ubicación de las imágenes



En seguida, se muestra el panorama final en el cual es posible apreciar que tiene similitud con el escenario que se muestra algunas figuras más arriba, si bien se presentan efectos de distorsión en algunos lugares de la imagen panorámica, es posible que sea a causa de que dron durante el trayecto no se desplaza de forma completamente paralela al suelo, recordemos que los rotores giran en sentidos contrarios y estos hace que no sea completamente estable durante todo el recorrido.

Figura 7.9: Panorama final



7.3. Validación del Algoritmo

Si bien se observa que el panorama final tiene cierto parecido con el escenario, se precisa analizar si en realidad las imágenes corresponden una con la otra. La fase de validación se encarga de realizar esta rectificación a partir de dos imágenes, donde una es una vista aérea del escenario y la otra es el panorama resultante del algoritmo, enseguida se muestran:

Figura 7.10: Escenario

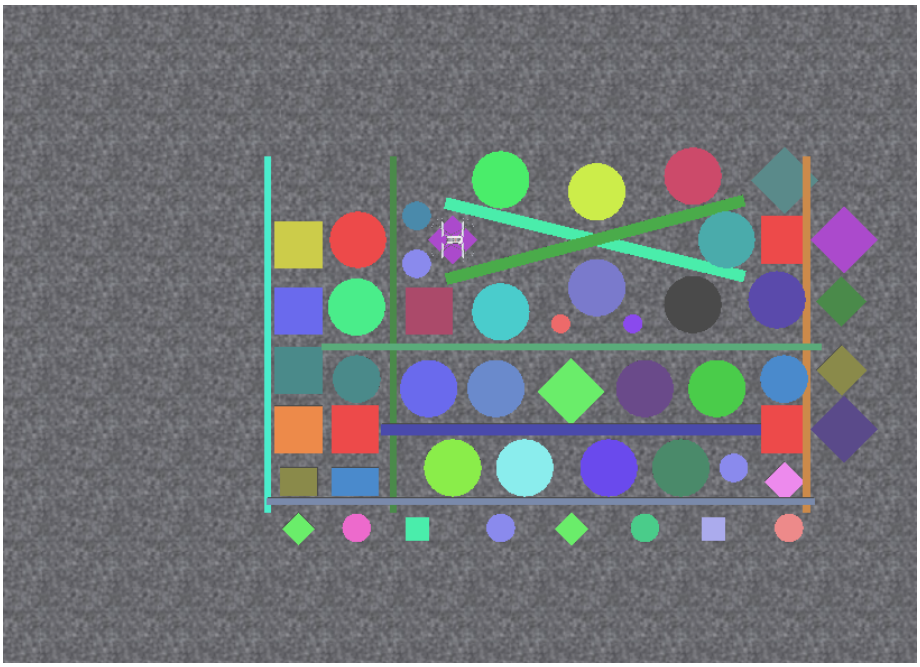
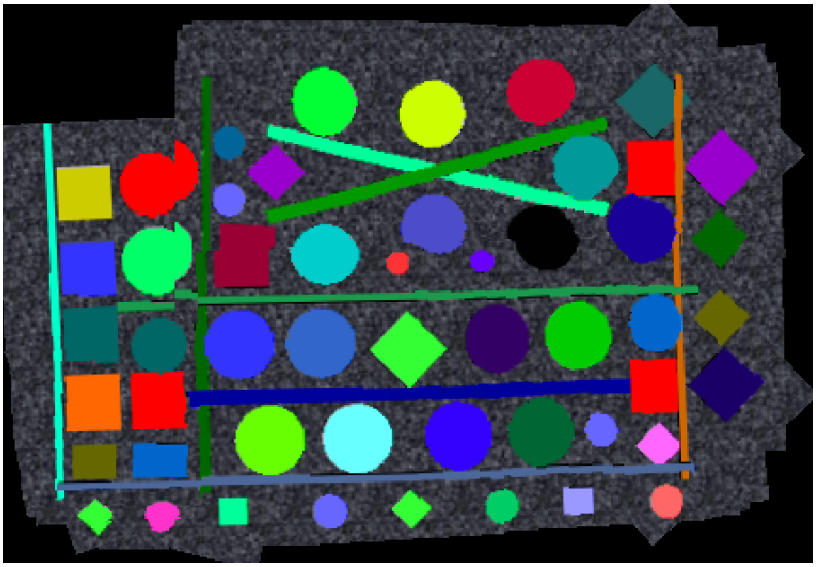
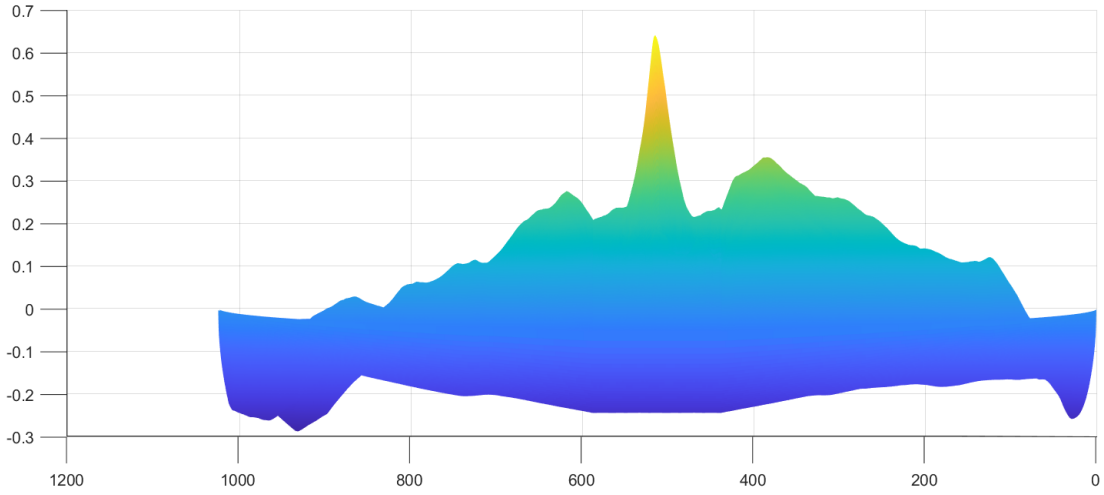


Figura 7.11: Panorama



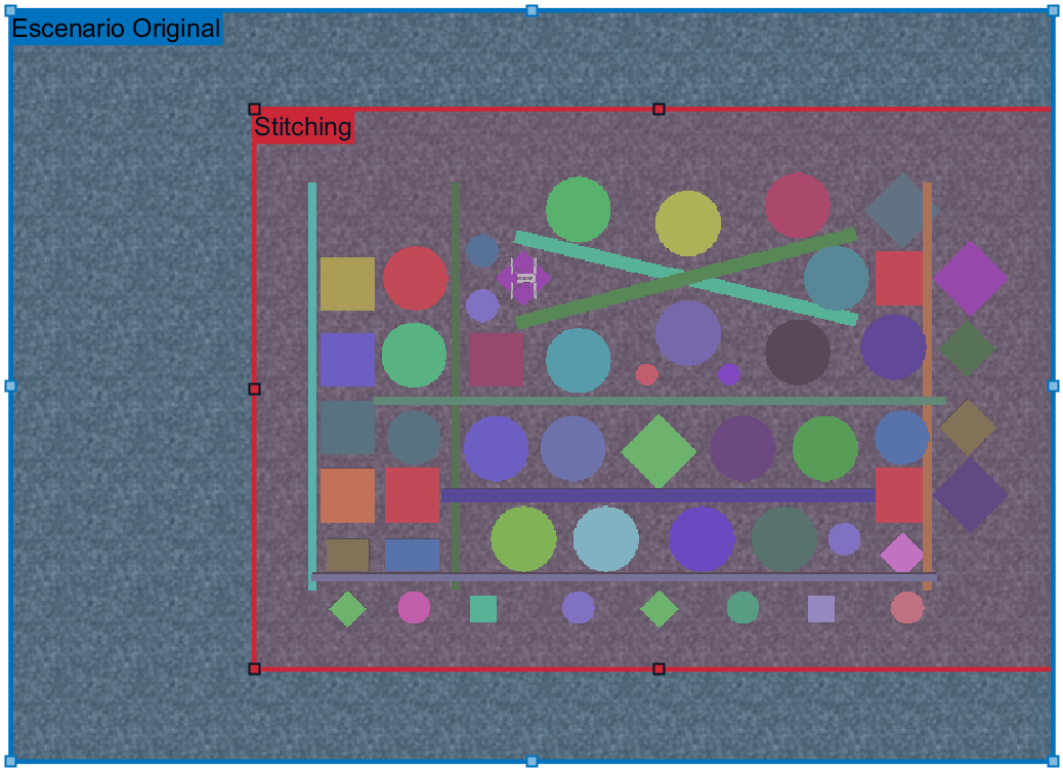
Una vez se tienen, es posible compararlas usando correlación cruzada, la función `surf` permite observar esta similitud de forma gráfica, lo que nos da como resultado las siguientes gráficas, donde se aprecia que la zona de color azul tiene poca o nula coincidencia entre ambas imágenes, por el contrario, la parte amarilla representa las zonas de mayor similitud.

Figura 7.12: Similitud



También se evalúa gráficamente si una imagen contiene a la otra, dicho en otras palabras, se determina si la imagen del panorama se encuentra dentro de la imagen del escenario, esto da como resultado la siguiente figura.

Figura 7.13: Comparacion



Adicionalmente, tomando como base el concepto visto previamente de la detección de características SURF, se introducen las mismas dos imágenes para realizar una comparación de características y así detectar puntos en común, lo que se obtiene como resultado las siguientes figuras, las cuales dejan ver algunas de las coincidencias en común, donde realizando un análisis cuantitativo de las dos imágenes nos encontramos con que cada una tiene aproximadamente 160 características propias y a la hora de compararlas, las características en común son 51, eso más o menos equivale a una coincidencia entre ellas del 32 %, sin embargo, hay que recordar que el algoritmo es lo suficiente robusto para detectar características con detalles minúsculos y hay secciones en las imágenes que detecta como si fueran la misma característica, como por ejemplo los recuadros rojos donde el dron realiza los giros, estos puntos de la imagen son lo suficientemente similares como para distinguir entre uno y otro a la hora de realizar la validación del algoritmo. Estas similitudes no son inconveniente en el momento de realizar la unión de imágenes, la que se evita este tipo de errores en la sección de ajuste de paquete.

Figura 7.14: Comparación características SURF

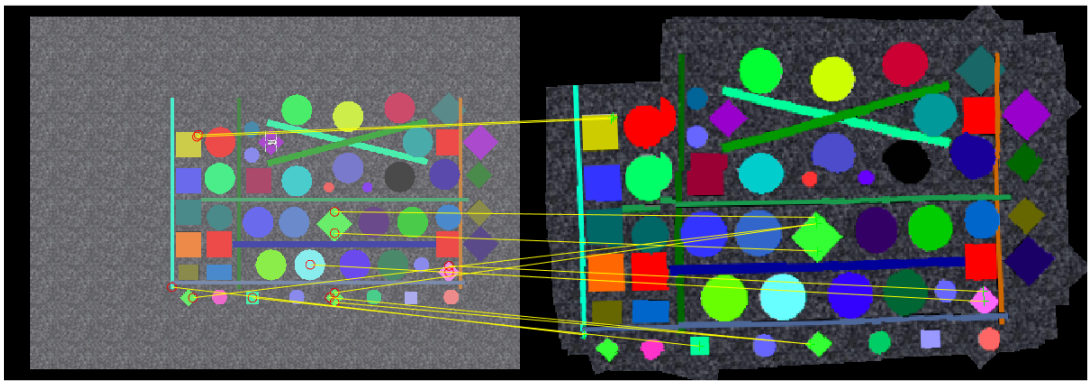


Figura 7.15: Comparación características SURF

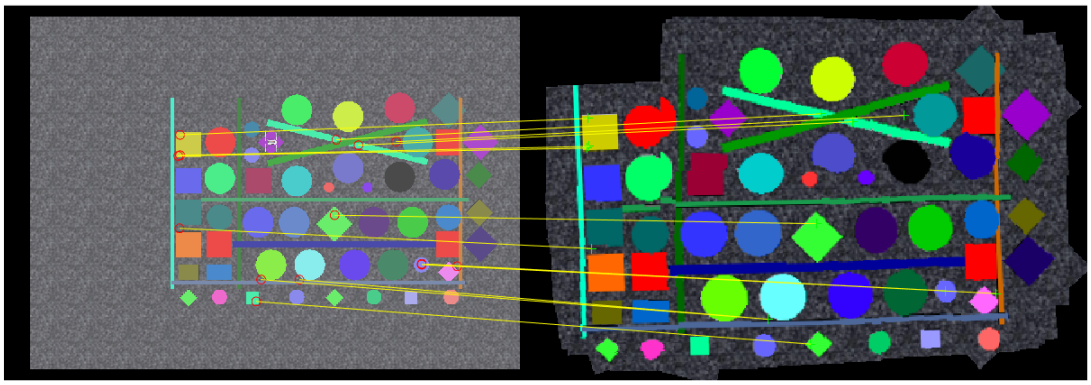
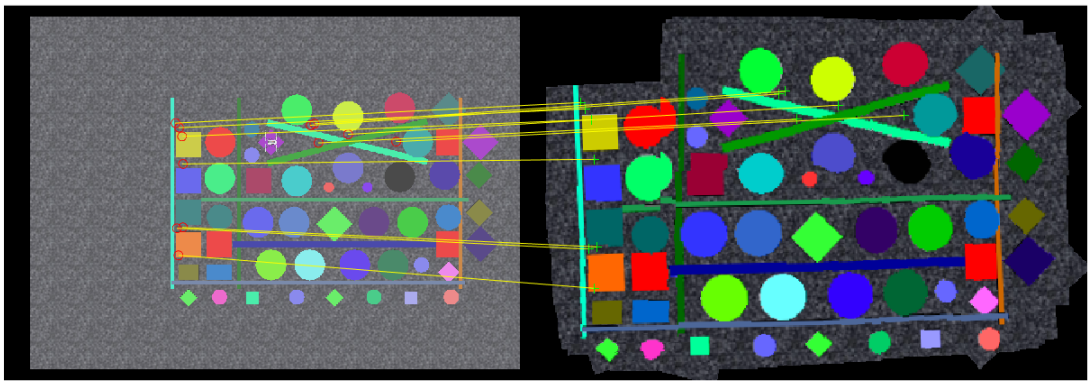


Figura 7.16: Comparación características SURF



Capítulo 8

Conclusiones

- Se diseñó e implementó un algoritmo de Stitching automático capaz de realizar la unión de cada una de las imágenes.
- La etapa de control de vuelo y diseño de la pista fue un proceso exitoso, ya que el dron logra seguir el plan de vuelo.
- El algoritmo de Stitching por su parte, demostró un funcionamiento adecuado, ya que localiza la correcta ubicación cada una de las 120, sin embargo, sufre de problemas de Ghosting en algunas zonas.
- La correlación cruzada, al igual que la detección y comparación de características SURF fueron herramientas que ayudan a realizar la validación del algoritmo de Stitching.
- En la coordinación conjunta entre las fases 1, 2 y 3 se evidenció un funcionamiento apropiado en el proceso de la unión de cada una de las fotografías capturadas.

Bibliografía

[1] J. Cortina, "Fotografía de arquitectura del movimiento moderno en valencia (1925-1965)." *Dialnet*, 2016. [Online]. Available: https://dialnet.unirioja.es/servlet/articulo?codigo=8410544

[2] M. Ayamga, S. Akaba, and A. A. Nyaaba, "Multifaceted applicability of drones: A review," *Technological Forecasting and Social Change*, vol. 167, p. 120677, 6 2021.

[3] InstitutoEducativoAeronáuticodeColombia. ¿licencia para volar drones en colombia? esto debes saber. [Online]. Available: https://iaerocol.co/blog/licencia-para-volar-drones-en-colombia-esto-debes-saber/

[4] P. Santamaria. (2018) Parrot mambo fpv, análisis. review con características, precio y especificaciones. [Online]. Available: https://www.xataka.com/analisis/parrot-mambo-fpv-analisis-el-dron-mas-divertido-para-volar-sin-miedo

[5] D. C. Tsouros, S. Bibi, and P. G. Sarigiannidis, "A review on uav-based applications for precision agriculture," *Information 2019, Vol. 10, Page 349*, vol. 10, p. 349, 11 2019. [Online]. Available: https://www.mdpi.com/2078-2489/10/11/349/htmhttps://www.mdpi.com/2078-2489/10/11/349

[6] MATLAB. Fly a parrot minidrone using the quadcopter simulink model - matlab y simulink - mathworks américa latina. [Online]. Available: https://la.mathworks.com/help/supportpkg/parrot/ug/fly-parrot-minidrone-quadcopter-simulink-model.html

[7] L. E. O. Collazos, C. F. Pineda, and E. W. Herrera, "Inseguridad rural y asociatividad: Una investigacion sobre violencia y formas organizativas en zonas de conflicto."

[8] C. Y. Chen and R. Klette, "Image stitching - comparisons and new techniques," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1689, pp. 615–622, 1999. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-48375-6_73

[9] A. Zomet, A. Levin, S. Peleg, and Y. Weiss, "Seamless image stitching by minimizing false edges," *IEEE Transactions on Image Processing*, vol. 15, pp. 969–977, 4 2006.

[10] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *International Journal of Computer Vision 2006 74:1*, vol. 74, pp. 59–73, 12 2006. [Online]. Available: https://link.springer.com/article/10.1007/s11263-006-0002-3

[11] J. Quintero. (2018) Implementación de un drone de apoyo al centro de vigilancia de unicentro bogotá. [Online]. Available: http://repository.unipiloto.edu.co/handle/20.500.12277/9971

[12] areadron.com. ¿quÉ tipos de drones existen? [Online]. Available: https://www.areadron.com/que-tipos-de-drones-existen/

[13] H. T. Fly. Roll, pitch, and yaw. [Online]. Available: https://howthingsfly.si.edu/flight-dynamics/roll-pitch-and-yaw

[14] A. Civita, S. Fiori, and G. Romani, "A mobile acquisition system and a method for hips sway fluency assessment," *Information (Switzerland)*, vol. 9, 12 2018.

[15] MathWorks. Tipos de imagen - matlab and simulink - mathworks américa latina. [Online]. Available: https://la.mathworks.com/help/matlab/creating_plots/image-types.html

[16] HISOUR. Sistema de color yuv – hisour arte cultura historia. [Online]. Available: https://www.hisour.com/es/yuv-color-system-25916/

[17] DEXON. What are rgb and yuv color spaces? | news. [Online]. Available: https://dexonsystems.com/news/rgb-yuv-color-spaces

[18] G. Ward, "Hiding seams in high dynamic range panoramas," *Proceedings - APGV 2006: Symposium on Applied Perception in Graphics and Visualization*, p. 150, 2006.

[19] H. S. Sawhney, "True multi-image alignment and its application to mosaicing and lens distortion correction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 235–243, 3 1999.

[20] H. Y. Shum and R. Szeliski, "Systems and experiment paper: Construction of panoramic image mosaics with global and local alignment," *International Journal of Computer Vision 2000 36:2*, vol. 36, pp. 101–130, 2000. [Online]. Available: https://link.springer.com/article/10.1023/A:1008195814169

[21] P. F. McLauchlan and A. Jaenicke, "Image mosaicing using sequential bundle adjustment," *Image and Vision Computing*, vol. 20, pp. 751–759, 8 2002.

[22] P. Islamabad, "A comparative analysis of sift, surf, kaze, akaze, orb, and brisk," 2018.

[23] D. Bojanic, K. Bartol, T. Pribanic, T. Petkovic, Y. D. Donoso, and J. S. Mas, "On the comparison of classic and deep keypoint detector and descriptor methods," *International Symposium on Image and Signal Processing and Analysis, ISPA*, vol. 2019-September, pp. 64–69, 9 2019.

[24] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision 2004 60:2*, vol. 60, pp. 91–110, 11 2004. [Online]. Available: https://link.springer.com/article/10.1023/B:VISI.0000029664.99615.94

[25] R. Arandjelovic and A. Zisserman, "Three things everyone should know to improve object retrieval," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2911–2918, 2012.

[26] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3951 LNCS, pp. 430–443, 2006. [Online]. Available: https://link.springer.com/chapter/10.1007/11744023_34

[27] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence*

and Lecture Notes in Bioinformatics), vol. 3951 LNCS, pp. 404–417, 2006. [Online]. Available: https://link.springer.com/chapter/10.1007/11744023_32

[28] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2548–2555, 2011.

[29] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2564–2571, 2011.

[30] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6314 LNCS, pp. 778–792, 2010. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-15561-1_56

[31] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, "Kaze features," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7577 LNCS, pp. 214–227, 2012. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-33783-3_16

[32] E. Valveny, J. Gonzàlez, and R. Baldrich. Clasificación de imágenes: ¿cómo reconocer el contenido de una imagen? - universitat autònoma de barcelona - información del curso | coursera. [Online]. Available: https://www.coursera.org/learn/clasificacion-imagenes/home/info

[33] J. Pascau, J. D. Gispert, and R. Martinez, "Registro de imágenes en medicina nuclear," *Revista de la Real Academia de Ciencias Exactas, Físicas y Naturales*, 2002. [Online]. Available: https://www.researchgate.net/publication/233413374_Registro_de_imagenes_en_medicina_nuclear

[34] M. A. Fischler and R. C. Bolles, "Random sample consensus," *Communications of the ACM*, vol. 24, pp. 381–395, 6 1981. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/358669.358692

[35] R. Ebker, "Random sample consensus explained," *Baeldung on Computer Science*, 2022. [Online]. Available: https://www.baeldung.com/cs/ransac

[36] M. Brown and D. G. Lowe, "Recognising panoramas," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2, pp. 1218–1225, 2003.

[37] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment – a modern synthesis," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1883, pp. 298–372, 2000. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-44480-7_21

[38] J. C. Yoo and T. H. Han, "Fast normalized cross-correlation," *Circuits, Systems and Signal Processing 2009 28:6*, vol. 28, pp. 819–843, 8 2009. [Online]. Available: https://link.springer.com/article/10.1007/s00034-009-9130-7

[39] J. P. Lewis, "Fast normalized cross-correlation."

[40] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, 1st ed. USA: Addison-Wesley Longman Publishing Co., Inc., 1992.

[41] MathWorks. Parallel computing toolbox - matlab. [Online]. Available: https://la.mathworks.com/products/parallel-computing.html

[42] ——, *Parallel Computing Toolbox™ User's Guide*, 2004. [Online]. Available: https://la.mathworks.com/help/pdf_doc/parallel-computing/parallel-computing.pdf

[43] ——. 3d world editor. [Online]. Available: https://lost-contact.mit.edu/afs/inf.ed.ac.uk/group/teaching/matlab-help/R2016b/sl3d/3dworldeditor-app.html

[44] MATLAB. Detect surf features and return surfpoints object - matlab detectsurffeatures - mathworks américa latina. [Online]. Available: https://la.mathworks.com/help/vision/ref/detectsurffeatures.html#namevaluepairarguments