



industriales
etsii

**Escuela Técnica
Superior
de Ingeniería
Industrial**

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Diseño de un sistema de identificación y conteo de árboles para el sector agrícola.

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

Autor: JESÚS PUJANTE SÁEZ
Director: FRANCISCO JAVIER GARRIGÓS GUERRERO
Codirector: JOSÉ JAVIER MARTÍNEZ ÁLVAREZ



**Universidad
Politécnica
de Cartagena**

Cartagena, 14/10/2018

- 1. Introducción**
- 2. Uso de drones**
- 3. Legislación**
- 4. Objetivos**
- 5. Fotogrametría**
 - 5.1 Escala
 - 5.2 Distancia focal
- 6. Búsqueda de un software analizador de imágenes**
- 7. Planificadores de vuelo**
 - 7.1 DJI Ground Station Pro
 - 7.2 Pix 4D Capture
 - 7.3 UGCS
 - 7.4 Mission Planner
 - 7.5 Litchi
 - 7.6 DroneDeploy
- 8. Planificador de vuelo escogido**
 - 8.1 Instalación
 - 8.2 Funciones utilizadas
- 9. Finca estudiada**
- 10. Primera prueba**
 - 10.1 Stitching
- 11. Segunda prueba**
 - 11.1 Tipos de stitching
- 12. Proceso de conteo**
 - 12.1 Cambio a 8 bits
 - 12.2 Threshold
 - 12.3 Watershed
 - 12.4 Escalar imagen
 - 12.5 Analyze particles
- 13. Diferencia de luz solar en la toma de imágenes**
- 14. Script**
 - 14.1 Mejora
- 15. Comparativa imágenes drone/google maps**
- 16. Presupuesto**
- 17. Posibles mejoras**
- 18. Futuros desarrollos**
- 19. Conclusiones**
- 20. Bibliografía**
- 21. Fuente de las figuras**
- 22. Anexos**

1. Introducción

Este trabajo nace del interés que están mostrados algunas empresas hortofrutícolas de la Región de Murcia en la utilidad que pueden tener los drones aplicados al campo de la agricultura. Esto combinado con sistemas de gestión de datos, sensores...etc, es lo que se define como agricultura de precisión.

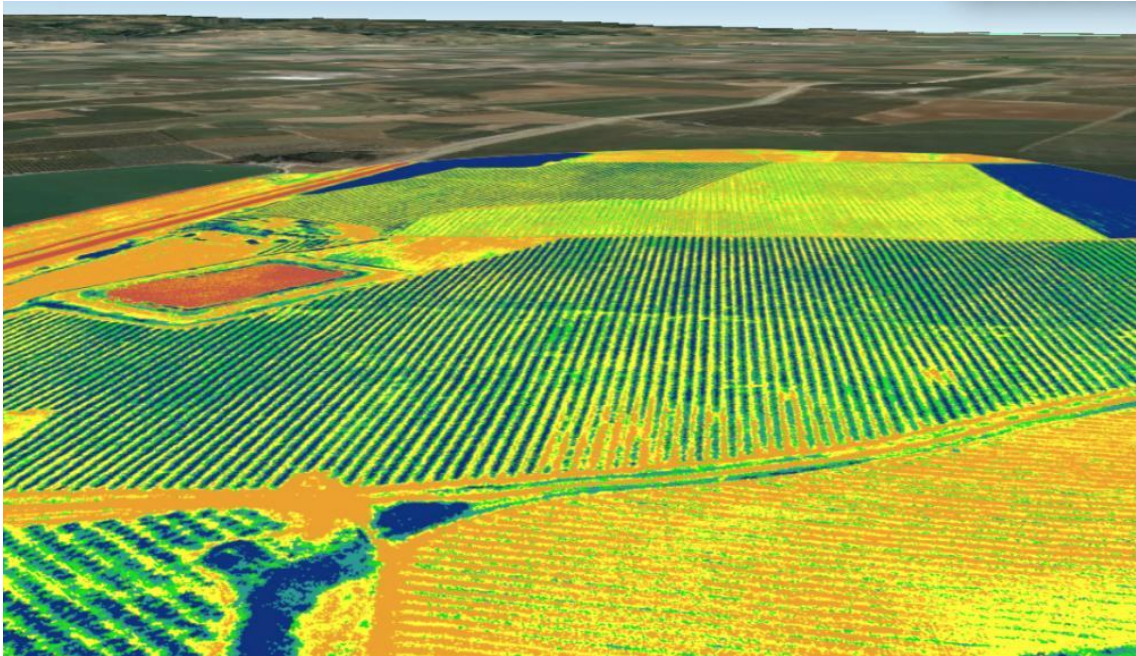


Figura 1: Fotografía aérea de un drone equipado con cámara térmica

Estos procesos se realizan mediante sistemas de navegación por satélite, de información geográfica y sensores situados en la parcela. Los sistemas recogen información que después se utiliza para tomar decisiones con mayor precisión, y también para optimizar el rendimiento de los cultivos.

Los pasos a seguir para poder implementar el control y el seguimiento de los cultivos son los siguientes:

1. Adquisición de datos:

La variedad y cantidad de datos es tan amplia como nuestra capacidad tecnológica y como el número y variedad de sensores que tengamos. Alguno de los sensores más utilizados son: Humedad en Suelo, Caudalímetro, Conductividad Eléctrica (CE), geolocalización, Medidor pH, tipo de suelo, probabilidad de plagas y enfermedades...etc.

2. Análisis de datos:

Una vez son registrados los datos estos deben ser tratados para facilitar su interpretación y entendimiento, para ello se usan todas las herramientas estadísticas y matemáticas que tenemos a nuestro alcance como pueden ser: AgGis, clasificación de datos, gráficas, mapeado...etc.

3. Toma de decisiones. Preventivas y de gestión:

Al tener toda la información sobre nuestro cultivo o plantación es el momento de actuación del ingeniero agrónomo o del técnico agrícola, la suma de los datos más los conocimientos agronómicos mas su experiencia crean las condiciones perfectas para que se puedan tomar las mejores decisiones sobre el cultivo.

Estas acciones pueden ser tanto preventivas (ej. adelantarse a plagas y enfermedades) o de gestión (ej. conocer cuánto y cuando hay que regar con el consiguiente ahorro) gracias a las previsiones y la mediciones.

4. Monitorización del rendimiento:

Al finalizar la campaña es el momento de evaluar la producción, las decisiones tomadas y el rendimiento final, con el objetivo de aprender de los errores y conocer los aciertos. Esto es posible gracias a la continua monitorización del cultivo, además la capacidad de almacenar la información permite la comparación entre campañas.

2. Uso de drones

Actualmente y cada vez más se pueden ver drones sobrevolando los campos para diferentes usos, entre los cuales destacan los siguientes:

- Fumigación de precisión: Es uno de los problemas más importante a los que se enfrentan los agricultores, el uso de plaguicidas y pesticidas. Tanto para el riesgo que pueden conllevar para la salud humana, como por el malgasto que se produce aplicándolo de forma tradicional. Ya en los años 80, en Japón, Yamaha diseñó una aeronave de control remoto para hacer más eficiente la fumigación. Tras años de desarrollo hoy en día más de 7000 agricultores nipones confían en esta tecnología para reducir sus costes y ser más precisos a la hora de atacar las plagas.



Figura 2: Drone realizando labores de fumigación

- **Detección de plagas y malas hierbas:** El control en tiempo real de las plantaciones ha sido, hasta ahora, algo complejo, que requería de personal de campo que estuviese siempre pendiente de las plantaciones. Y aún así, en ocasiones, la detección de plagas llegaba demasiado tarde. España es pionera en este campo, ya que muchas bodegas trabajan ya con drones para la detección temprana de grama, una mala hierba perenne de difícil control. Gracias a las aeronaves no tripuladas, los viticultores pueden detectar las zonas problemáticas de forma temprana para centrar sus esfuerzos para evitar que las malas hierbas lleguen a los cultivos.
- **Peritaje e inventario de terreno de cultivo:** Mientras el minifundio es habitual en zonas del norte de España, los latifundios son más comunes en zonas del sur de España, como es Andalucía donde es conocida por sus grandes extensiones de olivares.

Contabilizar las plantaciones, para el control de las subvenciones agrícolas o realizar peritajes, es algo complicado y costoso en ante extensiones de tan grandes de terreno.

En estas regiones ya se están implementando sistemas de drones destinados al control de grandes plantaciones, el inventario de cultivos, la gestión de ayudas públicas y la eliminación del fraude.

3. Legislación

Desde el 30 de Diciembre de 2017 la ley que aplica al uso civil de aeronaves pilotadas por control remoto (RPAs) en España es el Real Decreto Real Decreto 1036/2017.

La normativa actual de drones solamente obliga a sacarse la licencia de piloto a las personas que vayan a desempeñar una actividad profesional con drones. Estas personas tendrán que acreditar unos conocimientos teóricos y prácticos, además de acreditar un certificado médico de clase LAPL (para drones de < 25KG) o de Clase II (para drones >25KG).

Los usuarios de aeronaves pilotadas por control remoto destinadas exclusivamente a vuelos recreativos deberán de cumplir las siguientes condiciones:

- Volar a una distancia mínima de 8 km de cualquier aeropuerto o aérodromo.
- Volar fuera del espacio aéreo controlado.
- No sobrepasar los 120 metros de altura sobre el suelo, o sobre el obstáculo más alto situado dentro de una radio de 150 metros desde la aeronave.
- Volar de día y en buenas condiciones meteorológicas. Aquí hay que destacar que si la aeronave pesa menos de 2 kilogramos están permitidos los vuelos nocturnos siempre que no se superen los 50 metros de altura.
- Los vuelos siempre será dentro del alcance visual del piloto (VLOS).
- Las aeronaves de menos de 250 gramos podrán volar en ciudad y sobre aglomeraciones de personas y edificios siempre y cuando no se superen los 20 metros de altura.
- Aunque no es obligatorio para el uso recreativo, sí es muy recomendable contar con un seguro de responsabilidad civil.

El espacio aéreo está regulado por la AESA, por lo tanto, esta nueva legislación sigue manteniendo la prohibición de sobrevolar núcleos urbanos o espacios con una alta masificación de gente sin el consentimiento especial por parte de la Agencia Española de Seguridad Aérea.



Figura 3: Agencia Estatal de Seguridad Aérea perteneciente al ministerio de fomento

4. Objetivos

Hoy en día en la Región de Murcia existen algunas empresas que se dedican a la agricultura de precisión, pero en particular a temas más relacionados con detección temprana de plagas y termografía.

Los servicios que ofrecen estas empresas requieren de una inversión inicial muy elevada por parte de los agricultores, ya que es necesario el uso de cámaras termográficas, sensores y drones con una mayor autonomía.

Por lo tanto uno de los objetivos de este trabajo es poder demostrar que es posible automatizar un proceso que hasta ahora se viene realizando de forma manual, utilizando únicamente un drone y software de licencia libre.

El principal objetivo de este trabajo es poder realizar el conteo de árboles en una plantación agrícola mediante técnicas de análisis de imágenes suministradas a través de un Drone DJI Phantom 4 Pro.

Las fases del trabajo de una forma más detallada son las expuestas a continuación:

- Realizar un análisis comparativo y/o desarrollo de algoritmos de stitching adecuados para esta aplicación específica.
- Estudio de estrategias de captura de imágenes y desarrollo de rutinas de control de vuelo de forma automatizada más adecuadas para este caso de uso.
- Desarrollo de un algoritmo de procesamiento de imágenes para la extracción de características de interés: número de árboles, volumen de la copa, producción estimada, etc., así como métricas estadísticas relevantes.
- Comparación de los resultados obtenidos con otros sistemas o métodos existentes

5. Fotogrametría

Antes de empezar con la toma de imágenes con el Drone se plantea la siguiente cuestión: Cuando se toman las imágenes, ¿cómo se interpretan estas imágenes en cuanto a la distancia real que tiene cada porción de terreno?

Plantear esta cuestión fue un punto de partida para descubrir e investigar sobre el tema de las escalas para imágenes aéreas y mapas topográficos.

Para comenzar se explicará en qué consiste el concepto de las escalas y posteriormente el concepto de distancia focal y para qué sirven en el análisis de imágenes.

5.1 Escala

La escala es la relación matemática que existe entre las dimensiones reales y las de la fotografía.

Las escalas se escriben en forma de razón dónde el antecedente indica el valor en la fotografía y el consecuente el valor de la realidad.

Por ejemplo, la escala 1:500 significa que 1cm del plano equivale a 5m en la realidad.

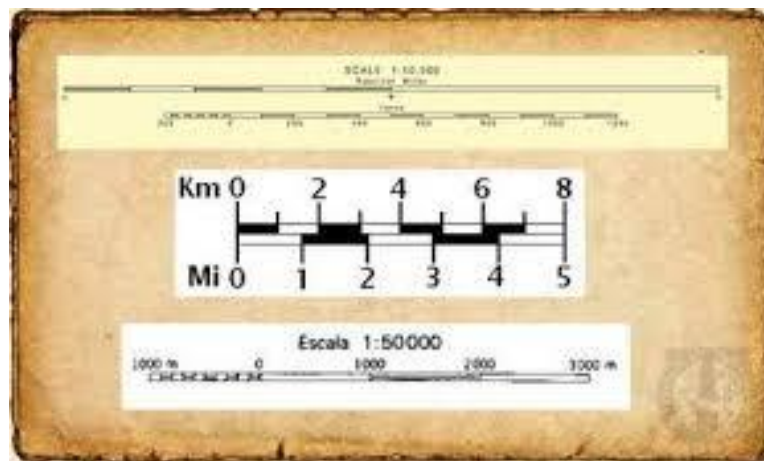


Figura 4: Escalas topográficas

5.2 Distancia focal

La distancia focal o longitud focal de una lente es la distancia entre el centro óptico de la lente y el foco. La inversa de la distancia focal de una lente es la potencia y se mide en dioptrías.

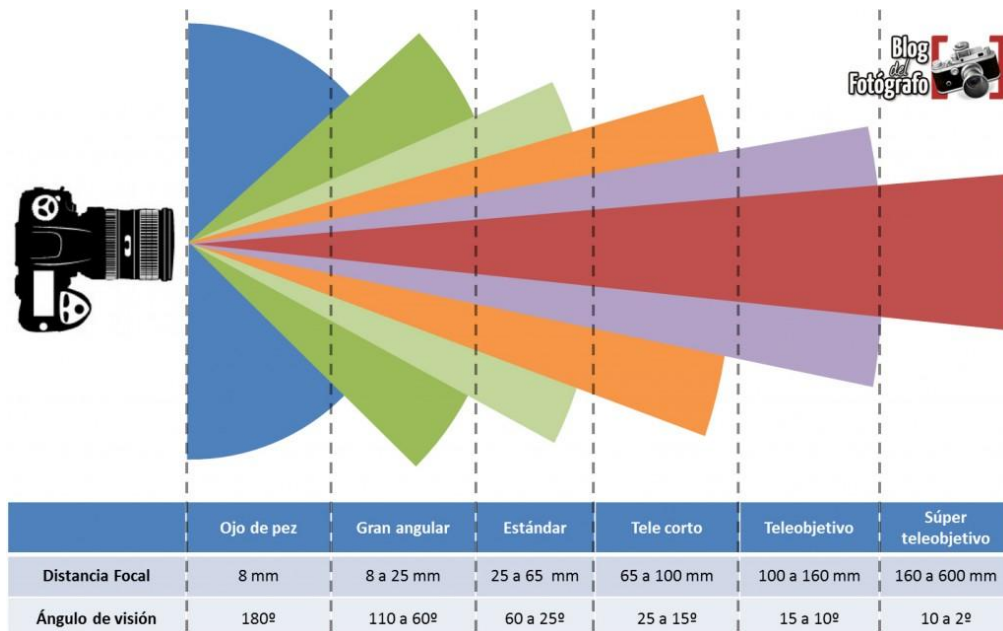


Figura 5: Tipos de objetivos según la distancia focal y el ángulo de visión

Según se ha podido comprobar en las características de la lente de la cámara que lleva incorporada el drone, tenemos una distancia focal equivalente a 24 mm, por lo tanto se dispone de un objetivo gran angular, que, aunque existe el ojo de pez que es capaz

de capturar imágenes con mayor ángulo de visión, un objetivo gran angular no está nada mal para realizar el estudio.

Esta imagen resume muy bien gráficamente el concepto de la distancia focal.

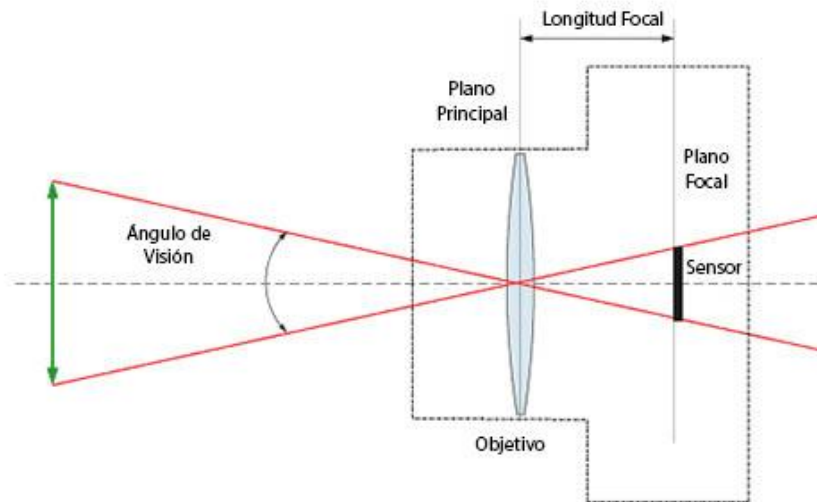


Figura 6: Concepto de distancia focal en objetivos

Por lo tanto, una vez explicado este concepto, lo realmente importante está aquí, ya que existe una relación entre la distancia focal y la altura a la que se realiza la imagen mediante la que se obtiene la escala, por lo que según la altura a la que se realice la imagen se podrá conocer en todo momento las dimensiones reales del terreno.

Esta relación es la siguiente:

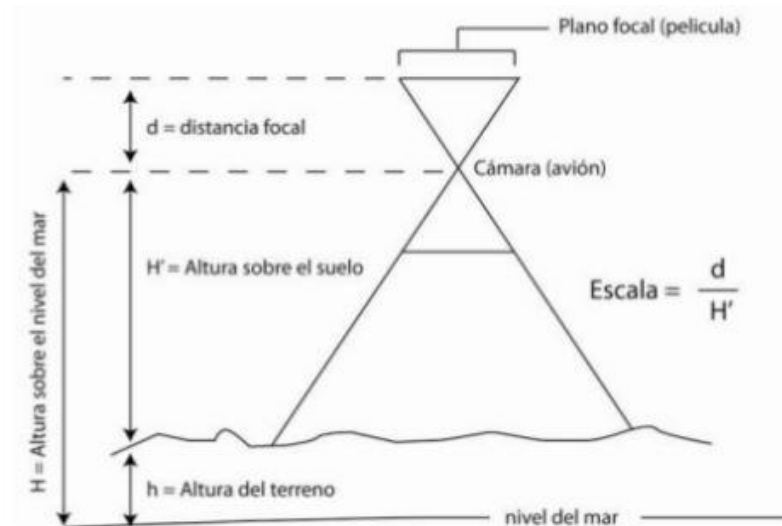


Figura 7: Distancia focal en cámara aérea

6. Búsqueda de un software analizador de imágenes

La primera fase del trabajo es de las más importantes, ya que hay que decidir que software es el más idóneo para realizar el estudio y análisis de imágenes. Para ello se va a realizar una comparativa de los diferentes softwares disponibles en el mercado, y se van a clasificar según el precio, sector en el que se aplica, tipo de plataforma GNU, lenguaje de programación...etc, es decir, se va a realizar un análisis-comparativo, gracias al cuál se podrá determinar la herramienta que más conviene para llevar a cabo el proyecto.

HERRAMIENTA	PRECIO	PROGRAMACIÓN	VENTAJAS	INCONVENIENTES
Cellprofilter	GNU	Software cerrado	Ninguna	Exclusivo para contar células
eCognition developer	2.000 €	Software cerrado	Software potente y con multiples usos de conteo	Precio
Aforge.net	GNU	C#	Plataforma GNU con total libertad para desarrollar scripts y librerías	No es tan fácil hacer scripts como con Image J, más desarrollo de código
ImageJ	GNU	Scripts	Plataforma GNU con total libertad para desarrollar scripts y librerías, enfocada solamente a imágenes	No es tan potente como algunos softwares privados que ofrecen algunas empresas del sector
Matlab	Licencia gratuita (Universidad)	Matlab	Es posible desarrollar programas potentes de procesamiento de imágenes	Mayor tiempo de programación
OpenCV	GNU	C++,Phyton y Java	Plataforma GNU con total libertad para desarrollar scripts y librerías	No es tan fácil hacer scripts como con Image J, más desarrollo de código

Al realizar la búsqueda se han encontrado muchas herramientas de videovigilancia para contar personas, pero estas herramientas no sirven, ya que no son gratuitas, están muy limitadas y su uso es exclusivo para las cámaras que ellos comercializan, por lo tanto quedan descartados estos softwares.

También se han encontrado muchos softwares comerciales que por su precio tan alto, el uso de estos es inviable para el proyecto, ya que lo que se pretende es desarrollar

una herramienta utilizando plataformas GNU que permitan realizar los conteos con una precisión aceptable.

Al igual que se han encontrado plataformas GNU y de pago, también a través de algunos artículos recientemente publicados en internet, centros de investigación y universidades de algunos países como Estados Unidos, están desarrollando algoritmos muy potentes que pueden realizar esta tarea, aunque en muchos de los estudios las imágenes que utilizan para el análisis son imágenes de alta calidad que proporcionan aviones con cámaras que llevan incorporada la tecnología LIDAR.

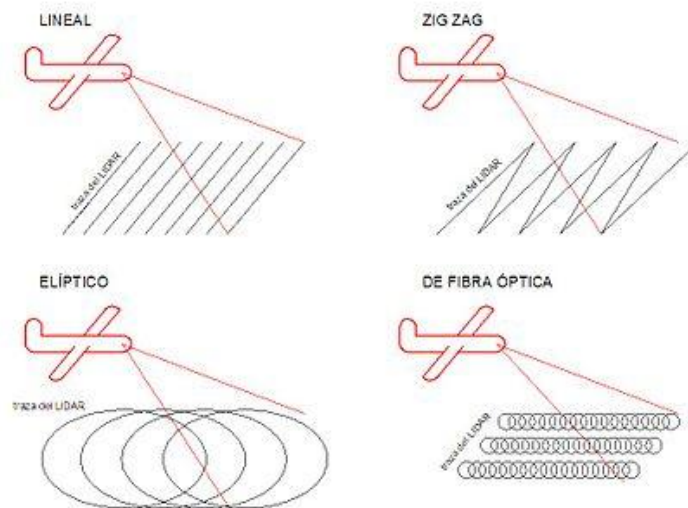


Figura 8: Tipos de barridos en tecnología LIDAR

Esta tecnología permite realizar una nube de puntos sobre el terreno utilizando un haz de laser pulsado y hacer un mapeado del área estudiada muy preciso, pero dados los recursos limitados de los que se disponen sería inviable la utilización de esta tecnología.

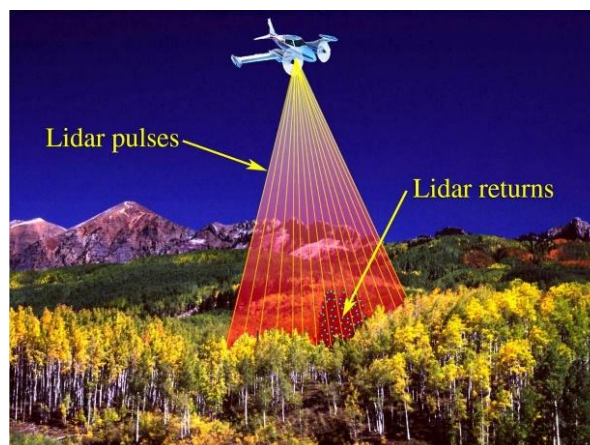


Figura 9: LIDAR

Por lo tanto una vez realizado el estudio-comparativo sobre las herramientas de las que se disponen para la realización del trabajo, se llega a la conclusión de que la herramienta más idónea para realizar el estudio será ImageJ, ya que es una plataforma

GNU que permite desarrollar librerías y scripts enfocados únicamente al procesamiento y análisis de imágenes, por lo que es la herramienta que mejor se ajusta a las necesidades reales del proyecto.

7. Planificadores de vuelo

Después de definir y explicar los conceptos básicos sobre la técnica en la obtención de imágenes y el fundamento científico, es imprescindible la utilización de un planificador de vuelo que permita definir diferentes rutas de vuelo para que el drone vaya recorriendo la malla seleccionada y vaya tomando fotografías que serán las que posteriormente se utilizarán para realizar un stitching y obtener la imagen completa de la finca estudiada.

Primeramente se hará un estudio de los diferentes planificadores de vuelo que existen en el mercado, sus ventajas y desventajas.

7.1 DJI Ground Station Pro

Recientemente la firma DJI ha sacado su propia aplicación orientada a la planificación y el vuelo autónomo de las aeronaves de la firma. Es compatible con los modelos más actuales de la marca (phantom, inspire, mavic...).

Es una aplicación muy recomendable para drones de la marca ya que todo el desarrollo está pensado íntegramente para la marca. El único inconveniente es que de momento solo está disponible para iPad.

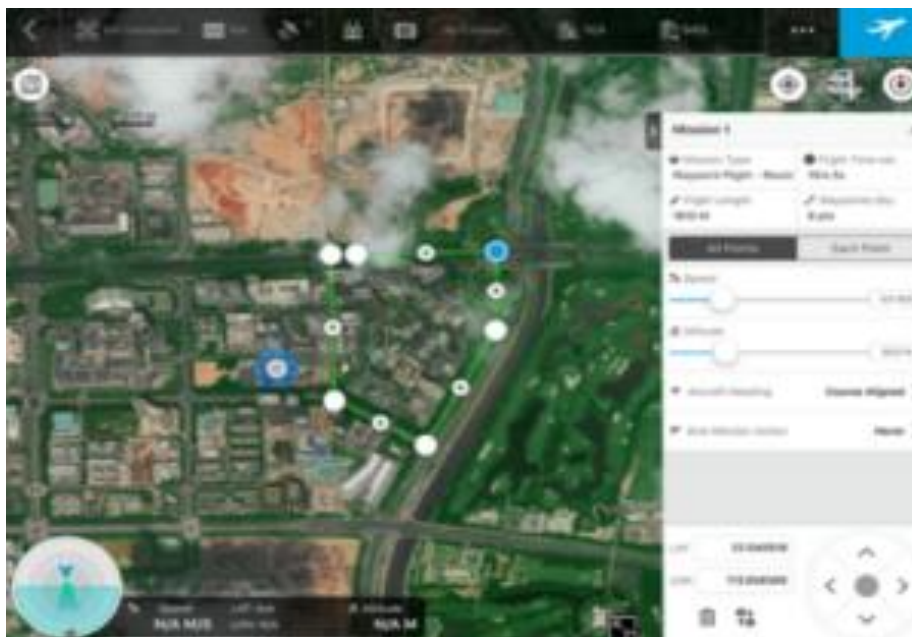


Figura 10: Interfaz DJI Ground Station Pro

7.2 Pix 4D Capture

Es una de las aplicaciones más conocidas para la planificación de vuelo. Se trata de una aplicación móvil disponible tanto para Android como para iOS. Es compatible con la mayoría de drones profesionales más comercializados (phantom, inspire, mavic, bebop 2, 3dr solo...).



Figura 11: Interfaz de Pix 4D Capture

El funcionamiento de esta aplicación es realmente sencillo. Una vez que se tenga clara la zona que queremos mapear, se abre la aplicación y se selecciona el área a sobrevolar. Como en otras apps, es muy sencillo configurar aspectos relativos a al vuelo: velocidad, ángulo de inclinación de la cámara o el porcentaje de solape entre imágenes. Una vez definido y calculadas las rutas, la aplicación se comunicará con el dron y éste empezará el vuelo de forma autónoma.

Esta app permite definir vuelos en forma de cuadrícula, doble cuadrícula (solapando dos cuadrículas perpendiculares), vuelos circulares, incluso unir varios de los anteriores dentro de un único vuelo.

7.3 UGCS

La principal ventaja de esta aplicación es que es compatible con multitud de drones y de controladoras. Es una aplicación de escritorio compatible con Windows, MacOS y Linux. A diferencia de las aplicaciones móviles, con este software se tendrá que planificar las misiones desde el equipo. No es una aplicación tan sencilla, pero el potencial de esta herramienta es ilimitado. La cantidad de configuraciones y herramientas que presenta es enorme.

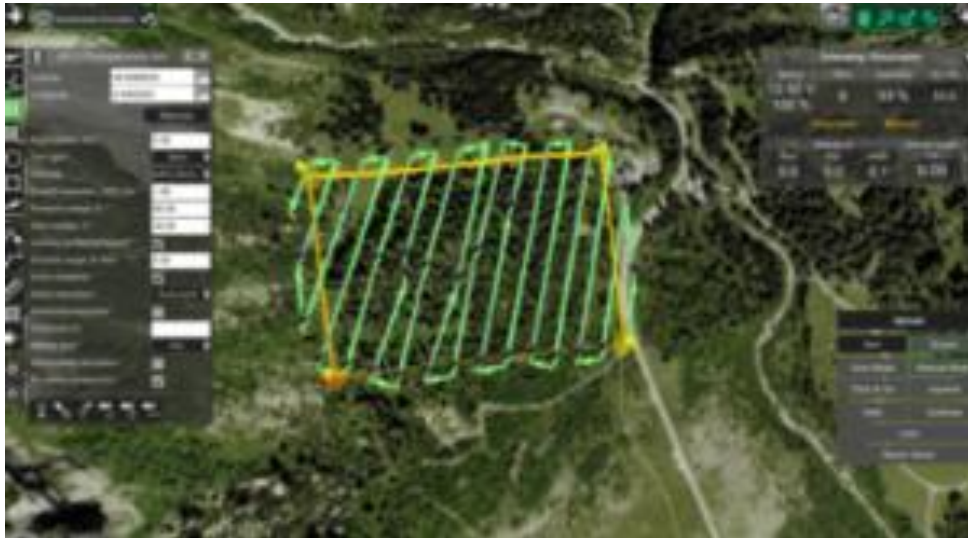


Figura 12: Interfaz de UGCS

Soporta misiones con varios vuelos y múltiples cambios de batería, telemetría, control del dron mediante teclado y joystick, control de múltiples drones simultáneamente, dispositivos de fabricantes menos conocidos, etc. Además, es el único software que permite importar perfiles de terreno de forma que sean tenidos en cuenta en la planificación de la misión. Hay varias versiones, desde gratuita hasta empresariales de pago.

7.4 Mission Planner

Es una aplicación muy completa, muy similar a la anteriormente comentada UGCS, con la que se pueden configurar infinidad de parámetros de nuestro dron y de la controladora. Está disponible para Windows pero también existe una versión multiplataforma llamada APM Planner. Con este software será necesario planificar el vuelo desde el PC y posteriormente pasar la misión al dron.



Figura 13: Interfaz de Mission Planner

Esta aplicación es específica del controlador de vuelo ardupilot, aunque hay varias empresas que dan soporte para otros dispositivos. Un elemento interesante es que permite ajustar y cambiar los parámetros de la controladora para diferentes configuraciones físicas de drones (alas fijas, cuatrimotores, hexarotores, octocopteros... y diferentes formas de estructura).

La principal razón para utilizar mission planner es poder cambiar los parámetros del controlador de vuelo.

7.5 Litchi

Esta aplicación está disponible para Android y iOS. Es compatible con la gran mayoría de drones de la marca DJI (phantom, inspire, mavic...). Es una aplicación de pago, tanto en la versión Android como en la versión iOS.



Figura 14: Interfaz de Litchi

7.6 DroneDePloy

Esta aplicación se caracteriza por ser muy intuitiva y sencilla de utilizar. La aplicación está disponible tanto para dispositivos Android como iOS y es compatible con la mayoría de los drones de la firma DJI.

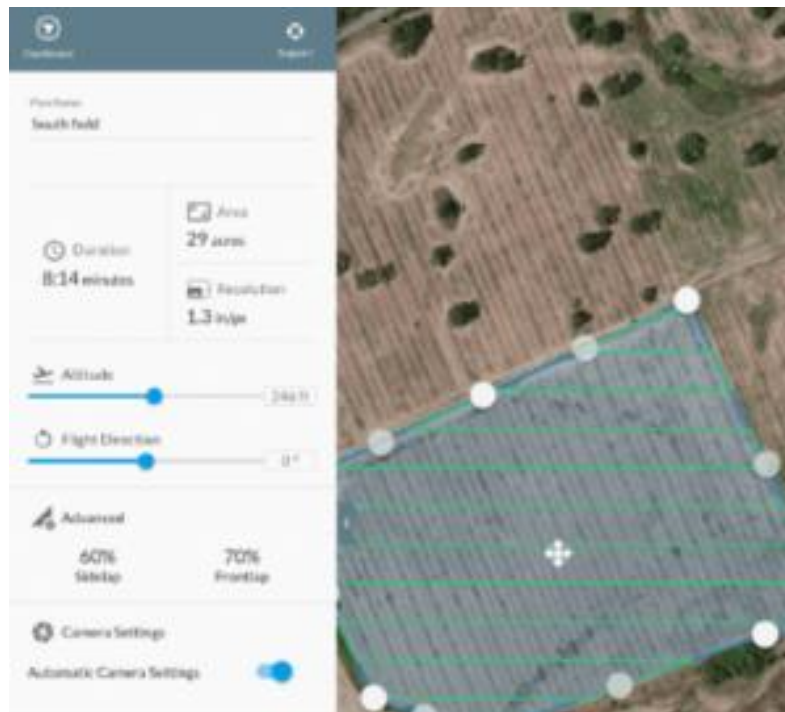


Figura 15: Interfaz de DroneDePloy

La planificación del vuelo del dron es realmente sencilla y muy similar a la aplicación de Pix4d. Primero habrá que seleccionar el área que se quiere sobrevolar y seguidamente elegir los diferentes parámetros como la altura o el frontlap/sidelap. Seguidamente la aplicación hará un checklist de que todo esté correcto y el dron comenzará el vuelo de forma autónoma.

Una funcionalidad interesante es que es posible añadir un vuelo circular al final de la misión. Las fotografías generadas en este último pase serán muy útiles a la hora de conseguir mejores reconstrucciones en 3D.

8. Planificador de vuelo escogido

La conclusión a la que se llega después de estudiar todos los planificadores de vuelo disponibles en el mercado es que una buena opción para comenzar a realizar las imágenes de la finca estudiada es mediante el software Pix4d Capture.

Es el software que a priori mejor interfaz tiene para el uso que el trabajo requiere de toma de imágenes aéreas en fincas agrícolas.

8.1 Instalación

La instalación de cualquier software externo en el dron es algo más laboriosa, ya que en la versión normal la pantalla en la que se realizan todas las operaciones es un

Smartphone propio o una tablet, por lo que la instalación del software se realiza mediante Appstore o Playstore, ya sea iOS o Andorid respectivamente.

Pero en nuestro caso, al tener la versión Phantom 4 pro +, en vez de tener que utilizar nuestro Smartphone, nos viene ya incorporado en el mando un Smartphone android que viene configurado por el fabricante DJI.

Por lo tanto para instalar el planificador lo primero que hay que hacer es descargar la versión de Pix4d Capture en su versión apk, que se encuentra en el siguiente directorio:

<https://apkpure.com/es/pix4dcapture/com.pix4d.pix4dmapper>

Una vez descargado hay que copiarlo en la tarjeta SD que lleva incorporada el mando del drone.

Buscamos la carpeta en la que se ha copiado el archivo apk y se procede a instalar el planificador.

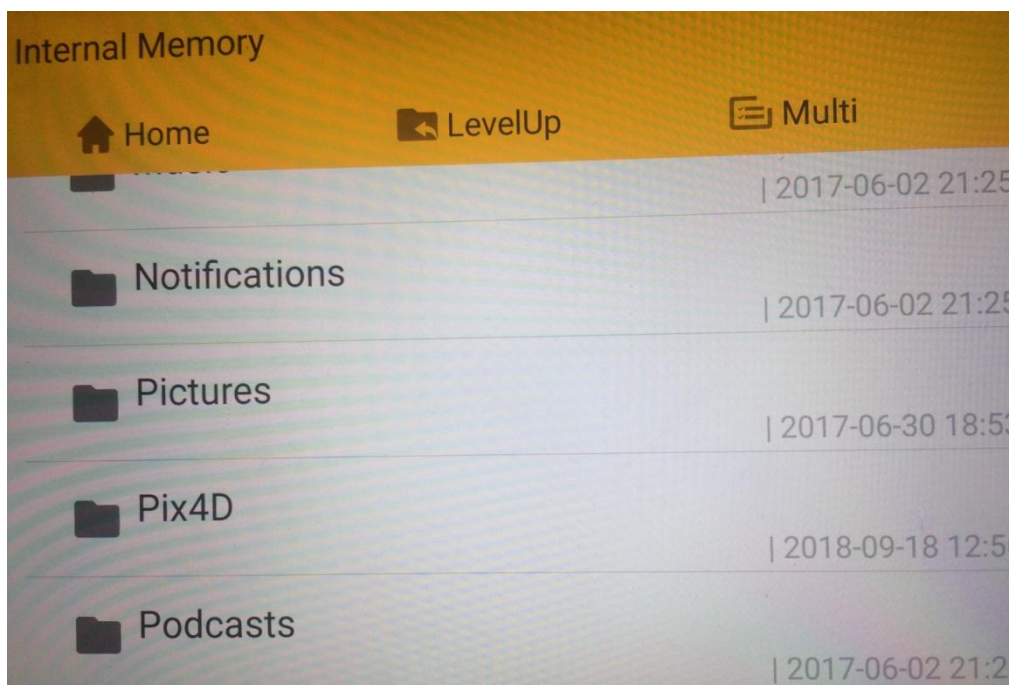


Figura 16: Carpeta de instalación

8.2 Funciones utilizadas

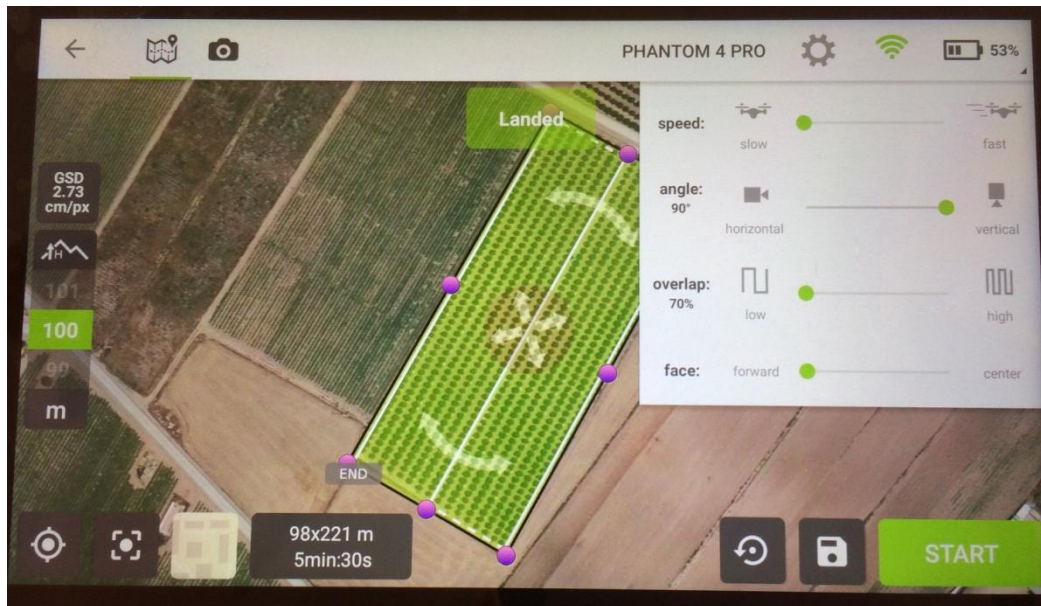


Figura 17: Parámetros seleccionados

Solapamiento: Es el porcentaje de similitud que habrá cada dos fotos, es decir, si existe un 100% de solapamiento equivale a que las dos fotos son exactamente iguales.

Altura de vuelo: Este parámetro se selecciona desplazando la barra verticalmente que se encuentra a la izquierda de la imagen.

Ángulo: Es el ángulo sobre el terreno que va a tener la cámara a la hora de realizar las fotografías. En este caso se selecciona 90 grados, ya que van a ser imágenes cenitales.

Velocidad: Es la velocidad a la que se toman las imágenes, es decir si se disminuye la velocidad de vuelo, el tiempo de vuelo será menor, pero las imágenes serán menos precisas y se podría obtener un desalineamiento si no se selecciona la velocidad más baja.

Conexión cable mando: Para poder vincular el drone con el planificador de vuelo, es necesario conectar la conexión USB que tiene el mando por la parte trasera a una conexión micro-USB que incorpora. Si no se realiza esta operación, el drone no es capaz de vincularse con el controlador de vuelo y no puede comenzar el vuelo.



Figura 18: Conexión para vincular el drone con el planificador de vuelo

9. Finca estudiada

La finca que se utiliza para realizar el trabajo será una finca propiedad de la empresa hortofrutícola BELSAN que se encuentra en el término municipal de Torre-Pacheco.

Los motivos por los que se ha escogido esta finca es porque esta finca se encuentra alejada de núcleos urbanos y dónde con previa autorización por parte de la empresa, está permitido la grabación y toma de fotografías aéreas.

Además la plantación es de tipo monocultivo de limones donde el espesor y separación entre árbol es relativamente pequeña, por lo tanto será un gran reto llevar a cabo el conteo.



Figura 19: Finca de estudio

10. Primera prueba

Una vez instalado el planificador se procede a realizar la primera captura de imágenes para después analizarlas y ver las posibles mejoras.

Al ser la primera la altura de vuelo se va a fijar en 100 metros y un solapamiento al 70%, ya que varios artículos recomiendan esta altura de vuelo para realizar labores de conteo: <http://www.efilogy.es/industria40/drones-en-agricultura/>

Se han escogido estos parámetros de vuelo, ya que una altura de 70 metros es la mínima a la que posiblemente se tendrán que realizar las pruebas, ya cuanto menor sea la altura de vuelo mayor será el número de fotografías que tendremos que tomar.

También se ha escogido el porcentaje de solapamiento al 70%, ya que es el mínimo que la herramienta permite seleccionar y al igual que la altura de vuelo influye en el número de imágenes capturadas, el solapamiento también y cuanto menor sea el solapamiento menor será el número de imágenes.

Hay que tener en cuenta que realizando estas pruebas lo que se pretende conseguir son las condiciones y parámetros más ajustados, que finalmente permitan obtener un mejor conteo de los árboles.

Por lo tanto el número de imágenes es importante, ya que cuanto menor sea el número de imágenes, menor será el tiempo de procesamiento en la herramienta que desarrollemos para realizar el stitching y el conteo.

Es por ello que llevan a cabo diferentes pruebas para poder comparar y llegar a la conclusión de cuál es la mejor altura de vuelo.

El resultado de la prueba es el siguiente:

- 36 imágenes
- Espacio: 297 MB
- Hora: 20:05

Después de analizar las imágenes, nos fijamos en la interfaz del planificador de vuelo:



Figura 20: Interfaz Pix4DCapture

De las 36 imágenes que ha realizado el dron se puede observar que solo es importante la franja central marcada en rojo en la imagen anterior, ya que es la franja en la cual se captura todo el ancho de la finca:

Imagen de banda lateral:

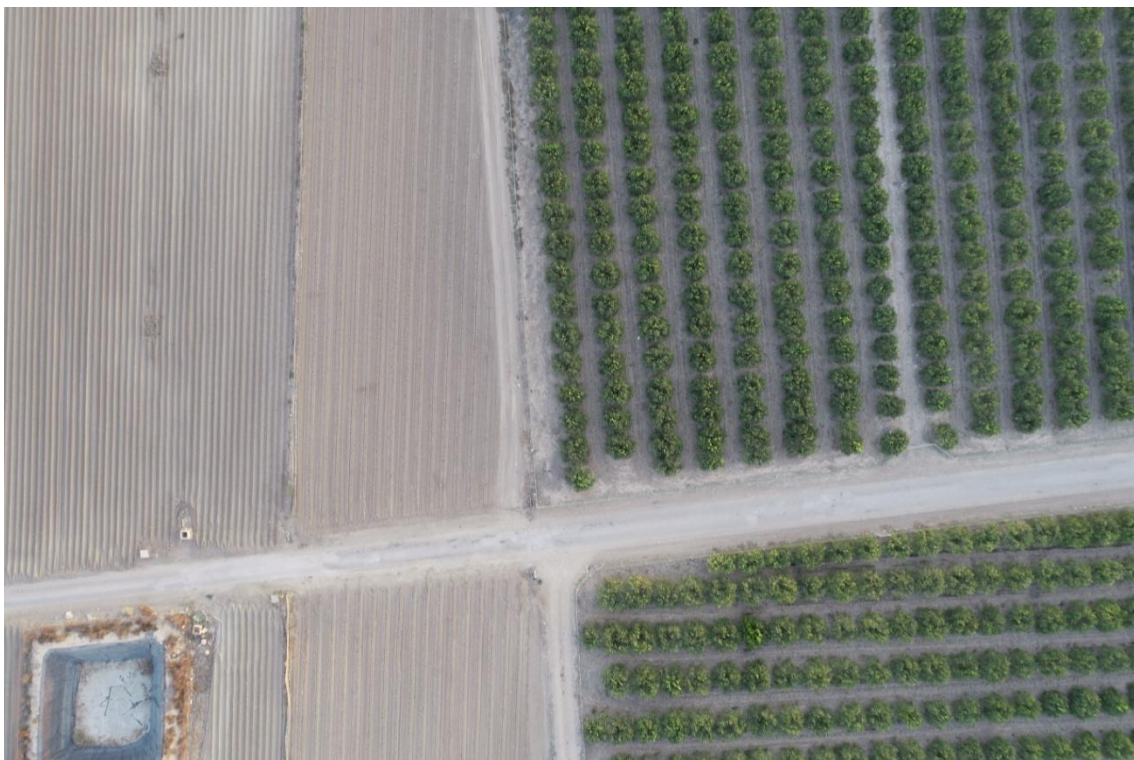


Figura 21: Banda lateral

Imagen de la banda central:

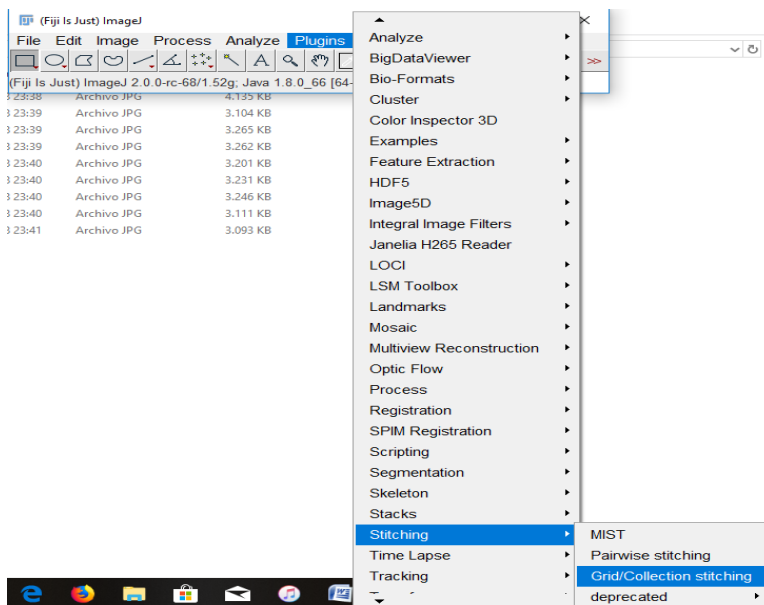


Figura 22: Banda central

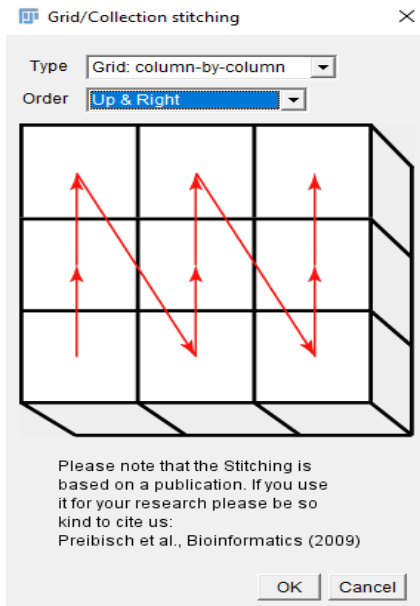
Descartando las imágenes de las bandas lateral, nos quedamos con las 9 imágenes que ha capturado el drone en la banda central.

10.1 Stitching

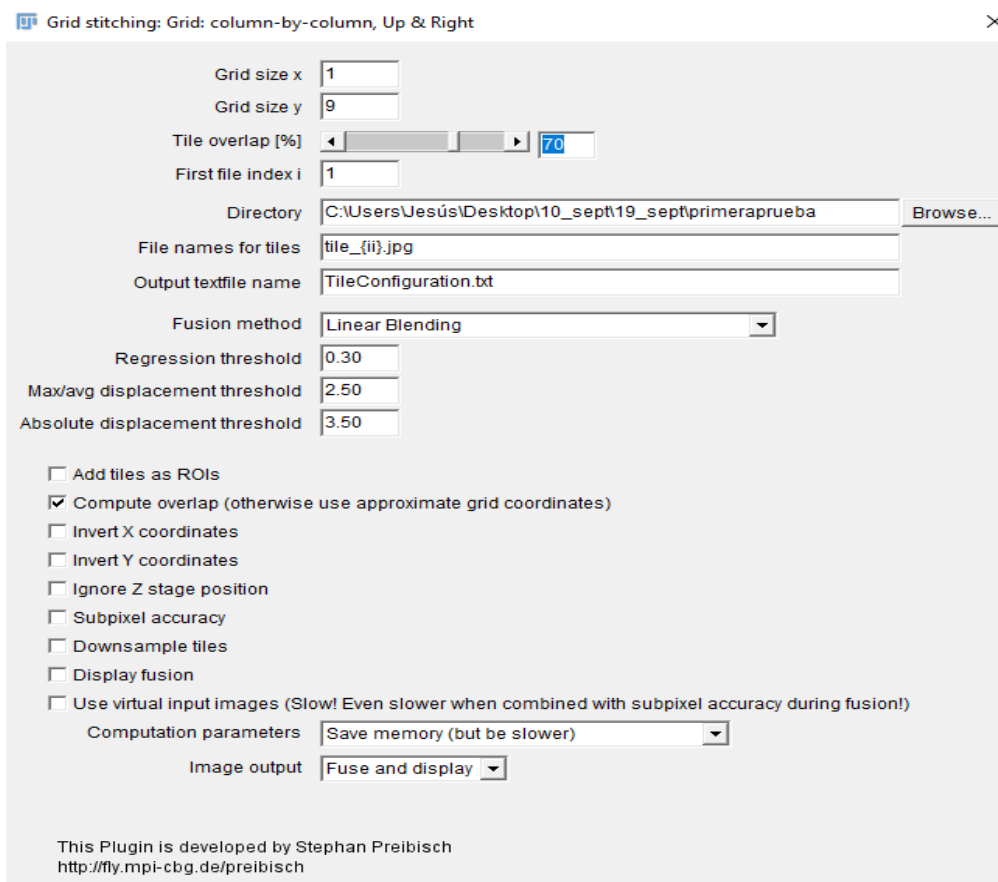
A continuación se procede a realizar el proceso de stitching con el software ImageJ:



Se utiliza el plugin de stitching dentro de ImageJ con la opción de Grid/Collection que permite realizar el stitching según filas y columnas, como si fuera una matriz:



Se selecciona la opción de columna en columna, ya que en nuestro caso el stitching se basa en unir las 9 imágenes de la finca que están en esa disposición.



Como el grid que se quiere obtener es una matriz que tiene 1 columna y 9 filas seleccionamos 1 en el eje X y 9 en el eje Y.

En tile overlap seleccionamos un 70% que es el solapamiento que en principio tienen las imágenes según los parámetros del planificador de vuelo.

Las imágenes tienen que ser llamadas como: tile_{ii}.jpg

El plugin leerá las imágenes en el siguiente orden: tile_01.jpg, tile_02.jpg...tile_09.jpg

El resultado obtenido es el siguiente:



Figura 23: Stitching nueve imágenes

Como se observa el stitching no es muy bueno, ya que el programa trata de unir 9 imágenes con mucho solapamiento.

Una solución a este problema puede ser descartar imágenes de la siguiente forma:

- Al tener 9 imágenes se puede descartar la imagen número 1 y la 9, que son la primera y la última.

- Nos quedamos las imágenes 2,5,8
- Se descartan las imágenes 1,3,4,6,7,9

Se han descartado las imágenes 1,9 porque son las que menos interesan, ya que son las que menos superficie de árboles capturan al ser las últimas de la columna.

Por lo tanto se coge la imagen 2 y se van descartando las dos siguientes.

El inconveniente de este método es que no se conoce realmente el solapamiento que tienen las imágenes, por lo tanto hay que ir probando valores de solapamientos y escoger el que mejor se ajusta.

Después de probar diferentes valores, el que mejor se ajusta es un 30% de solapamiento. Este es el resultado:

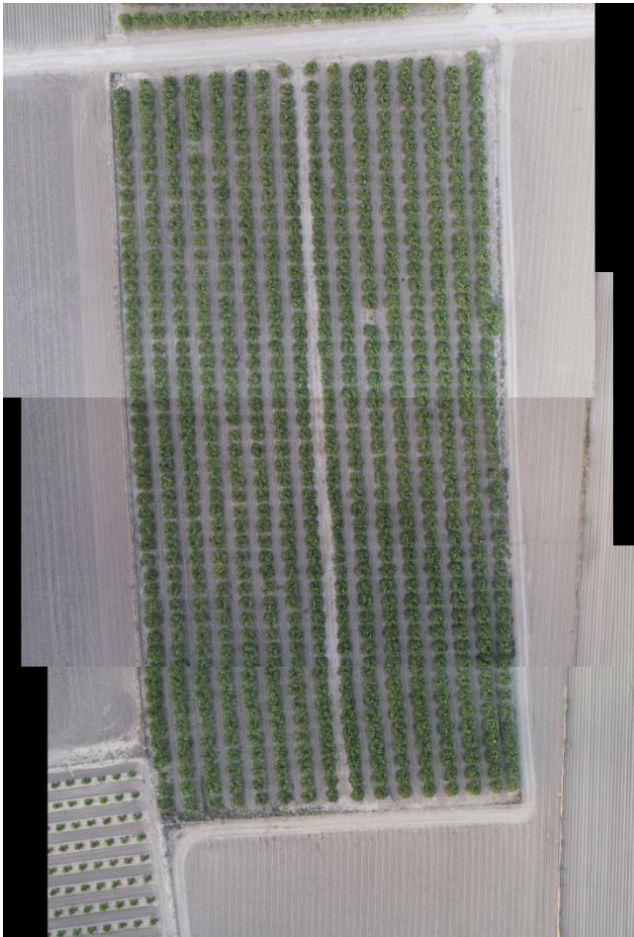


Figura 24: Stitching tres imágenes

Finalmente se ha realizado un buen stitching, pero es necesario hacer otra toma de imágenes, esto se debe a que la hora a la que se ha realizado la prueba la luz solar es mínima si se compara con horas centrales del día.

Este es un factor a tener en cuenta muy importante, ya que como se puede observar más adelante, a la hora de realizar el conteo de los árboles hay que binarizar la imagen y realizar una segmentación de los píxeles modificando el Threshold. Con estas operaciones es posible diferenciar los árboles de la tierra mediante una modificación de los umbrales de color.

11.Segunda prueba

En esta segunda prueba como antes se ha comentado hay que realizar la toma de imágenes a una hora central del día.

Esta vez se realizarán las fotografías a las 13:53.

Tenemos 4 métodos para realizar el stitching:

- Linear blending
- Average
- Median
- Intensity of random input tile

Por lo tanto se va a realizar el stitching con los 4 métodos posibles y después de analizarlos se escogerá el que proporcione un mejor resultado.

Ha habido que realizar un pequeño ajuste en el parámetro de solapamiento, ya que con el 30% alguna imagen no se ajustaba bien a la siguiente, esto puede ser debido a una posible ráfaga de viento o a la tonalidad de la imagen, ya que la luz solar en la imagen no es igual en este caso, por lo que utilizando un 13% se obtienen muy buenos resultados.

11.1 Tipo de stitching

Utilizando el mismo método que en la primera prueba se realiza el stitching con ImageJ obteniendo el siguiente resultado para cada método:



Figura 25: Linear blending



Figura 26: Average



Figura 27: Intensity of random input file



Figura 28: Median

Se puede comprobar que los 4 métodos realizan un stitching muy aceptable, pero a la hora de trabajar con el algoritmo de conteo a priori la mejor opción parece la de Intensity of random input tile, ya que aunque se nota el corte de la unión de la segunda imagen con la tercera, es en donde más nítidos se ven los árboles, que finalmente es lo más importante a la hora de proceder con el conteo.

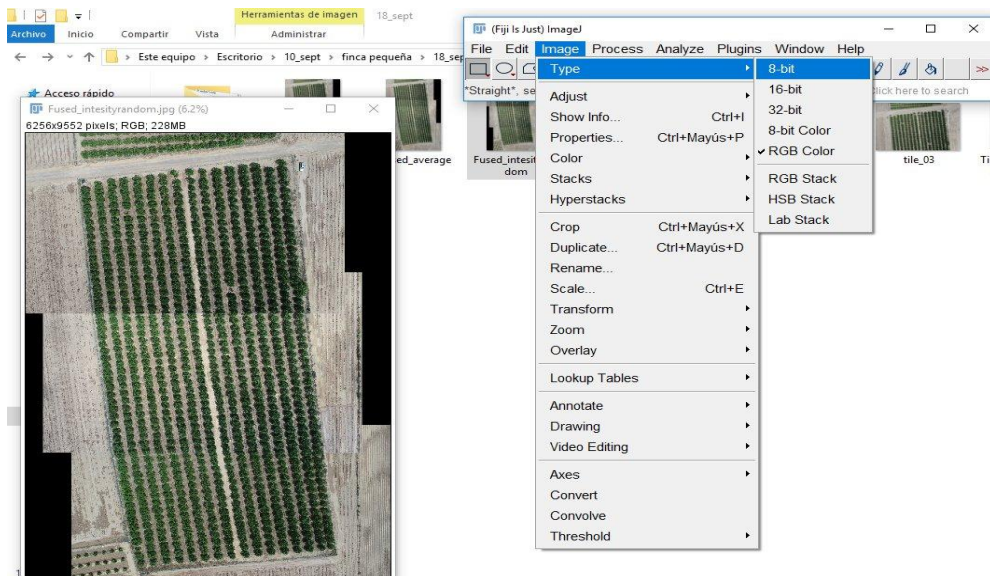
12. Proceso de conteo

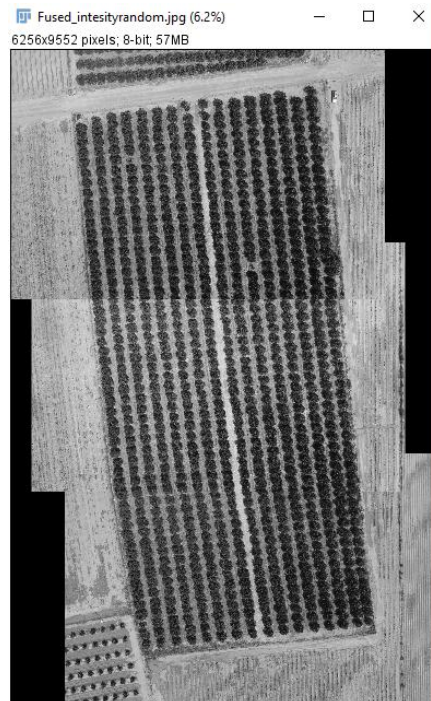
Una vez elaborado todo el proceso de stitching vamos a comenzar con la segunda parte del trabajo que consiste en realizar una serie de operaciones a través del software ImageJ.

12.1 Cambio a 8 bits

Para ello, antes de ajustar los filtros de la imagen lo primero de todo es ajustar la imagen a los requisitos que exige ImageJ para poder modificarla.

ImageJ exige pasar del formato de imagen original RGB a un formato de 8bit, mediante la siguiente operación:

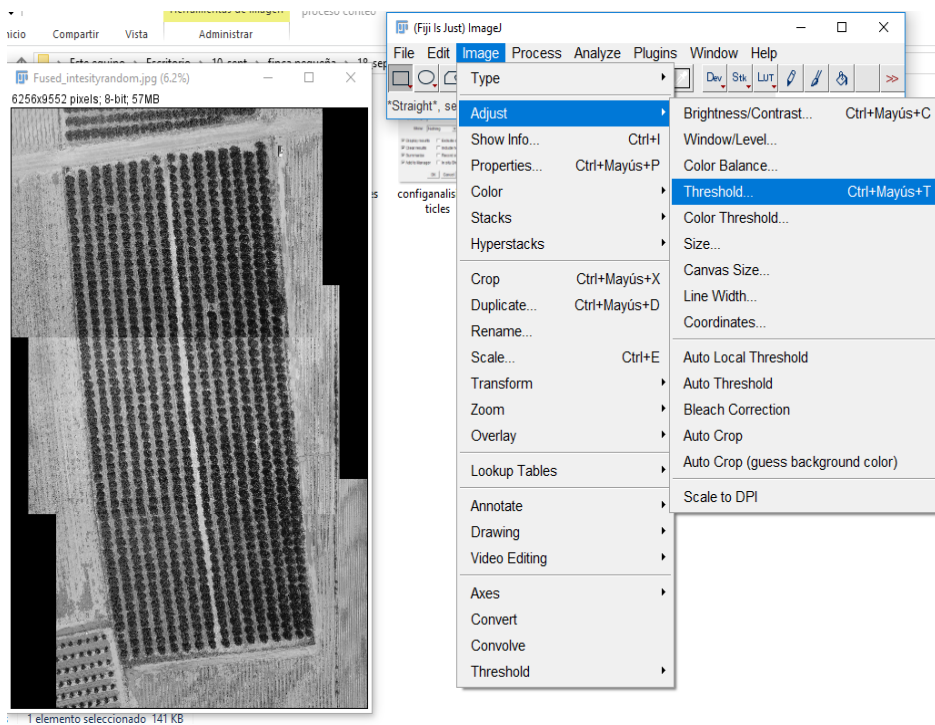


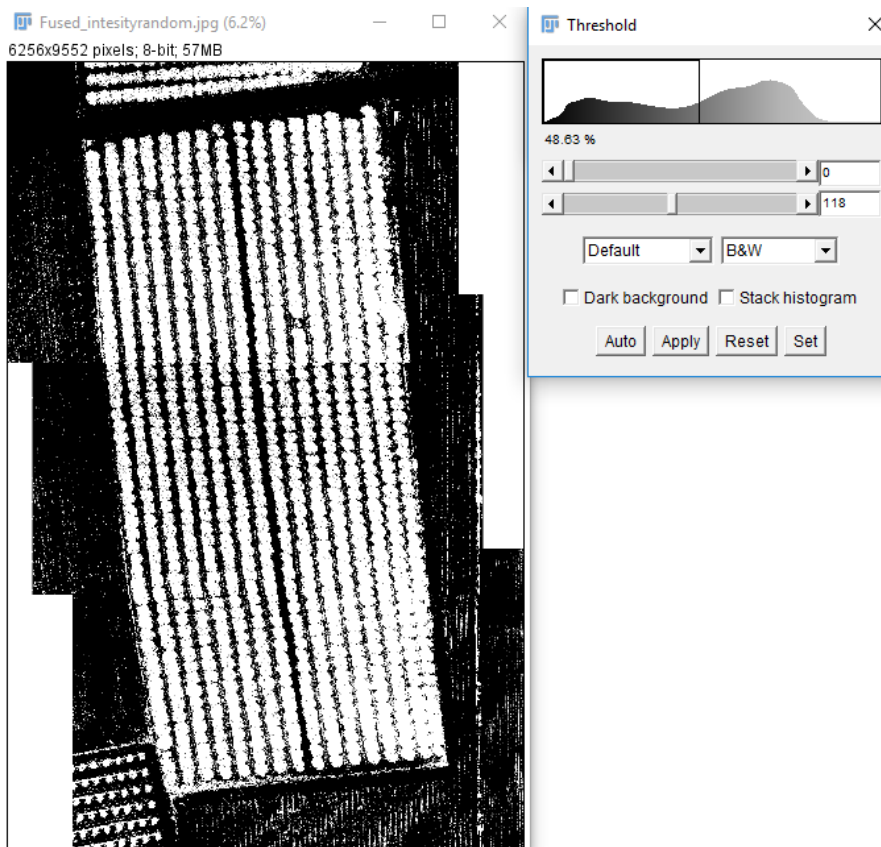


Lo que se consigue es pasar de una imagen RGB de 228MB a una imagen en escala de grises de 57MB.

12.2 Threshold

Ahora se procede a filtrar la imagen para poder diferenciar mediante las tonalidades diferentes tonalidades los árboles de los demás elementos de la imagen que no tienen importancia, como es la tierra o elementos que no formen parte de los árboles.





Al realizar esta operación es imprescindible desmarcar la opción de dark background, para que consiga diferenciar los árboles en color blanco y los demás elementos de la finca en color negro.

Con esta opción de modificar el threshold hay diferentes opciones de tonalidades, existe la opción de rojo y negro, pero se ha escogido la opción de blanco y negro porque es la más intuitiva para este tipo de operaciones.

A continuación se ajusta manualmente el nivel de threshold para que se puedan diferenciar los arboles de la tierra lo mejor posible, obteniendo el resultado en la imagen anterior.

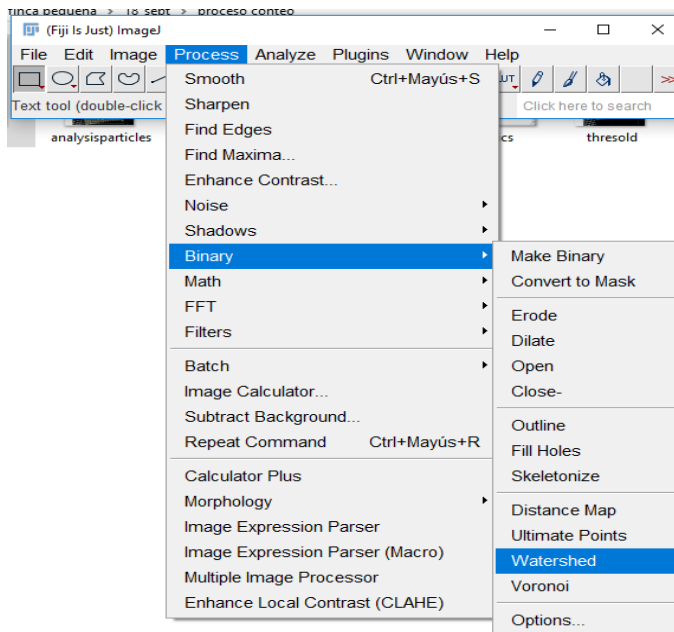
12.3 Watershed

Si nos fijamos en la imagen anterior ya filtrada se puede observar como muchos árboles están juntos, es decir, al ser la separación entre árboles tan pequeña es difícil diferenciar las formas redondas de los árboles.

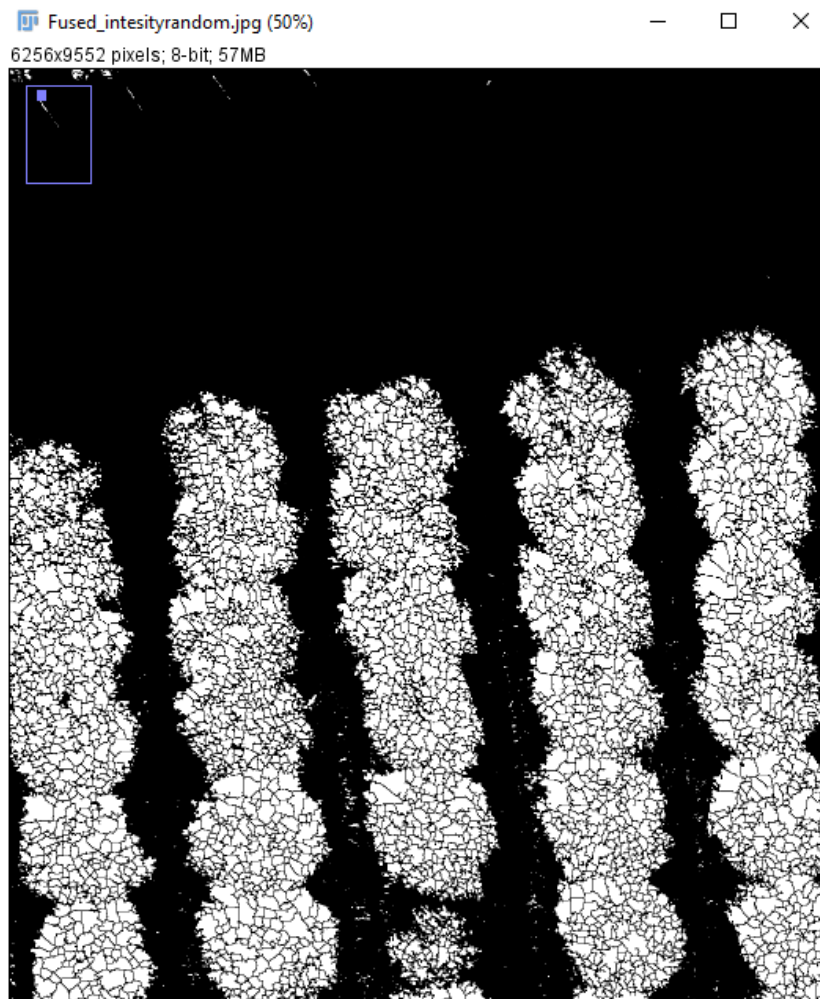
Existe un algoritmo, llamado watershed que consigue dividir los árboles, que son las porciones que están en color blanco en porciones circulares, ya que como se puede observar muchos árboles aparecen como si estuvieran juntos, pero en la realidad esto no es así. Por lo tanto es necesaria la aplicación de este algoritmo.

Como el plugin no estaba disponible en la herramienta, se ha tenido que buscar a través de <https://github.com/> y se ha encontrado el siguiente código disponible en el Anexo 1.

A continuación se ha instalado en la carpeta de Process/Binary:



Al aplicar el algoritmo de watershed en la imagen se puede observar lo siguiente:



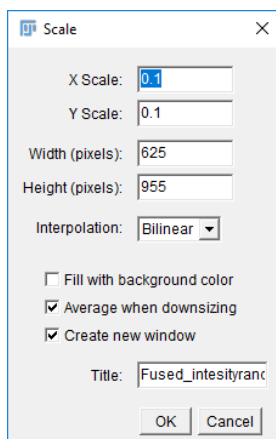
Si nos fijamos detenidamente en la imagen se observa como el algoritmo no ha funcionado como estaba previsto, ya que la división la ha realizado dentro del propio árbol, por lo que a la hora de analizar las partículas blancas va a contar dentro de cada árbol varias a la vez. Por lo tanto hay que plantear una solución a este problema.

Una posible solución a este gran problema puede ser el de reducir el número de píxeles de la imagen, ya que las imágenes que han tomado el drone son de mucha calidad, pero para este trabajo no es tan importante el número de píxeles.

Al tener tanta resolución esto provoca que la imagen tenga un mayor número de píxeles, por lo tanto al aplicar el threshold se va a tener dentro de cada árbol una mayor cantidad de píxeles negros, que son los equivalentes a la tierra. Esto ocurre en la imagen de mayor resolución, por lo tanto si se consigue disminuir el número de píxeles vamos a tener menos cantidad de píxeles negros por árbol y esto favorecerá a que el algoritmo watershed que divide las partículas pueda actuar de una forma más eficiente hacia lo que se pretende conseguir.

12.4 Escalar imagen

La imagen que se va a analizar tiene una resolución de 6256x9552 píxeles. Por lo tanto se va a disminuir la cantidad de píxeles por diez y comprobar si los resultados son aceptables. Para ello se aplica la opción Scale, que nos ofrece ImageJ dentro de su menú:



Si se pretende reducir por diez el número de píxeles habrá que multiplicar por 0,1 tanto en el eje X como en Y:



Con esta operación lo que se consigue es pasar de una resolución de 6256x9552 pixeles a 625x955 y reducir el tamaño de la imagen a 2,3MB. Gracias a esta operación también se consigue reducir el tiempo de procesamiento al disminuir su tamaño.

Después de escalar la imagen, se modifica el formato de imagen a 8 bits, se aplican las operaciones de threshold y para terminar el algoritmo de división watershed obteniendo el siguiente resultado:

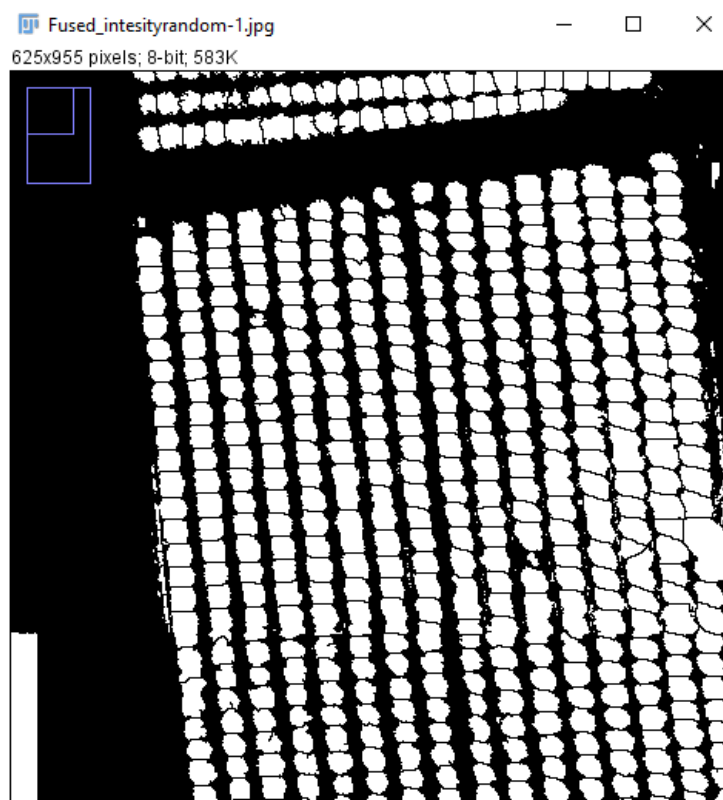


Figura 29: Aplicación de algoritmo Watershed

Como podemos observar en la imagen anterior, esta vez sí se ha logrado que el algoritmo haga correctamente la división de los núcleos, por lo tanto la solución propuesta de escalar la imagen y reducir el número de píxeles por diez es una muy buena opción para continuar el proyecto.

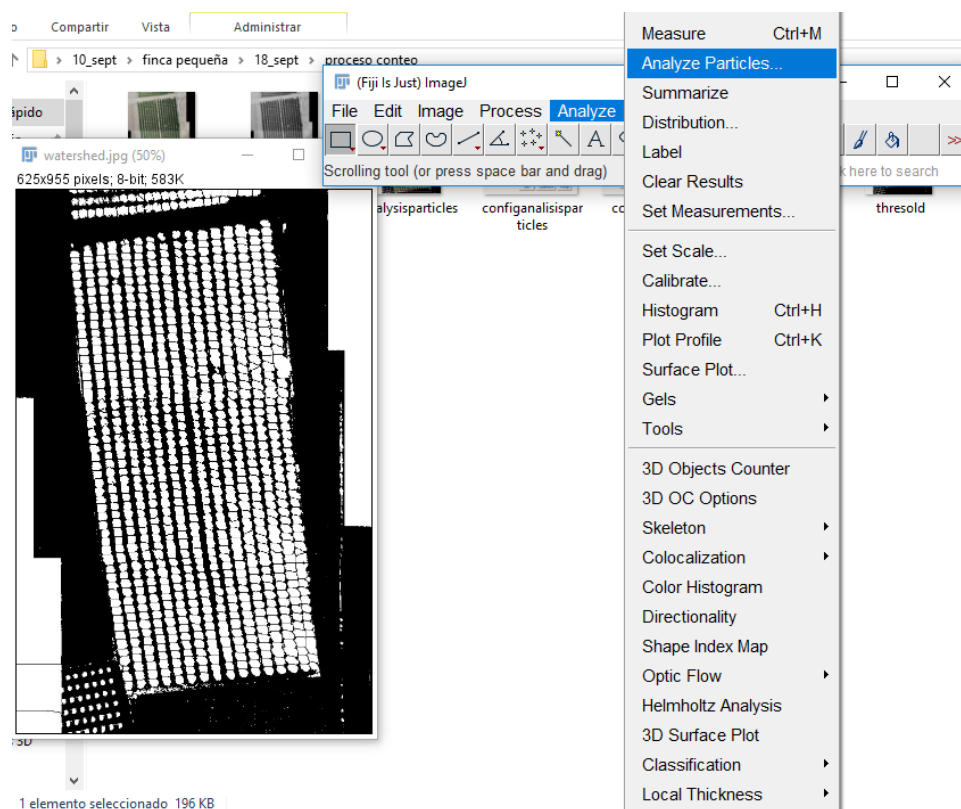
Ya estamos preparados para realizar la última operación que consiste en analizar y contar los núcleos según el rango de circularidad y la cantidad analizada en píxeles cuadrados.

Antes de pasar a realizar el análisis de los núcleos, se ha de comprobar la diferencia que existe entre realizar la toma de imágenes en una hora central del día, cuando el sol está incidiendo sobre la superficie a analizar de forma perpendicular, y cuando se realizan en concreto sobre las 20:00 de un día de verano. Estas últimas imágenes tomadas a las 20:00 corresponden al apartado de la primera prueba.

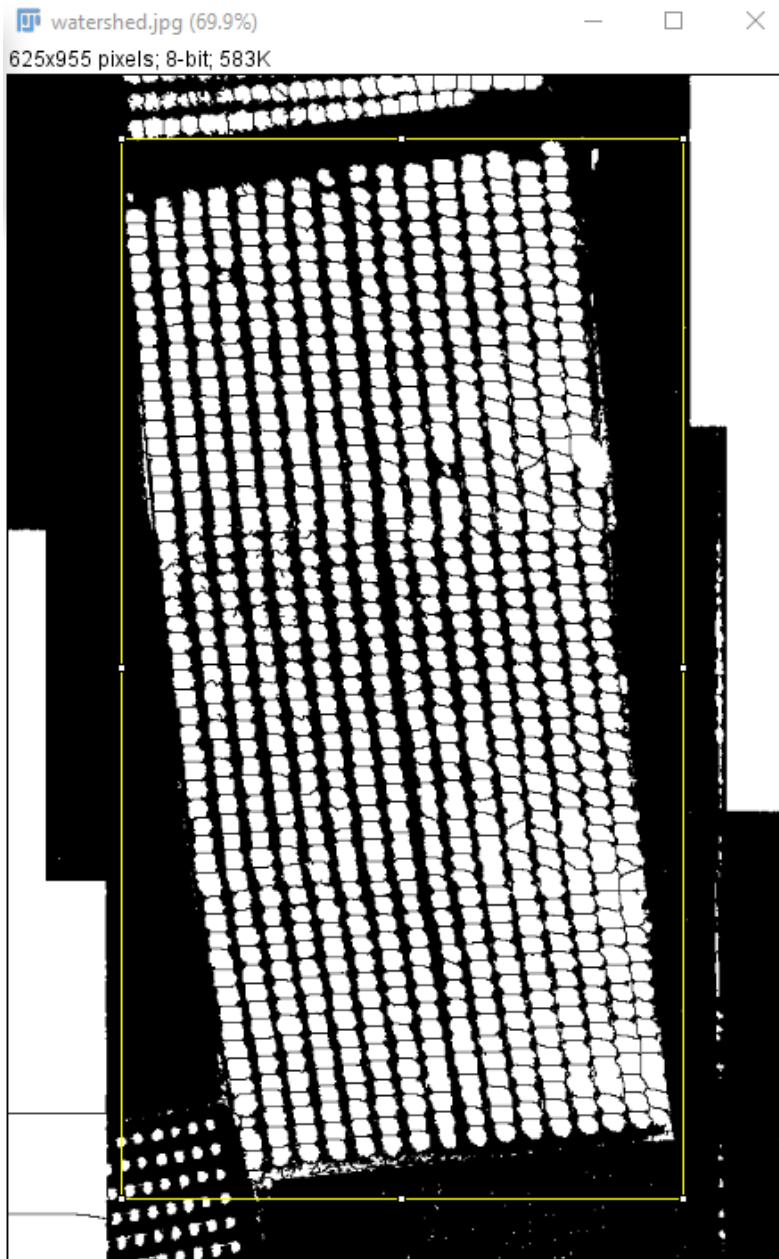
12.5 Analyze Particles

Una vez realizado el stitching y todas las demás operaciones anteriores pasamos a la última parte del trabajo, que sería realizar un conteo de los árboles con la imagen filtrada usando la función de threshold y el algoritmo de división de núcleos watershed.

Para ello se va a utilizar una función de análisis que tiene ImageJ llamada Analyze Particles



Esta función lo que permite es contar y analizar cada partícula en función del tamaño en pixeles cuadrados y de la circularidad.



Para comenzar habría que seleccionar el área que se va a analizar, ya que así se pueden descartar las partículas que no tienen importancia en el análisis.

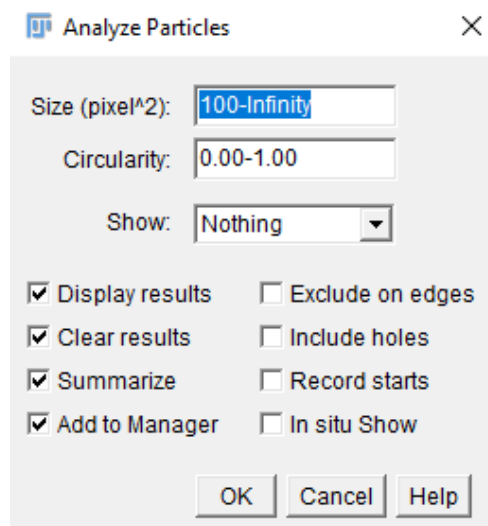


Figura 30: Parámetros Analyze Particles

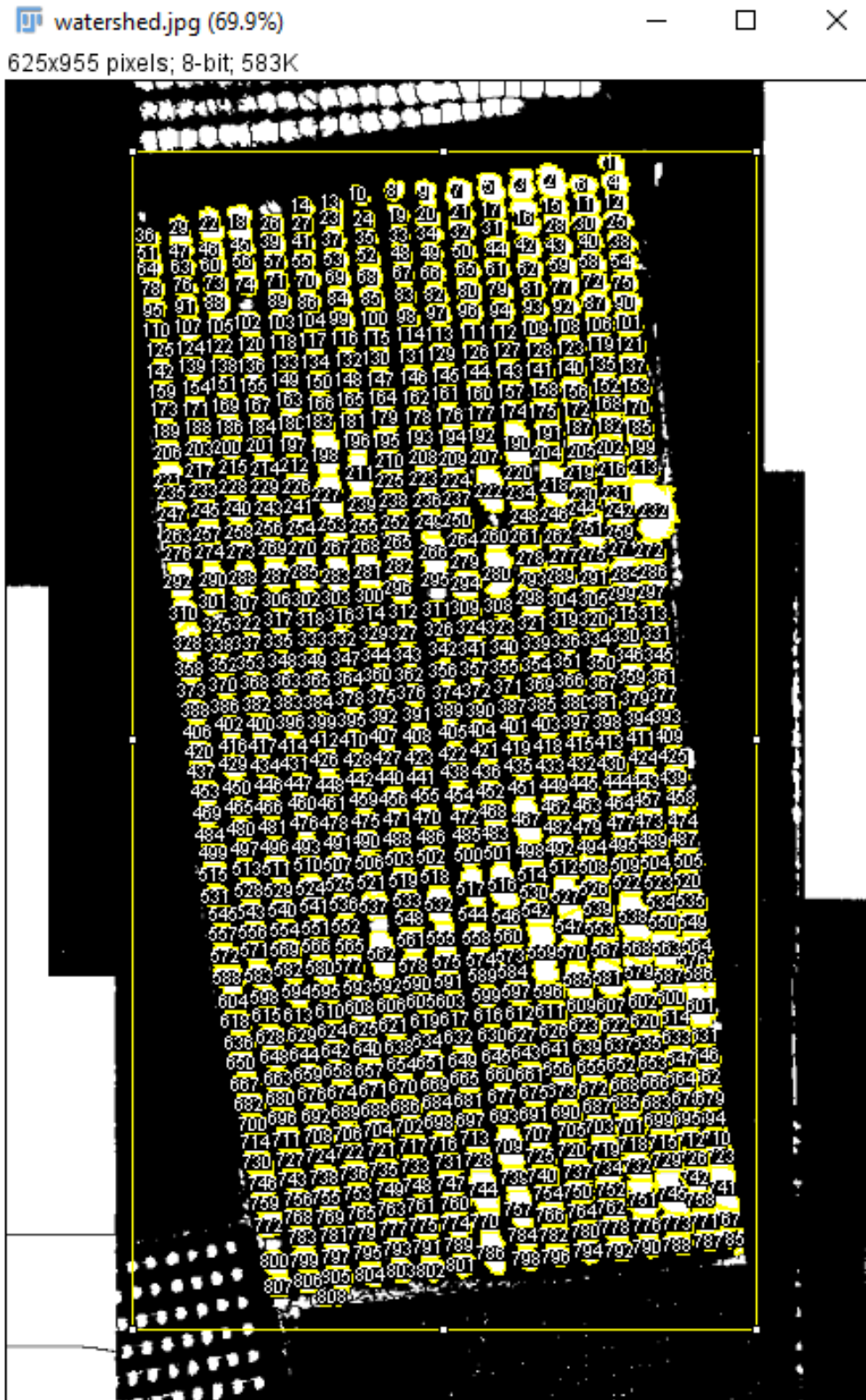
En el apartado de size seleccionamos el tamaño de las partículas que se quieren analizar en pixeles cuadrados.

Se marca un valor de 100 pixel² hasta infinito, ya que teniendo en cuenta que la imagen tiene un tamaño de 625x955 pixeles, los árboles que se quieren analizar tienen un tamaño aproximado de 10x10 pixeles tanto en el eje X como en Y. Por lo tanto el algoritmo analizará las partículas que tengan como mínimo ese tamaño, ya que se ha seleccionado el intervalo hasta el infinito.

En el apartado de circularidad, según el desarrollador una partícula puede tener una circularidad entre 0 y 1, siendo 1 el valor de un círculo perfecto, por lo tanto se selecciona todo el intervalo desde 0 hasta 1, ya que los núcleos no son círculos perfectos.

Para un mayor aporte de información en el análisis, se seleccionan las opciones de Display results, Clear results, Summarize y Add to manager, ya que así mostrará en pantalla todos los parámetros analizados.

A continuación se aplica el algoritmo de análisis obteniendo los siguientes resultados:



Summary

Slice	Count	Total Area	Average Size	%Area	Mean
watershed.jpg	808	171151	211.821	44.594	255

Una vez aplicado el analizador, se obtienen los siguientes resultados:

	Nº Árboles	Precisión
Conteo ImageJ	808	93,10%
Conteo Manual	868	100%

Realizadas todas las operaciones de stitching y de conteo se obtiene una precisión de un 93,10%, por lo que teniendo en cuenta que la finca analizada tiene una separación muy pequeña respecto a las fincas de nueva plantación que respetan la separación reglamentaria de 2 metros, obtenemos un resultado muy aceptable dadas las complejidades del estudio y la edad de los árboles.

Una vez que se conocen las diferentes operaciones que hay que llevar a cabo con ImageJ, es hora de desarrollar el mismo proceso, pero en vez de realizar todas las operaciones seleccionadas de forma manual, programar un script que permita realizar todas las operaciones de forma automática, tanto las referidas a stitching hasta el posterior conteo.

13. Diferencia de luz solar en la toma de imágenes:

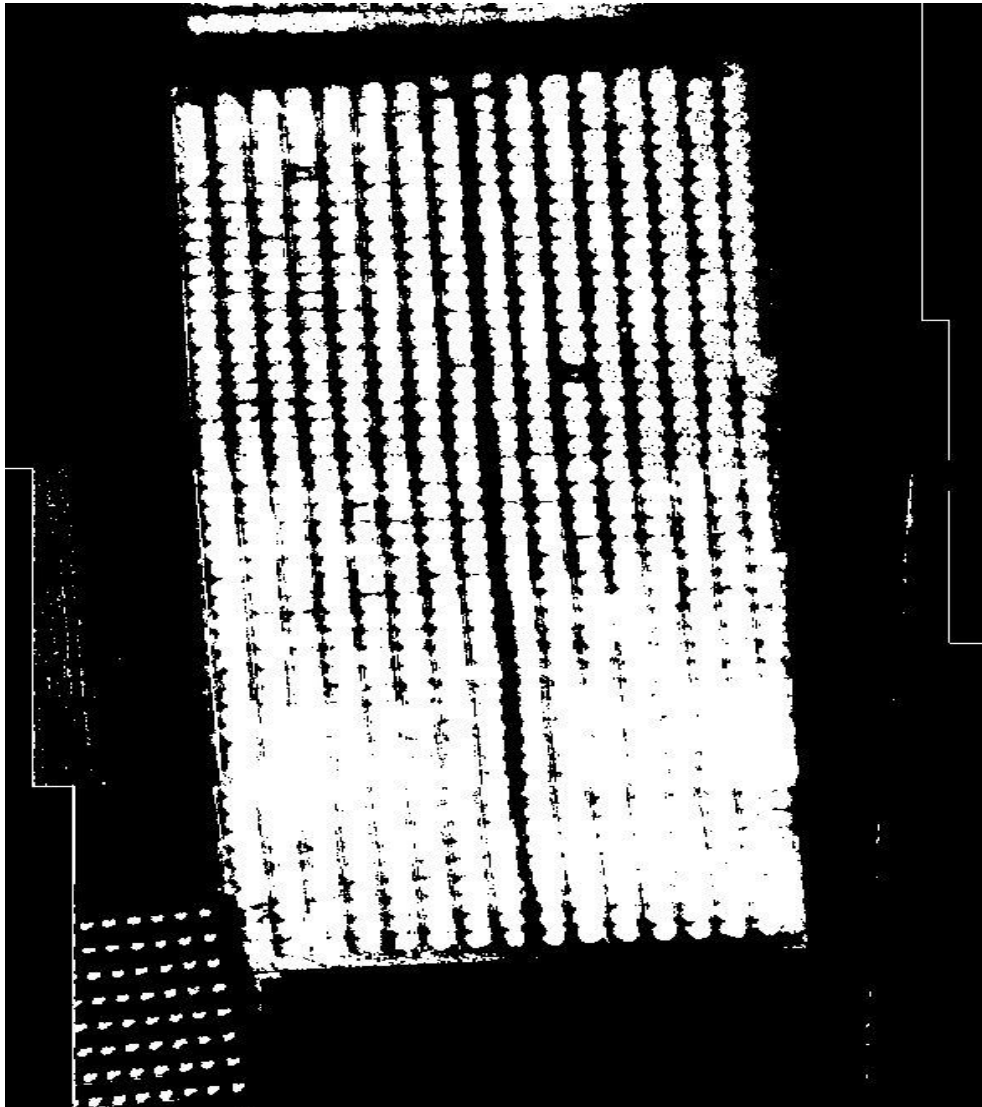


Figura 31: 20:00

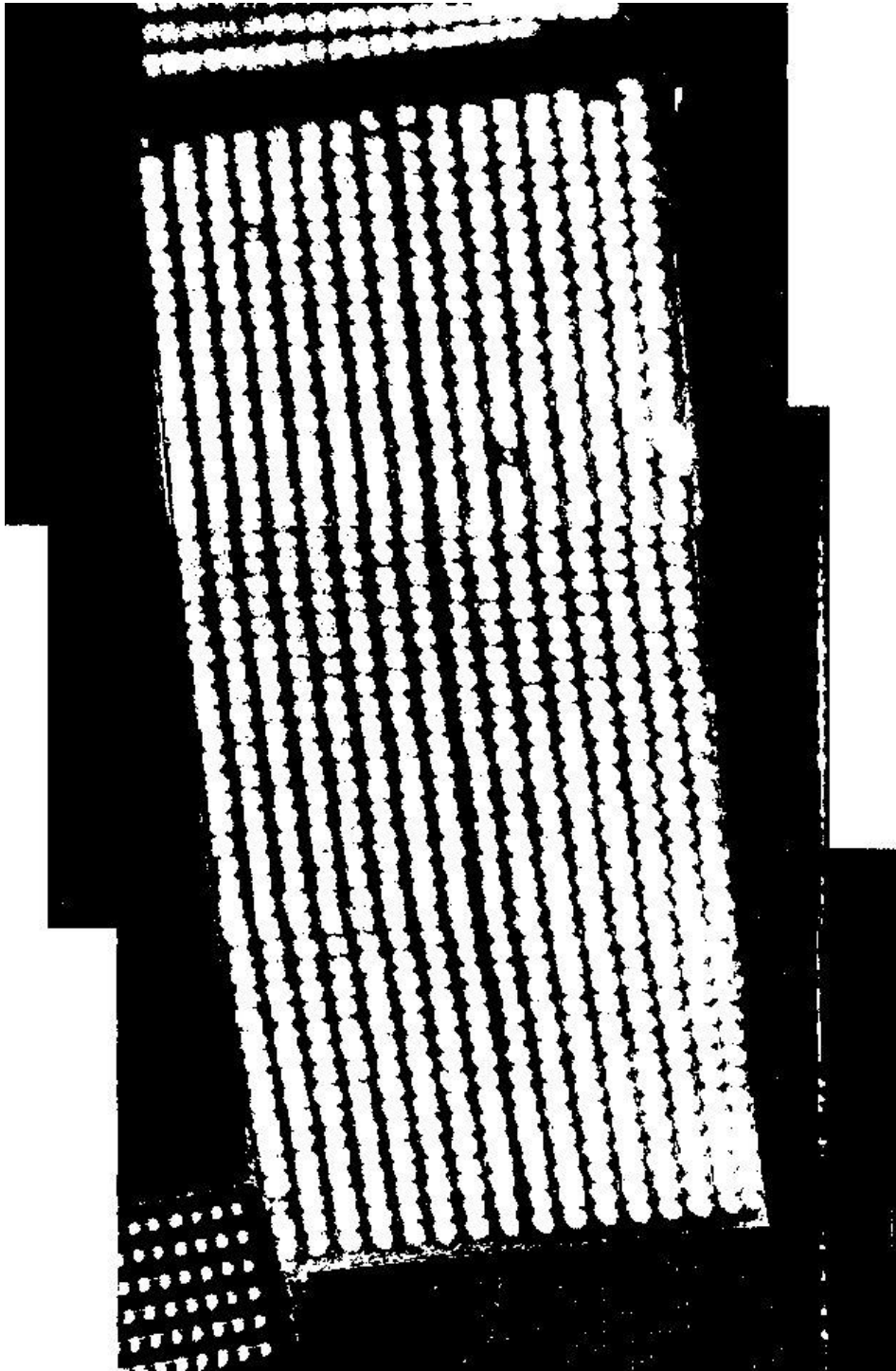
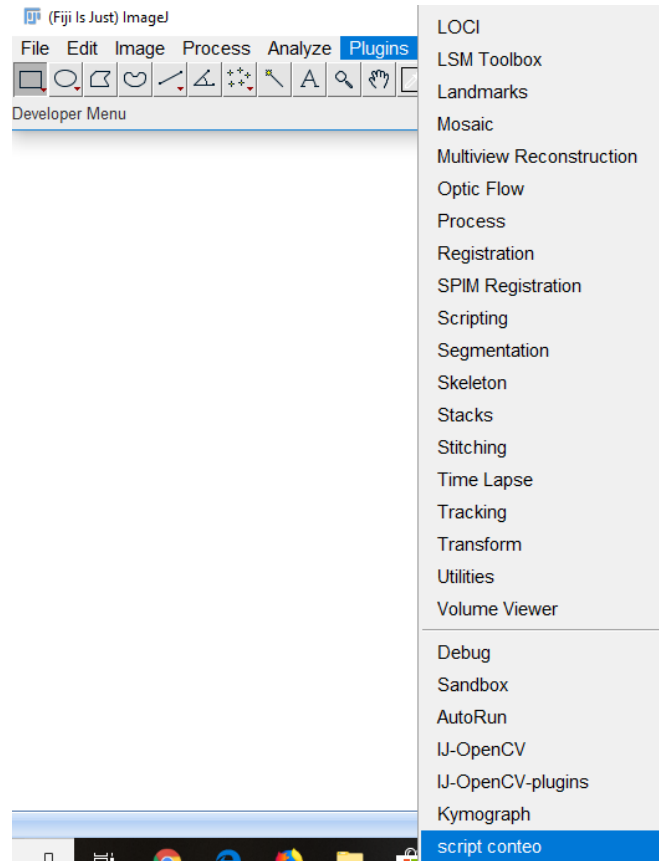


Figura 32: 14:00

Como se puede observar, cuando se realiza la toma de imágenes a las 20:00 horas, las siluetas de los árboles están menos diferenciadas respecto a las tomadas a las 14:00 horas.

Esto ocurre porque en las horas centrales del día el sol incide sobre la superficie capturada con el mismo ángulo que la cámara del dron captura las imágenes, por lo tanto se van a minimizar las zonas de sombras que provocarían los árboles en el caso

contados y su tamaño medio, y también guarda la imagen filtrada mediante la función de threshold y con las partículas contadas sobre la imagen.



Una vez realizado el script se guarda y se instala en el apartado de plugin, para que así pueda ser usado siempre que sea necesario de una forma muy sencilla.

14.1 Mejora

A continuación se va a realizar una pequeña modificación en el plugin. Se va a añadir una función que permite ajustar de una mejor forma el área que se pretende analizar, ya que anteriormente el plugin analizaba las partículas de toda la imagen, pero esta vez vamos a seleccionar un área poligonal más ajustada a la zona donde se encuentran los árboles.

El script con esta nueva función añadida se encuentra en el anexo 3.

```
18 makePolygon(61,84,178,925,564,879,471,25);|
```

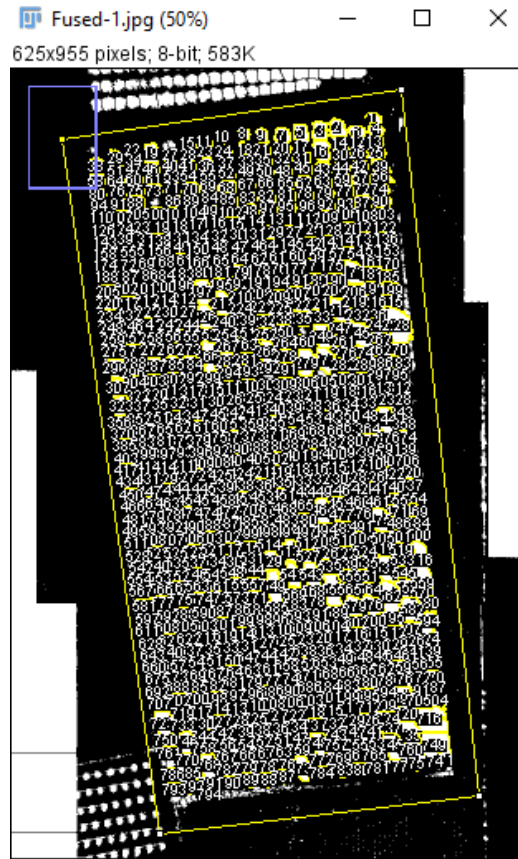


Figura 33: Perímetro seleccionado

Se procede a ejecutar el plugin, obteniendo los siguientes resultados en la carpeta elegida:

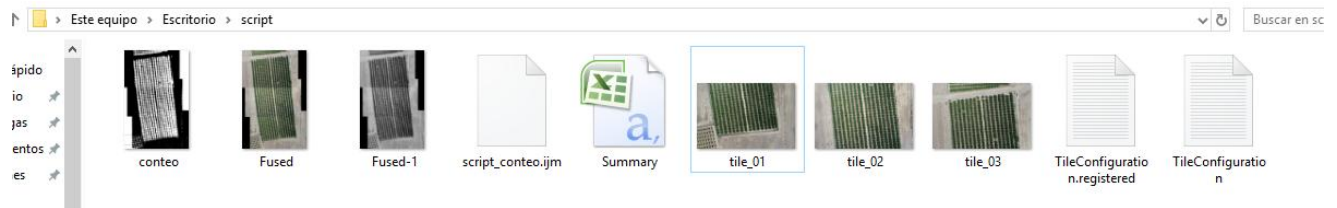


Figura 34: Resultados del script

El script guarda el archivo excel con el resumen del análisis y las imágenes más relevantes del proceso.

El tiempo de procesamiento del script es de 18,2 segundos.

Finished ... (18187 ms)

15.Comparativa imágenes drone/google maps

Una vez realizadas todas las pruebas pertinentes con el drone, se va a realizar una comparativa entre los resultados que se obtienen con las imágenes capturadas por el drone, y las correspondientes en google maps.

Para ello, lo primero es localizar la finca en google maps:



Localizada la finca, se intenta recortar la imagen completa para no tener que realizar el stitching, pero al recortarla se observa que no tiene la suficiente calidad como para realizar un análisis.



Figura 35: Imagen Google Maps de baja calidad

Al no tener la suficiente calidad habrá que aplicar un nivel más de zoom en google maps y realizar el stitching de dos imágenes, obteniendo el siguiente resultado:



Figura 36: Imagen Google Maps de alta calidad

Como se puede observar, el stitching que se ha realizado es de máxima calidad y apenas notamos los cortes de la unión. Esto es debido a que las dos imágenes que se han recortado en google maps tienen el mismo número de píxeles, ya que el satélite no toma muchas imágenes y luego realiza el stitching. El satélite lo que hace es tomar una imagen de grandes dimensiones a diferentes alturas, por lo tanto la finca completa tiene el mismo número de píxeles y el stitching realizado por ImageJ será mucho más fácil que el que se tiene elaborar con las imágenes tomadas desde el dron.

Una vez que se dispone de la imagen unida y con la suficiente calidad se procede a aplicar todas las operaciones de filtrado y los algoritmos de conteo:



Figura 37: Árboles contados

Slice	Count	Total Area	Average Size	%Area	Mean
Fused.jpg	736	120395	163.580	49.540	255

	Número de árboles	Precisión
Imágenes Drone	808	93,10%
Imágenes Google Maps	736	84,79%

Realizando una pequeña comparativa entre los resultados que se obtienen con las imágenes realizadas por el drone y las recogidas en google maps se pueden extraer las siguientes conclusiones:

- Al tomar las imágenes de google maps la calidad disminuye y al realizar el filtrado en threshold los contrastes en las imágenes realizadas por el drone van a ser mucho mayores que las tomadas en google maps, ya que al tomarlas con el drone se dispone de la libertad de escoger a qué hora realizar las capturas y la altura a la que se realizan, factores que tomando las imágenes del satélite no se pueden controlar.
- La diferencia entre realizar las imágenes con el drone y tomarlas de google maps es un error del 8,31% si se extraen de google maps. Esto equivale a que se dejan de analizar un total de 72 árboles.
- Por lo tanto no es aconsejable realizar el conteo con imágenes tomadas de google maps, ya que no es posible controlar ningún factor, desde la altura hasta la hora a la que se capturan las imágenes. Y además se añade un error de hasta el 8,31% en el conteo.
- Existiría una dependencia si se opta por utilizar imágenes de google maps, ya que en los casos de fincas de nueva plantación habría que esperar a que google actualizara las imágenes de sus satélites.

16. Presupuesto

Una parte esencial del trabajo es conocer realmente el coste que puede suponer la utilización de esta herramienta frente a las que ya existen en el mercado para comprobar si es o no viable.

	Unidad	Precio unitario	Total
Drone	1	1.515 €	1.515 €
Vuelo	0,5 Horas	100 €	50 €
Extracción imágenes	2 Horas	50 €	100 €
Stitching	0,5 Horas	50 €	25 €
Análisis	1 Horas	50 €	50 €
Informe resultados	2 Horas	50 €	100 €
TOTAL			1.840 €
Inversión inicial	1.515 €		
Coste estudio finca	325 €		

Se ha realizado una estimación de las horas que requiere cada fase del estudio y se ha fijado un precio por hora de trabajo.

Se ha decidido fijar un precio de 100€/hora de vuelo, ya que esta fase es la que más riesgo conlleva y en la que se requiere la presencia en la finca de un profesional que haga las planificaciones de vuelo según la demanda del agricultor.

El agricultor debería realizar una inversión inicial de 1515€ para la adquisición del drone. Es preferible realizar la compra del drone, ya que así puede utilizarlo para mediar las diferentes fincas que posea.

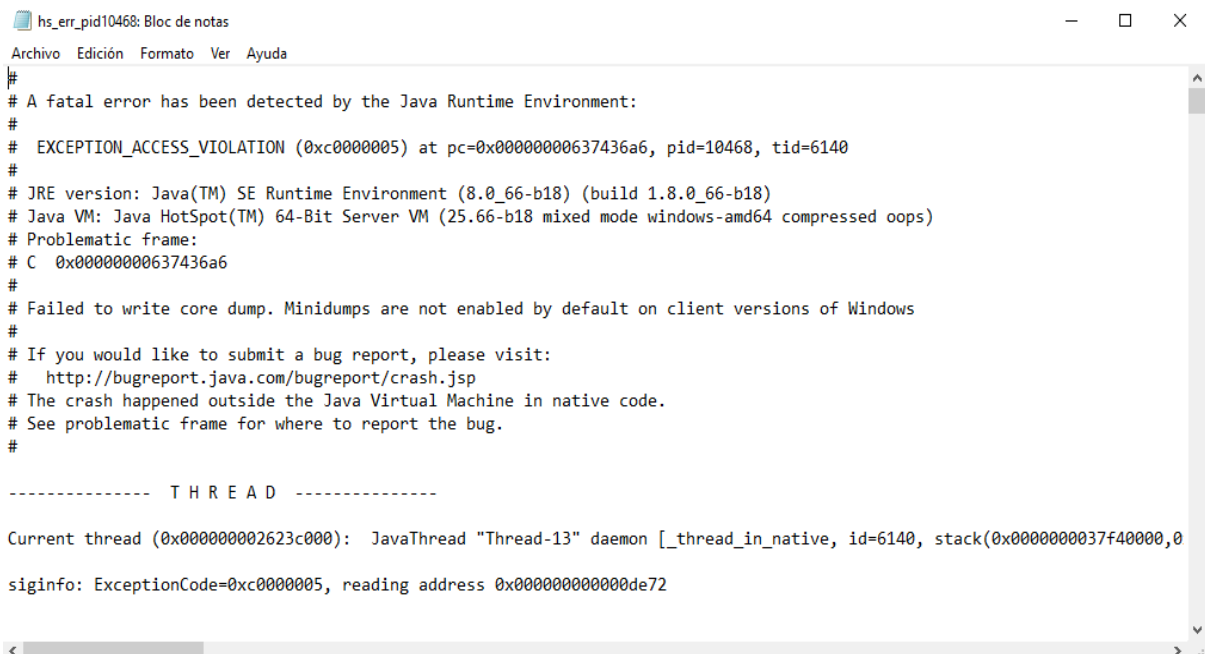
Resumiendo los costes que tiene la utilización de la herramienta desarrollada, al agricultor le supondría un coste de 325€ por cada finca en la que quiera realizar el conteo, teniendo en cuenta que ya ha realizado la compra del drone.

Por lo tanto en relación con otros softwares comerciales, la diferencia de precios es muy elevada. Suponiendo el uso de esta herramienta un coste muy inferior.

17. Posibles mejoras

A la hora de realizar la prueba con fincas de extensiones mucho más grandes existe una limitación a la hora de realizar el stitching con ImageJ.

Cuando se pretende realizar el stitching de más de 7 imágenes con ImageJ se cierra la aplicación y reporta el siguiente informe de error:



```
hs_err_pid10468: Bloc de notas
Archivo Edición Formato Ver Ayuda
#
# A fatal error has been detected by the Java Runtime Environment:
#
# EXCEPTION_ACCESS_VIOLATION (0xc0000005) at pc=0x0000000637436a6, pid=10468, tid=6140
#
# JRE version: Java(TM) SE Runtime Environment (8.0_66-b18) (build 1.8.0_66-b18)
# Java VM: Java HotSpot(TM) 64-Bit Server VM (25.66-b18 mixed mode windows-amd64 compressed oops)
# Problematic frame:
# C 0x0000000637436a6
#
# Failed to write core dump. Minidumps are not enabled by default on client versions of Windows
#
# If you would like to submit a bug report, please visit:
# http://bugreport.java.com/bugreport/crash.jsp
# The crash happened outside the Java Virtual Machine in native code.
# See problematic frame for where to report the bug.
#

----- T H R E A D -----
Current thread (0x000000002623c000):  JavaThread "Thread-13" daemon [_thread_in_native, id=6140, stack(0x0000000037f40000,0
siginfo: ExceptionCode=0xc0000005, reading address 0x00000000000de72
```

Figura 38: Informe de error

ImageJ está limitado por cuestiones de memoria y no es posible aumentar el espacio.

Se han realizado diferentes pruebas con un equipo de 16GB de memoria RAM y como máximo es posible realizar el stitching en un total de 10 imágenes.

También se ha probado reducir la calidad de las imágenes escalándolas por 0,1, pero al realizar el stitching ocurre lo mismo. Por lo tanto el problema no es del equipo utilizado, sino del software ImageJ que tiene estas limitaciones.

El equipo con el que se ha realizado el estudio tiene 8GB y como máximo es posible unir un total de 7 imágenes, por lo tanto la solución para fincas de extensiones más grandes sería utilizar una biblioteca de OpenCV junto con la interfaz de Python.

El código está disponible en el anexo 4.

El algoritmo de costura panorámica consta de cuatro pasos:

- **Paso 1:** Detecta los puntos clave (DoG, Harris, etc.) y extrae los descriptores invariantes locales (SIFT, SURF, etc.) de las dos imágenes de entrada.

- **Paso 2:** Hace coincidir los descriptores entre las dos imágenes.
- **Paso 3:** Usa el algoritmo RANSAC para estimar una matriz de homografía usando los vectores de características coincidentes.
- **Paso 4:** Aplica una transformación-warping usando la matriz de homografía obtenida del paso 3.

La ventaja de utilizar este algoritmo, es que usa funciones parecidas a ImageJ, pero dispone de una mayor libertad de modificación.

El inconveniente sería que habría que realizar un proceso de programación y análisis bastante laborioso que aumentaría exponencialmente el presupuesto.

18. Futuros desarrollos

En este trabajo se ha desarrollado una herramienta que permite de forma automática realizar el conteo de árboles en una finca agrícola, pero sería muy interesante poder aplicar este trabajo como punto de partida para desarrollar futuras aplicaciones que permitan automatizar y mejorar la eficacia en algunos tratamientos y procesos.

Algunos futuros desarrollos podrían ser los siguientes:

- En lugar de tomar las imágenes a 100 metros de altura, reducir la altura hasta aproximadamente los 5 metros para poder capturar con más detalle cada árbol. Con esto lo que se puede conseguir es medir de forma más exacta la frondosidad de cada árbol para ir comparando la frondosidad en de cada árbol según las zonas de la finca y analizar si todos los árboles tienen la misma frondosidad.

En el caso de que no tengan la misma frondosidad, habría que analizar qué áreas de la finca es dónde se producen estos fenómenos y ver si hay alguna correlación entre las frondosidades y la humedad en la tierra.

Este posible desarrollo tendría una aplicación directa en el campo de los drones y el procesamiento de imágenes, y aparte llevaría una parte de sensórica asociada al terreno.



Figura 39: Sensor aplicado en la agricultura

- Una vez que ha sido posible contar los árboles, el siguiente paso sería poder contar los frutos que tiene cada árbol a través de cámaras térmicas, ya que los frutos se encuentran a una temperatura distinta a las hojas, tronco y tierra.

Este sería un desarrollo en el que están interesadas todas empresas hortofrutícolas, ya que les permitiría tener una aproximación de la producción antes de realizar la recolección. Lo que les permite anticiparse y poder adaptarse mejor a la demanda y al precio del mercado.



Figura 40: Drone realizando labores de agricultura de precisión

19. Conclusiones

Lo primero de todo es agradecer a la Universidad Politécnica de Cartagena por prestar los medios necesarios para poder llevar a cabo esta investigación y a la empresa hortofrutícola Belsan por permitir la captura de imágenes en sus fincas.

Con este trabajo se pone de manifiesto el poder que tienen algunas herramientas de licencia pública como ImageJ para poder desarrollar herramientas capaces de realizar procesos que hasta ahora se han ido realizando de forma manual, de forma automática. Además se consigue poner la primera piedra en el camino para que se puedan desarrollar futuras herramientas que puedan ayudar al diagnóstico temprano de cualquier plantación, o que puedan aportar soluciones inteligentes teniendo en cuenta todos los datos disponibles.

Los drones tienen múltiples aplicaciones en la agricultura, y van a ser los principales protagonistas durante los próximos años, ya que van a permitir automatizar muchas tareas que hasta ahora se han ido llevando a cabo por medios humanos.

Este trabajo lo demuestra, ya que permite automatizar el proceso de conteo de árboles en parcelas agrícolas que hasta ahora se ha tenido que realizar manualmente en el caso de no disponer de los planos de plantación. Que en muchos casos no se disponen de ellos.

Se han cumplido con todos los objetivos inicialmente propuestos, como han sido las planificaciones de rutinas de vuelo autónomo, desarrollo de algoritmos capaces de realizar el stitching de las imágenes capturadas y el posterior análisis y las comparativas de resultados según el método de obtención de las imágenes.

El script realizado nos permite disponer de un 93,10% de acierto en el conteo de árboles. Este resultado creemos que es muy aceptable, teniendo en cuenta los medios disponibles, por lo que pensamos que este margen podría llegar a superarse con otras herramientas y medios, pero a priori nos pone de manifiesto lo que podemos llegar a ser capaces de realizar con un dron comercial y un software de código libre.

Por lo tanto esperamos que este trabajo pueda servir como base para futuros trabajos que permitan mejorar procesos que hasta ahora se han venido realizando de forma manual. Pudiendo mejorar la rentabilidad de los procesos y la competitividad en el sector agrícola.

20. Bibliografía

<http://inia.uy/Documentos/P%C3%BAblicos/INIA%20Tacuaremb%C3%B3/Jornada-Tecnica-Forestal%209%20nov%2016/Patricio%20Birriel.pdf>

<https://qampo.es/la-agricultura-de-precision/>

<https://imagej.net/Welcome>

<http://www.unoosa.org/documents/pdf/psa/activities/2008/colombia/presentations/3-1.pdf>

<https://www.efilogy.es/>

https://docs.opencv.org/trunk/d8/d19/tutorial_stitcher.html

<http://agriculturers.com/drones-para-agricultura-beneficios-y-casos-reales/>

https://elpais.com/politica/2015/03/17/actualidad/1426610676_564585.html

http://repositorio.udec.cl/bitstream/handle/11594/407/Dendrometria_Basica.pdf?sequence=1

<http://43jaiio.sadio.org.ar/proceedings/CAI/17.pdf>

<https://www.publico.es/ciencias/agricultura-drones-salvar-olivos.html>

<https://riunet.upv.es/bitstream/handle/10251/86353/MITSIKOSTAS%20-%20Monitorizaci%F3n%20y%20optimizaci%F3n%20de%20tierras%20con%20drones%20y%20fotogrametr%EDa%20a%20E9rea%20para%20apli....pdf?sequence=4>

<https://www.dronair.es/nueva-ley-sobre-el-uso-de-drones-en-espana-2>

<https://imagej.net/Stitch and Align a sequence of grid images Tutorial>

<http://www.aerial-insights.co/blog/normativa-drones-espana/>

<https://www.todrone.com/nueva-legislacion-drones-2018/>

<http://www.efilogy.es/industria40/drones-en-agricultura/>

https://imagej.net/Classic_Watershed

<http://hemeroteca.unad.edu.co/index.php/publicaciones-e-investigacion/article/view/1585/1930>

21. Fuente de las figuras

Figura 1: <https://www.agroproprod.com/noticias/la-agricultura-precision-podria-alimentar-la-humanidad-manera-ecologica/>

Figura 2: <https://360radio.com.co/con-drones-de-fumigacion-colombia-combatira-cultivos-ilicitos/>

Figura 3: <https://www.cimformacion.com/blog/aeronautica/que-es-aesa/>

Figura 4: <http://www.tierraquebrada.com/2013/conceptos-basicos-de-cartografia-la-escala/>

Figura 5: <https://www.creativosonline.org/blog/basico-para-fotografos-clases-de-objetivos.html>

Figura 6: <http://todo-fotografia.com/tecnica/el-factor-de-multiplicacion-de-la-distancia-focal/>

Figura 7: <http://drones.uv.es/aplicaciones-de-drones-a-la-gestion-del-patrimonio/>

Figura 8: https://es.m.wikipedia.org/wiki/Archivo:Tipos_de_LIDAR.jpeg

Figura 9: https://www.researchgate.net/figure/Figura-22-Esquema-de-la-captura-de-datos-LiDAR-desde-un-avion_fig3_277739441

Figura 10: <http://www.aerial-insights.co/blog/6-aplicaciones-para-planificar-el-vuelo-de-tu-dron/>

Figura 11: <http://www.aerial-insights.co/blog/6-aplicaciones-para-planificar-el-vuelo-de-tu-dron/>

Figura 12: <http://www.aerial-insights.co/blog/6-aplicaciones-para-planificar-el-vuelo-de-tu-dron/>

Figura 13: <http://www.aerial-insights.co/blog/6-aplicaciones-para-planificar-el-vuelo-de-tu-dron/>

Figura 14: <http://www.aerial-insights.co/blog/6-aplicaciones-para-planificar-el-vuelo-de-tu-dron/>

Figura 15: <http://www.aerial-insights.co/blog/6-aplicaciones-para-planificar-el-vuelo-de-tu-dron/>

Figura 39: <http://www.dicyt.com/viewItem.php?itemId=1134>

Figura 40: <http://www.pregonagropecuario.com/cat.php?txt=7253>

22. Anexos

Anexo 1:

```
package
inra.ijpb.watershed;

import ij.ImagePlus;
import ij.ImageStack;
import ij.process.ImageProcessor;
/**
 * Several static methods for computing watershed in 2D/3D
 images.
 * @author Ignacio Arganda-Carreras
 */
public class Watershed
{
    /**
     * Compute fast watershed using flooding simulations,
 as described by
     * Soille, Pierre, and Luc M. Vincent. "Determining
 watersheds in
     * digital pictures via flooding simulations."
 Lausanne-DL tentative.
     * International Society for Optics and Photonics,
 1990.
     *
     * @param input original grayscale image (usually a
 gradient image)
     * @param mask binary mask to restrict the regions of
 interest
     * @param connectivity voxel connectivity to define
 neighborhoods
     * @return image of labeled catchment basins with dams
 (labels are 1, 2, ...)
     */
    public static ImagePlus computewatershed(
        ImagePlus input,
        ImagePlus mask,
        int connectivity )
    {
        WatershedTransform3D wt = new
WatershedTransform3D( input, mask, connectivity );

        return wt.apply();
    }
}
```

```

/**
 * Compute fast watershed using flooding simulations,
as described by
 * Soille, Pierre, and Luc M. Vincent. "Determining
watersheds in
 * digital pictures via flooding simulations."
Lausanne-DL tentative.
 * International Society for Optics and Photonics,
1990.
 *
 * @param input original grayscale image (usually a
gradient image)
 * @param mask binary mask to restrict the regions of
interest
 * @param connectivity voxel connectivity to define
neighborhoods
 * @param hMin the minimum value for dynamic
 * @param hMax the maximum value for dynamic
 * @return image of labeled catchment basins with dams
(labels are 1, 2, ...)
 */
public static ImagePlus computeWatershed(
    ImagePlus input,
    ImagePlus mask,
    int connectivity,
    double hMin,
    double hMax )
{
    if( connectivity == 6 || connectivity == 26 )
    {
        WatershedTransform3D wt = new
WatershedTransform3D( input, mask, connectivity );
        return wt.apply( hMin, hMax );
    }
    else if( connectivity == 4 || connectivity == 8 )
    )
    {
        WatershedTransform2D wt =
            new WatershedTransform2D(
input.getProcessor(),
                null != mask
? mask.getProcessor() : null, connectivity );
        final ImageProcessor ip = wt.apply( hMin,
hMax );
        if( null != ip )

```



```

        {
            String title = input.getTitle();
            String ext = "";
            int index = title.lastIndexOf( ".");
        );
            if( index != -1 )
            {
                ext = title.substring(
index );
                title = title.substring( 0,
index );
            }

            final ImagePlus ws = new
ImagePlus( title + "-watershed" + ext, ip );
            ws.setCalibration(
input.getCalibration() );
            return ws;
        }
        else
            return null;
    }
    else
        return null;
}

/**
 * Compute fast watershed using flooding simulations,
as described by
 * Soille, Pierre, and Luc M. Vincent. "Determining
watersheds in
 * digital pictures via flooding simulations."
Lausanne-DL tentative.
 * International Society for Optics and Photonics,
1990.
 *
 * @param input original grayscale image (usually a
gradient image)
 * @param mask binary mask to restrict the regions of
interest
 * @param connectivity voxel connectivity to define
neighborhoods
 * @return image of labeled catchment basins with dams
(labels are 1, 2, ...)
 */
public static ImageStack computeWatershed(

```

```

        ImageStack input,
        ImageStack mask,
        int connectivity )
    {
        final ImagePlus inputIP = new ImagePlus(
"input", input );
        final ImagePlus binaryMaskIP = ( null != mask )
? new ImagePlus( "binary mask", mask ) : null;
        WatershedTransform3D wt = new
WatershedTransform3D( inputIP, binaryMaskIP, connectivity );

        final ImagePlus ws = wt.apply();
        if( null != ws )
            return ws.getImageStack();
        else
            return null;
    }

/**
 * Compute fast watershed using flooding simulations,
as described by
 * Soille, Pierre, and Luc M. Vincent. "Determining
watersheds in
 * digital pictures via flooding simulations."
Lausanne-DL tentative.
 * International Society for Optics and Photonics,
1990.
 *
 * @param input original grayscale image (usually a
gradient image)
 * @param mask binary mask to restrict the regions of
interest
 * @param connectivity pixel connectivity to define
neighborhoods (4 or 8)
 * @return image of labeled catchment basins with dams
(labels are 1, 2, ...)
 */
public static ImageProcessor computeWatershed(
    ImageProcessor input,
    ImageProcessor mask,
    int connectivity )
    {
        WatershedTransform2D wt = new
WatershedTransform2D( input, mask, connectivity );
        return wt.apply();
    }

```

```
/**
 * Compute watershed with markers with an optional
binary mask
 * to restrict the regions of application
 *
 * @param input original grayscale image (usually a
gradient image)
 * @param marker image with labeled markers
 * @param binaryMask binary mask to restrict the
regions of interest
 * @param connectivity voxel connectivity to define
neighborhoods (4 or 8 for 2D, 6 or 26 for 3D)
 * @param getDams select/deselect the calculation of
dams
 * @return image of labeled catchment basins (labels
are 1, 2, ...)
 */
public static ImagePlus computWatershed(
    ImagePlus input,
    ImagePlus marker,
    ImagePlus binaryMask,
    int connectivity,
    boolean getDams )
{
    return computWatershed( input, marker,
binaryMask, connectivity,
                            getDams, true );
}
/**
 * Compute watershed with markers with an optional
binary mask
 * to restrict the regions of application
 *
 * @param input original grayscale image (usually a
gradient image)
 * @param marker image with labeled markers
 * @param binaryMask binary mask to restrict the
regions of interest
 * @param connectivity voxel connectivity to define
neighborhoods (4 or 8 for 2D, 6 or 26 for 3D)
 * @param getDams select/deselect the calculation of
dams
 * @param verbose flag to display messages in the log
window
 * @return image of labeled catchment basins (labels
are 1, 2, ...)
```

```
*/
public static ImagePlus computeWatershed(
    ImagePlus input,
    ImagePlus marker,
    ImagePlus binaryMask,
    int connectivity,
    boolean getDams,
    boolean verbose )
{
    if( connectivity == 6 || connectivity == 26 )
    {
        MarkerControlledWatershedTransform3D wt =
            new
MarkerControlledWatershedTransform3D( input, marker,
                                     binaryMask,
connectivity );
        wt.setVerbose( verbose );
        if( getDams )
            return
wt.applyWithPriorityQueueAndDams();
        else
            return
wt.applyWithPriorityQueue();
    }
    else if( connectivity == 4 || connectivity == 8
)
    {
        MarkerControlledWatershedTransform2D wt =
            new
MarkerControlledWatershedTransform2D(

            input.getProcessor(), marker.getProcessor(),
                                     null !=
binaryMask ? binaryMask.getProcessor() :
                                     null,
connectivity );
        wt.setVerbose( verbose );
        ImageProcessor ip;
        if( getDams )
            ip =
wt.applyWithPriorityQueueAndDams();
        else
            ip = wt.applyWithPriorityQueue();
        if( null != ip )
        {
            String title = input.getTitle();
```

```

        String ext = "";
        int index = title.lastIndexOf( "."
);
        if( index != -1 )
        {
            ext = title.substring(
index );
            title = title.substring( 0,
index );
        }
        final ImagePlus ws = new
ImagePlus( title + "-watershed" + ext, ip );
        ws.setCalibration(
input.getCalibration() );
        return ws;
    }
    else
        return null;
}
else
    return null;
}
/**
 * Compute watershed with markers with an optional
binary mask
 * to restrict the regions of application
 *
 * @param input original grayscale image (usually a
gradient image)
 * @param marker image with labeled markers
 * @param binaryMask binary mask to restrict the
regions of interest
 * @param connectivity voxel connectivity to define
neighborhoods
 * @param getDams select/deselect the calculation of
dams
 * @return image of labeled catchment basins (labels
are 1, 2, ...)
 */
public static ImageStack computeWatershed(
    ImageStack input,
    ImageStack marker,
    ImageStack binaryMask,
    int connectivity,
    boolean getDams )
{

```

```

        return computeWatershed( input, marker,
binaryMask, connectivity,
                                getDams, true );
    }
    /**
     * Compute watershed with markers with an optional
binary mask
     * to restrict the regions of application
     *
     * @param input original grayscale image (usually a
gradient image)
     * @param marker image with labeled markers
     * @param binaryMask binary mask to restrict the
regions of interest
     * @param connectivity voxel connectivity to define
neighborhoods
     * @param getDams select/deselect the calculation of
dams
     * @param verbose flag to display messages in the log
window
     * @return image of labeled catchment basins (labels
are 1, 2, ...)
     */
    public static ImageStack computeWatershed(
        ImageStack input,
        ImageStack marker,
        ImageStack binaryMask,
        int connectivity,
        boolean getDams,
        boolean verbose )
    {

        final ImagePlus inputIP = new ImagePlus(
"input", input );
        final ImagePlus markerIP = new ImagePlus(
"marker", marker );
        final ImagePlus binaryMaskIP = ( null !=
binaryMask ) ?
                                new ImagePlus( "binary mask",
binaryMask ) : null;
        ImagePlus ws = computeWatershed( inputIP,
markerIP, binaryMaskIP,
                                connectivity, getDams, verbose );
        if ( null != ws )
            return ws.getImageStack();
        else

```

```
        return null;
    }
    /**
     * Compute watershed with markers with an optional
binary mask
     * to restrict the regions of application
     *
     * @param input original grayscale image (usually a
gradient image)
     * @param marker image with labeled markers
     * @param binaryMask binary mask to restrict the
regions of interest
     * @param connectivity voxel connectivity to define
neighborhoods
     * @param getDams select/deselect the calculation of
dams
     * @return image of labeled catchment basins (labels
are 1, 2, ...)
     */
    public static ImageProcessor computeWatershed(
        ImageProcessor input,
        ImageProcessor marker,
        ImageProcessor binaryMask,
        int connectivity,
        boolean getDams )
    {

        return computeWatershed( input, marker,
binaryMask, connectivity,
                                getDams, true );
    }
    /**
     * Compute watershed with markers with an optional
binary mask
     * to restrict the regions of application
     *
     * @param input original grayscale image (usually a
gradient image)
     * @param marker image with labeled markers
     * @param binaryMask binary mask to restrict the
regions of interest
     * @param connectivity voxel connectivity to define
neighborhoods
     * @param getDams select/deselect the calculation of
dams
     * @param verbose flag to display log messages
```

```

        * @return image of labeled catchment basins (labels
are 1, 2, ...)
        */
        public static ImageProcessor computeWatershed(
            ImageProcessor input,
            ImageProcessor marker,
            ImageProcessor binaryMask,
            int connectivity,
            boolean getDams,
            boolean verbose )
        {
            MarkerControlledWatershedTransform2D wt =
                new
MarkerControlledWatershedTransform2D( input, marker,
                binaryMask,
connectivity );
            wt.setVerbose( verbose );
            if( getDams )
                return
wt.applyWithPriorityQueueAndDams();
            else
                return wt.applyWithPriorityQueue();
        }
/**
 * Compute watershed with markers
 *
 * @param input original grayscale image (usually a
gradient image)
 * @param marker image with labeled markers
 * @param connectivity voxel connectivity to define
neighborhoods
 * @param getDams select/deselect the calculation of
dams
 * @return image of labeled catchment basins (labels
are 1, 2, ...)
 */
        public static ImagePlus computeWatershed(
            ImagePlus input,
            ImagePlus marker,
            int connectivity,
            boolean getDams )
        {
            MarkerControlledWatershedTransform3D wt = new
MarkerControlledWatershedTransform3D( input, marker, null,
connectivity );
            if( getDams )

```



```
        return
    wt.applyWithPriorityQueueAndDams();
    else
        return wt.applyWithPriorityQueue();
    }

/**
 * Compute watershed with markers
 *
 * @param input original grayscale image (usually a
gradient image)
 * @param marker image with labeled markers
 * @param connectivity voxel connectivity to define
neighborhoods
 * @param getDams select/deselect the calculation of
dams
 * @return image of labeled catchment basins (labels
are 1, 2, ...)
 */
public static ImageStack computeWatershed(
    ImageStack input,
    ImageStack marker,
    int connectivity,
    boolean getDams )
{
    if ( Thread.currentThread().isInterrupted() )

        return null;

    final ImagePlus inputIP = new ImagePlus(
"input", input );
    final ImagePlus markerIP = new ImagePlus(
"marker", marker );

    MarkerControlledWatershedTransform3D wt = new
MarkerControlledWatershedTransform3D( inputIP, markerIP, null,
connectivity );

    ImagePlus ws = null;
    if( getDams )
        ws = wt.applyWithPriorityQueueAndDams();

    else
        ws = wt.applyWithPriorityQueue();
}
```

```
        if( null == ws )
            return null;
        return ws.getImageStack();
    }

    /**
     * Compute watershed with markers
     *
     * @param input original grayscale image (usually a
    gradient image)
     * @param marker image with labeled markers
     * @param connectivity voxel connectivity to define
    neighborhoods
     * @param getDams select/deselect the calculation of
    dams
     * @return image of labeled catchment basins (labels
    are 1, 2, ...)
     */
    public static ImageProcessor computeWatershed(
        ImageProcessor input,
        ImageProcessor marker,
        int connectivity,
        boolean getDams )
    {

        MarkerControlledWatershedTransform2D wt =
            new
        MarkerControlledWatershedTransform2D( input, marker,

                                                null,
        connectivity );
        if( getDams )
            return
        wt.applyWithPriorityQueueAndDams();

        else
            return wt.applyWithPriorityQueue();
    }
}
```

Anexo 2:

```
run("Grid/Collection stitching",
"type=[Grid: column-by-column] order=[Up & Right]
grid_size_x=1 grid_size_y=3 tile_overlap=13 first_file_index_i=1
directory=C:/Users/Jesús/Desktop/script file_names=tile_{ii}.jpg
output_textfile_name=TileConfiguration.txt fusion_method=[Intensity of random input
tile]
regression_threshold=0.30 max/avg_displacement_threshold=2.50
absolute_displacement_threshold=3.50
compute_overlap computation_parameters=[Save memory (but be slower)]
image_output=[Fuse and display]");

saveAs("Jpeg", "C:/Users/Jesús/Desktop/script/Fused.jpg");
close();
open("C:/Users/Jesús/Desktop/script/Fused.jpg");
run("8-bit");
run("Scale...", "x=0.1 y=0.1 width=625 height=955 interpolation=Bilinear average
create");
selectWindow("Fused.jpg");
close();
saveAs("Jpeg", "C:/Users/Jesús/Desktop/script/Fused-1.jpg");
selectWindow("Fused-1.jpg");
setAutoThreshold("Default");
//run("Threshold...");
//setThreshold(0, 107);
setOption("BlackBackground", true);
run("Convert to Mask");
run("Watershed");
run("Analyze Particles...", "size=100-Infinity display clear summarize add");
saveAs("Results", "C:/Users/Jesús/Desktop/script/Summary.csv");
saveAs("Jpeg", "C:/Users/Jesús/Desktop/script/conteo.jpg");
```

Anexo 3:

```
run("Grid/Collection stitching",
"type=[Grid: column-by-column] order=[Up & Right]
grid_size_x=1 grid_size_y=3 tile_overlap=13 first_file_index_i=1
directory=C:/Users/Jesús/Desktop/script file_names=tile_{ii}.jpg
output_textfile_name=TileConfiguration.txt fusion_method=[Intensity of random input
tile]
regression_threshold=0.30 max/avg_displacement_threshold=2.50
absolute_displacement_threshold=3.50
compute_overlap computation_parameters=[Save memory (but be slower)]
image_output=[Fuse and display]");

saveAs("Jpeg", "C:/Users/Jesús/Desktop/script/Fused.jpg");
close();
open("C:/Users/Jesús/Desktop/script/Fused.jpg");
run("8-bit");
run("Scale...", "x=0.1 y=0.1 width=625 height=955 interpolation=Bilinear average
create");
selectWindow("Fused.jpg");
close();
saveAs("Jpeg", "C:/Users/Jesús/Desktop/script/Fused-1.jpg");
selectWindow("Fused-1.jpg");
setAutoThreshold("Default");
//run("Threshold...");
//setThreshold(0, 107);
setOption("BlackBackground", true);
run("Convert to Mask");
run("Watershed");
makePolygon(61,84,178,925,564,879,471,25);
run("Analyze Particles...", "size=100-Infinity display clear summarize add");
saveAs("Results", "C:/Users/Jesús/Desktop/script/Summary.csv");
saveAs("Jpeg", "C:/Users/Jesús/Desktop/script/conteo.jpg");
```

Anexo 4:

```
#include "opencv2/imgcodecs.hpp"
#include "opencv2/highgui.hpp"
#include "opencv2/stitching.hpp"
#include <iostream>
using namespace std;
using namespace cv;
bool try_use_gpu = false;
bool divide_images = false;
Stitcher::Mode mode = Stitcher::PANORAMA;
vector<Mat> imgs;
string result_name = "result.jpg";
void printUsage(char** argv);
int parseCmdArgs(int argc, char** argv);
int main(int argc, char* argv[])
{
    int retval = parseCmdArgs(argc, argv);
    if (retval) return EXIT_FAILURE;
    Mat pano;
    Ptr<Stitcher> stitcher = Stitcher::create(mode, try_use_gpu);
    Stitcher::Status status = stitcher->stitch(imgs, pano);
    if (status != Stitcher::OK)
    {
        cout << "Can't stitch images, error code = " << int(status) << endl;
        return EXIT_FAILURE;
    }
    imwrite(result_name, pano);
```

```
cout << "stitching completed successfully\n" << result_name << "
    saved!";

return EXIT_SUCCESS;

}

void printUsage(char** argv)
{
cout <<
"Images stitcher.\n\n" << "Usage :\n" << argv[0] <<" [Flags] img1 img2
    [...imgN]\n\n"
"Flags:\n"
" --d3\n"
" internally creates three chunks of each image to increase stitching
    success\n"
" --try_use_gpu (yes|no)\n"
" Try to use GPU. The default value is 'no'. All default values\n"
" are for CPU mode.\n"
" --mode (panorama|scans)\n"
" Determines configuration of stitcher. The default is 'panorama',\n"
" mode suitable for creating photo panoramas. Option 'scans' is
    suitable\n"
" for stitching materials under affine transformation, such as
    scans.\n"
" --output <result_img>\n"
" The default is 'result.jpg'.\n\n"
"Example usage :\n" << argv[0] << " --d3 --try_use_gpu yes --mode
    scans img1.jpg img2.jpg\n";
}

int parseCmdArgs(int argc, char** argv)
{
if (argc == 1)
{
printUsage(argv);
return EXIT_FAILURE;
}
```

```
}  
  
for (int i = 1; i < argc; ++i)  
{  
    if (string(argv[i]) == "--help" || string(argv[i]) == "/?")  
    {  
        printUsage(argv);  
        return EXIT_FAILURE;  
    }  
    else if (string(argv[i]) == "--try_use_gpu")  
    {  
        if (string(argv[i + 1]) == "no")  
            try_use_gpu = false;  
        else if (string(argv[i + 1]) == "yes")  
            try_use_gpu = true;  
        else  
        {  
            cout << "Bad --try_use_gpu flag value\n";  
            return EXIT_FAILURE;  
        }  
        i++;  
    }  
    else if (string(argv[i]) == "--d3")  
    {  
        divide_images = true;  
    }  
    else if (string(argv[i]) == "--output")  
    {  
        result_name = argv[i + 1];  
        i++;  
    }  
}
```

```
else if (string(argv[i]) == "--mode")
{
    if (string(argv[i + 1]) == "panorama")
        mode = Stitcher::PANORAMA;
    else if (string(argv[i + 1]) == "scans")
        mode = Stitcher::SCANS;
    else
    {
        cout << "Bad --mode flag value\n";
        return EXIT_FAILURE;
    }
    i++;
}
else
{
    Mat img = imread(argv[i]);
    if (img.empty())
    {
        cout << "Can't read image '" << argv[i] << "'\n";
        return EXIT_FAILURE;
    }
    if (divide_images)
    {
        Rect rect(0, 0, img.cols / 2, img.rows);
        imgs.push_back(img(rect).clone());
        rect.x = img.cols / 3;
        imgs.push_back(img(rect).clone());
        rect.x = img.cols / 2;
        imgs.push_back(img(rect).clone());
    }
}
```



```
else  
  
    imgs.push_back(img);  
}  
}  
  
return EXIT_SUCCESS;  
}
```