Universidad Nacional de Quilmes

Trabajo Final : Ingeniería en Automatización y Control Industrial

Técnicas de Visión Artificial aplicadas a imágenes de campos experimentales adquiridas desde un drone

Alumnos:
Braian Soullier
Nicolás Cuedo

Directores:
Damián Oliva
Ulises Bussi

1. Resumen

Los ingenieros agrónomos utilizan campos experimentales divididos en parcelas para estudiar el crecimiento de cultivos bajo distintos tratamientos de interés biológico en cada parcela. Para cuantificar el desarrollo de estos experimentos, deben desplazarse asiduamente por el campo experimental tomando mediciones manuales relacionadas con el crecimiento del cultivo.

Este proyecto se propone aumentar el grado de automatización del proceso de adquisición y análisis de datos en estos experimentos, facilitando el proceso de medición tedioso antes mencionado. Para esto, se propone utilizar un *drone* que vuele sobre el campo experimental adquiriendo imágenes del mismo durante los distintos estadios del crecimiento. Se estudian los siguientes problemas:

- Corrección de distorsiones introducidas por la cámara del *drone*
- Determinación de la altura y velocidad del vuelo para lograr imágenes de calidad
- La detección automática de plantas y parcelas
- La georeferenciación de las imágenes adquiridas durante distintos días con el objetivo de tener un registro detallado del crecimiento.

Índice

Índice de figuras

Índice de cuadros

2. Introducción

En la década de los setentas, el sector agropecuario comenzó a experimentar una nueva forma de realizar su labor a través de los estudios de la automatización de equipos agrícolas. Dos décadas más adelante, con la aparición de sistema de posicionamiento global (GPS), se generó un gran avance en este sector, dado que se vió la posibilidad de realizar elementos inteligentes, instalando el concepto de poder localizar a cada parcela del campo utilizando las nuevas tecnologías de geolocalización, e intentar optimizar ciertas variables agrícolas, como la aplicación de insumos. Gracias a esta aparición, se aprovechó el avance en la precisión de localización para reducir el consumo de insumos en los cultivos, impactando no solo sobre los costos financieros, sino que además se redujo ampliamente el impacto ambiental. A partir de esto, se instaló en la sociedad el concepto de *Agricultura de Precisión*, que hoy en día se define como el conjunto de técnicas que permiten mejorar la gestión de las parcelas agrícolas y toma de decisiones basándose en la tecnología de la información que integran datos de múltiples fuentes como sensores, *drones*, radares, entre otros.

La motivación de este proyecto nace a partir de la posibilidad de ampliar las herramientas que están al alcance del operador agropecuario para realizar investigaciones sobre los campos de cultivo. En ciertas circunstancias, el operador puede tomar algunas decisiones con el tratamiento de las parcelas mediante el estudio de la evolución de los campos de cultivo en las diferentes etapas del año, y además, mediante la localización e identificación geográfica de las parcelas de un campo. Esto permite reducir el tiempo de investigación de los campos y actuar de manera rápida y efectiva sobre ciertas parcelas que pueden requerir alguna atención química o agrícola.

Esto nos impulsa a proponer el objetivo general de este proyecto que se basa en desarrollar un sistema para la detección y georeferenciación de las diferentes parcelas de cultivo a partir de imágenes adquiridas desde una plataforma móvil aérea que vuela con ciertos parámetros (obtenidos de una serie de estudios previos) sobre un campo experimental. Tanto las coordenadas de posicionamiento, como los parámetros de orientación de la cámara se combinarán con los videos adquiridos para obtener un registro detallado de cada parcela y del crecimiento y evolución de la misma.

A partir del objetivo general propuesto en el párrafo anterior, se proponen los siguientes objetivos particulares:

Objetivo 1: Aprendizaje de la operación de un cuadrotor DJI Mavic Pro 2. Obtención de videos en campos experimentales.

La carrera Ingeniería de Automatización y Control Industrial de la Universidad Nacional de Quilmes adquirió un cuadrotor comercial de la marca DJI modelo Mavic Pro para la realización de este proyecto. El mismo viene equipado con una cámara de 20 Mega-píxeles usando tecnología de la marca Hasselblad, sensores de movimiento traseros y laterales,

sensor infrarrojo superior, diferentes modos de grabación a varias velocidades, sensores con IMU y GPS, que permiten almacenar la trayectoria del vuelo que realiza el *drone*, y una gran estabilidad(1). El objetivo de esta primer parte es capturar videos experimentales utilizando las precondiciones de vuelo determinadas a través de ciertos estudios previos realizados. Con esto, se evita que los videos capturados sean de una resolución baja, y obtener buenos resultados a la hora de realizar los estudios propuestos a lo largo del proyecto.

Objetivo 2: Calibración de la cámara y corrección de distorsiones.

Una vez tomadas las imágenes con los parámetros óptimos de vuelo, se realizará la calibración de cámara utilizando un tablero de ajedrez, por el cual se obtendrán los parámetros intrínsecos y los mismos se utilizarán para corregir la distorsión. Además, se verificará mediante ciertos estudios que los resultados sean acertados.

Objetivo 3: Detección automática de plantas y parcelas.

Combinando los videos capturados por el *drone* en los campos experimentales y mediante la utilización de algunos algoritmos de visión artificial, se segmentarán las diferentes secciones del campo de cultivo para identificar y etiquetar cada una de las parcelas localizadas.

Objetivo 4: Geo-referenciación de parcelas en campos experimentales y control de crecimiento.

Utilizando los videos capturados y los archivos *logfiles*, generados por el *drone* para dejar registro todos los parámetros de los vuelos realizados, se localizarán las parcelas del campo experimental a través de sus coordenadas geográficas, verificando los resultados con *Google Maps* (5). Se realizará un algoritmo donde el operador seleccione una parcela y se permita visualizar la evolución de la misma en el transcurso del tiempo y brindar las herramientas necesarias para realizar un análisis del control de crecimiento.

3. Calibración de cámara y corrección de distorsión

El proceso de calibración de una cámara es un paso de suma importancia, ya que permite obtener medidas de una escena a partir de varias imágenes de la misma. La exactitud de la calibración determinará la precisión de las medidas que se realicen a partir de las imágenes. Por este motivo, es imprescindible la calibración de la cámara y la elección del método de calibración.

El modelo físico de la cámara que se utiliza en este caso para realizar la calibración de la cámara del *drone* es el modelo *pinhole*. Este describe la relación matemática entre las coordenadas X, Y, Z de un punto 3D en el mundo y su proyección u, v en el plano de la imagen.

$$\begin{vmatrix} x \\ y \\ z \end{vmatrix} = R \begin{vmatrix} x \\ y \\ z \end{vmatrix} + t \tag{1}$$

$$\begin{aligned} x' &= x/z \\ y' &= y/z \\ u &= f_x * x' + c_x \\ v &= f_y * y' + c_y \end{aligned} \tag{2}$$

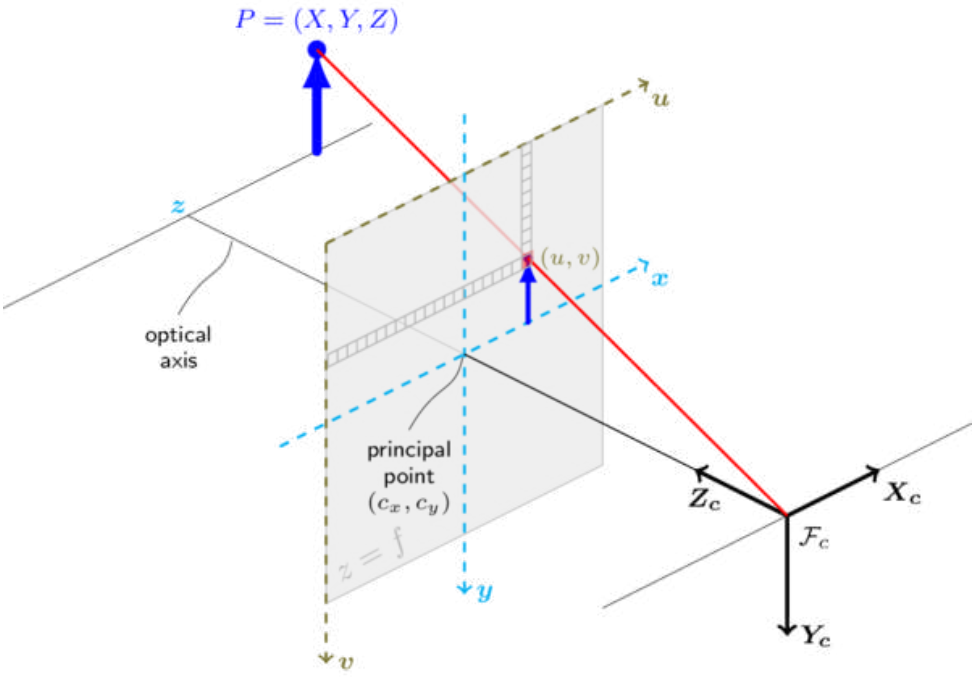


Figura 1: Modelo *pinhole*

Este modelo tiene parámetros intrínsecos y extrínsecos. Los parámetros intrínsecos son aquellos que describen el funcionamiento de la cámara, como la distancia focal, el punto principal y el centro óptico, que son específicos de una cámara. En otras palabras, en una cámara digital, los parámetros internos (o intrínsecos) definen las coordenadas en píxeles de un punto de una imagen virtual con respecto a las coordenadas en el cuadro de referencia de la cámara, siempre con el objetivo de conocer la distancia focal y los demás parámetros. Los parámetros extrínsecos son aquellos que definen la posición y la orientación del cuadro de referencia de la cámara con respecto al mundo real, es decir, dan la orientación externa de la cámara.

Las cámaras introducen distorsión en las imágenes de dos tipos: *radial* y *tangencial*. La distorsión radial se hace más grande cuanto más lejos esta del centro de la imagen y, por ejemplo, se puede observar en una imagen cuando se ven las líneas rectas como líneas curvas. Matemáticamente, se puede representar como:

$$x_{dist} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

$$y_{dist} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

En cambio, la distorsión tangencial se produce porque la lente de la cámara no está alineada perfectamente de manera paralela al plano de la imagen. Un ejemplo visual de esta distorsión podría hallarse cuando se observan objetos que presentan una distancia menor entre ellos que en el mundo real. El modelo matemático es el siguiente:

$$x_{dist} = x + [2p_1 xy + p_2(r^2 + 2x^2)]$$

$$y_{dist} = y + [2p_2 xy + p_1(r^2 + 2y^2)]$$

Es decir, se necesitan encontrar 5 parámetros: k_1, k_2, p_1, p_2 y k_3.
Para la corrección de la distorsión de las imágenes, es necesario realizar el proceso de calibración de la cámara. Como en nuestro caso contamos con una cámara móvil (incrustada en la plataforma móvil aérea) donde la posición y orientación son variables temporales, solo será necesario proceder con la obtención de los parámetros intrínsecos para realizar la calibración.

Dentro de los parámetros intrínsecos (propios de la cámara), se encuentra la distancia focal (f_x, f_y) y el centro óptico (c_x, c_y). Ambos son necesarios para la creación de la matriz de la cámara, necesaria para eliminar la distorsión de las imágenes. Esta matriz es única para una cámara específica, y una vez calculada, se puede utilizar para corregir todas las imágenes que se capturen solo con la misma cámara. Dicha matriz de tamaño tres por tres se expresa como:

$$\begin{bmatrix} \mathrm{f}_x & 0 & \mathrm{c}_x \\ 0 & \mathrm{f}_y & \mathrm{c}_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

3.1. Procedimiento de calibración

Para realizar la calibración de la cámara, se utiliza como patrón un tablero de ajedrez, como se observa en la Fig. 2, que contiene una cantidad de 10 cuadros horizontales y 7

cuadros verticales con un tamaño de 5 centímetros cada uno intercalando color de relleno entre blanco y negro.

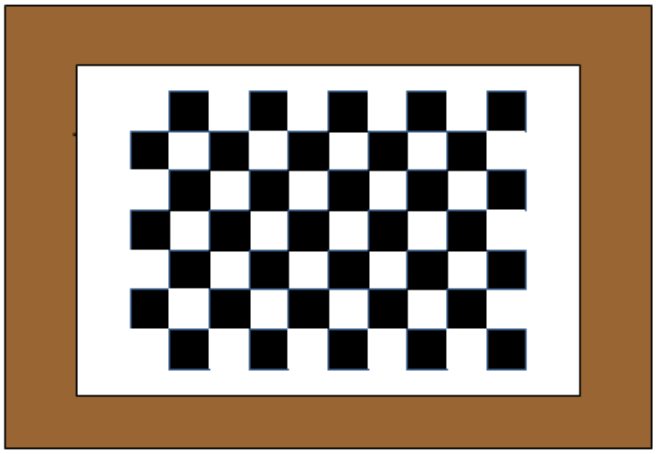


Figura 2: Tablero de calibración

Los datos de entrada necesarios para la calibración son el conjunto de puntos 3D (mundo real) y sus correspondientes puntos en la imagen. Los puntos 2D (puntos de la imagen) se pueden hallar de manera sencilla ya que se toman los puntos donde se unen dos cuadros negros.

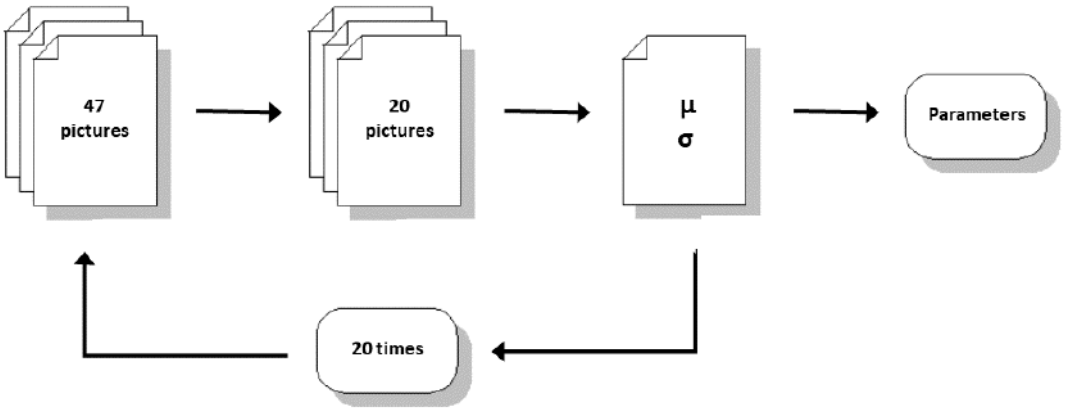


Figura 3: *Flowchart* de calibración

Para obtener las coordenadas de los puntos en el mundo real, en principio, para simplificar el procedimiento, se realiza la suposición de que el tablero siempre se mantuvo dentro del plano X e Y, por lo que la coordenada Z de los puntos se puede despreciar. Dicha esta suposición, se toman 47 imágenes con la cámara en una posición fija y el tablero de ajedrez con una posición y orientación diferente en todas las imágenes. Luego, se pasan los puntos (0,0),(0,1),(0,2)... para x e y, que denotan la ubicación de los mismos. En este caso, los resultados que se obtienen estarán en la escala del tamaño del cuadrado

del tablero de ajedrez. Pero como se conocen las dimensiones de los cuadros del tablero, se pueden pasar los valores reales. Para este tablero en particular, la longitud de cada lado de un cuadro es de 5cm, por lo tanto se pueden pasar los valores (0,0),(0,5),(0,10)... y los resultados quedan expresados en centímetros. Todo este procedimiento se puede leer detalladamente en la sección de calibración de la página de la librería *OpenCV* (16). Dicha esta aclaración, se procede a obtener las ~~cuarenta y siete~~ imágenes, como las que se muestran en la Fig. 4 , para comenzar con la calibración de la cámara.

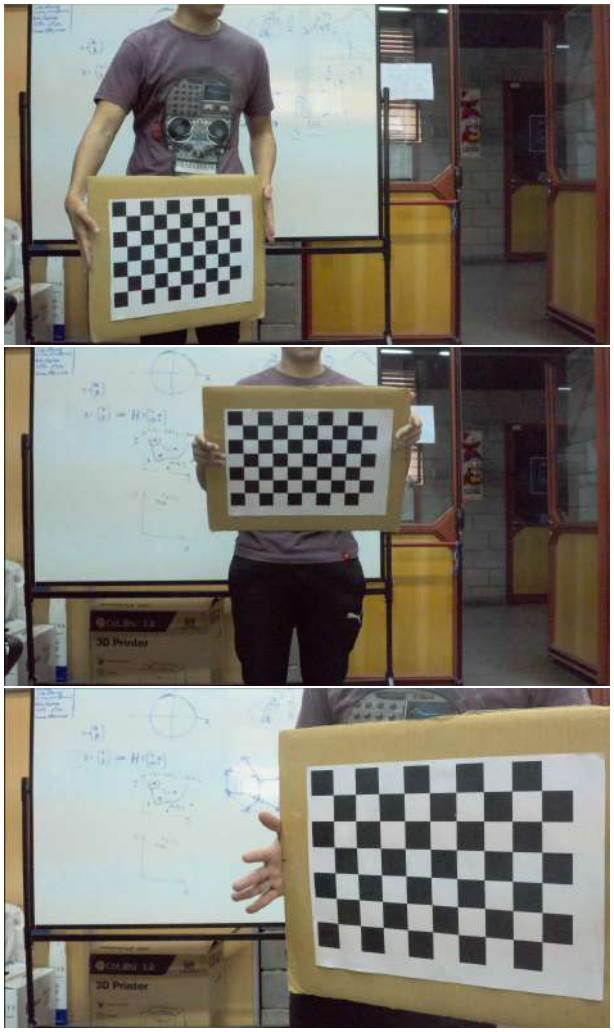


Figura 4: Tres imágenes de calibración

Luego, ~~se~~ escogen veinte imágenes de manera aleatoria dentro de las ~~cuarenta~~ y seis, se realiza el proceso de calibración. Por último, se obtienen los datos de la media y desviación estándar de dicha calibración. Este algoritmo se repite veinte veces y, para alcanzar los resultados finales, se realiza un promedio de los veinte valores de media y desviación estándar. Finalmente, estos parámetros son los que se utilizan para corregir la distorsión de las diferentes imágenes que obtiene la cámara.

3.2. Resultados de calibración

En una primera instancia se tomaron dos conjuntos de fotografías del tablero, en donde la cámara se configuró con una velocidad de obturación de $\frac{1}{60}s$, mientras que el segundo conjunto se configuró este parámetro con un valor de $\frac{1}{240}s$. Esta modificación entre ambos *set* de imágenes no debería alterar los parámetros que se obtienen al realizar la calibración, sin embargo, los resultados que se obtuvieron fueron muy diferentes.

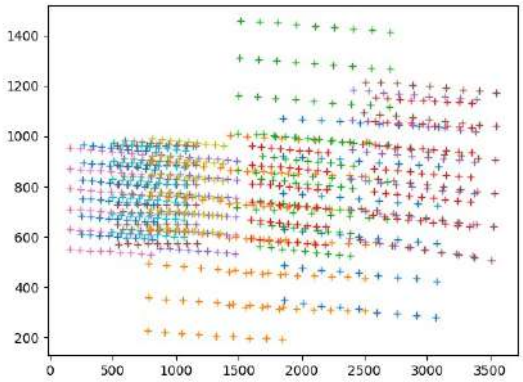
μ	**1/60s**	**1/240s**
f_x	5372.5	4913.8
f_y	5377.2	4905.1
c_x	1927.9	1965.9
c_y	1257.7	1238.8

Cuadro 1: Media(μ) para $\frac{1}{60}s$ y $\frac{1}{240}s$

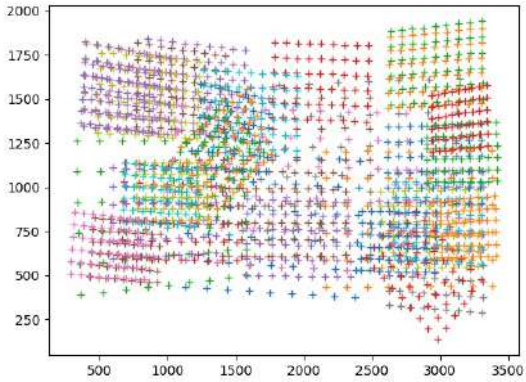
σ	**1/60s**	**1/240s**
f_x	1375.2	642.0
f_y	1376.6	640.3
c_x	339.3	99.9
c_y	220.9	112.8

Cuadro 2: Desviación estándar(σ) para $\frac{1}{60}s$ y $\frac{1}{240}s$

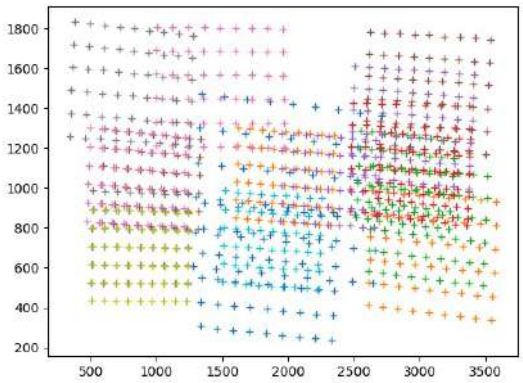
Esta diferencia es un inconveniente debido a que al calibrar la cámara con estas imágenes, nos conduce a una calibración de calidad baja. Para solucionar este inconveniente, se realizaron dos nuevos conjuntos de fotografías, en las cuales se ubicó al patrón de calibración en todas las posiciones posible de modo que cubra todo la apertura de la cámara. Las siguientes imágenes muestran la distribución del tablero con ambas velocidades de obturación, mostrando efectivamente la mejora en la distribución del tablero.



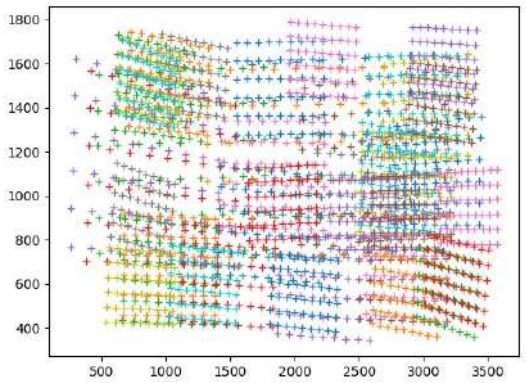
(a) Patrón mal distribuido - $\frac{1}{60}s$



(b) Patrón bien distribuido - $\frac{1}{60}s$



(c) Patrón mal distribuido - $\frac{1}{240}s$



(d) Patrón bien distribuido - $\frac{1}{240}s$

Figura 5: Distribución de patrón de calibración

Con este nuevo *dataset* de imágenes, se realizó el mismo procedimiento y los resultados fueron mas acertados, permitiendo avanzar con el proceso de corrección de distorsión.

μ	**1/60s**	**1/240s**
f_x	4515.3	4545.5
f_y	4515.2	4542.9
c_x	1953.4	1976.6
c_y	1203.0	1070.9

Cuadro 3: Media(μ) para $\frac{1}{60}s$ y $\frac{1}{240}s$

σ	1/60s	1/240s
f_x	160.3	79.3
f_y	159.0	75.1
c_x	64.1	111.6
c_y	142.4	40.4

Cuadro 4: Desviación estándar(σ) para $\frac{1}{60}s$ y $\frac{1}{240}s$

3.3. Corrección de distorsión

Como se mencionó al inicio de este capítulo, los parámetros obtenidos permiten corregir aquellas figuras que en el mundo real presentan una geometría recta, y en la imagen, se visualizan como una curva. La manera mas sencilla y utilizada en este caso fue haciendo uso de la función de des-distorsión (16) de OpenCV (7), que requiere de los siguientes parámetros:

Input:

- **src:** imagen de entrada distorsionada
- **cameraMatrix:** matriz de la cámara (eq. 3)
- **distCoeffs:** vector con coeficientes de distorsión (k_1, k_2, p_1, p_2 y k_3)

Output:

- **dst:** imagen de salida con corrección de distorsión

La siguiente imagen fue tomada sobre unos campos de cultivo con el *drone* que se utilizo en este proyecto , y demuestra el resultado de una correcta corrección de distorsión. Si se observa en la imagen superior ~~de la Fig. 6~~ y se focaliza en la calle horizontal que bordea los campos, se aprecia como la calle presenta cierta curvatura, mientras que en la imagen inferior de la ~~Fig. 6,~~ con uso de la función *undistort* (16) del paquete de *OpenCV* (7), la curvatura de la calle desaparece.

(a) imagen distorsionada

(b) imagen corregida

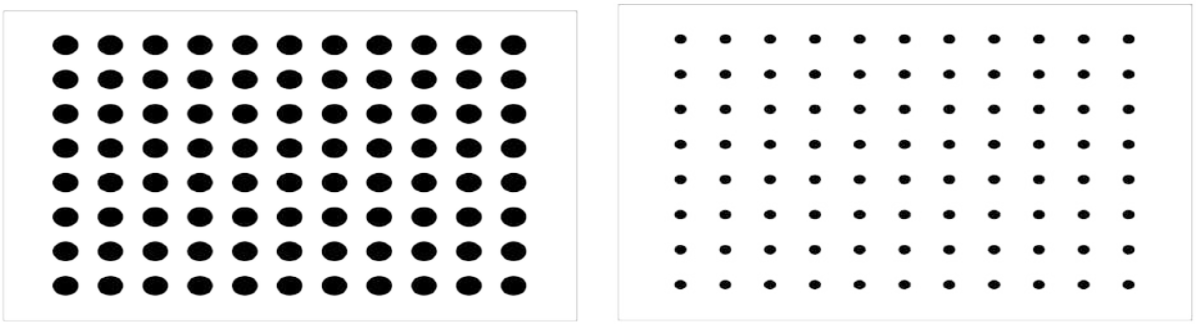
Figura 6: Corrección de distorsión

Si bien este método sencillo que corrige la distorsión en todas las imágenes capturadas por el *drone* permite realizar un futuro procesamientos sobre las mismas (como obtener cierta información con procesamientos digitales), cabe destacar que existe una mínima perdida de información sobre los bordes de la imágenes, que a nuestro labor, impacta de manera poco significativa.

4. Condiciones de Vuelo

Uno de los principales objetivos de este proyecto es poder adquirir imágenes de alta calidad para tratar de no perder ningún detalle físico del campo experimental para su posterior estudio. Esto implica realizar un estudio de las condiciones de vuelo del cuadrotor *Mavic Pro 2*, teniendo en cuenta los parámetros variables de su cámara integrada, la posición GPS, la altura y la velocidad con la cual se realiza el vuelo. Es importante tener en cuenta que el *drone* genera por cada vuelo un *dataset* llamado *DJIFlightRecord*. Con este archivo, se puede ingresar a la página del fabricante (23) y acceder a un archivo de formato CSV (*Comma-Separated Values*), en el cual se puede tener acceso a la información del vuelo realizado. Gracias a estos archivos, podemos saber con precisión las variables expuestas para el estudio de las condiciones de vuelo.

La meta de este estudio es intentar medir el "borroneo" de las imágenes adquiridas. Cuantificar el "borroneo" en una imagen no es una tarea sencilla , ya que para esto hay que tomar ciertos criterios de análisis. En este trabajo se realizó una serie de vuelos variando la altura, la velocidad y los parámetros de la cámara. El propósito de estos vuelos es poder captar imágenes del *drone* volando sobre dos patrones de círculos, del cual sabemos con exactitud el diámetro de cada circulo y la cantidad de los mismos, y poder realizar un análisis de como se ven para las diferentes condiciones de vuelo. En la *figura 7* se muestran los patrones utilizados para los experimentos.



(a) Patrón de ∅ 1cm (b) Patrón de ∅ 0.5cm

Figura 7: Patrones con círculos de diferentes diámetros

Estos patrones se situaron a un distancia aproximada de 4.30 metros, como se muestra en la *figura 8* de representación.

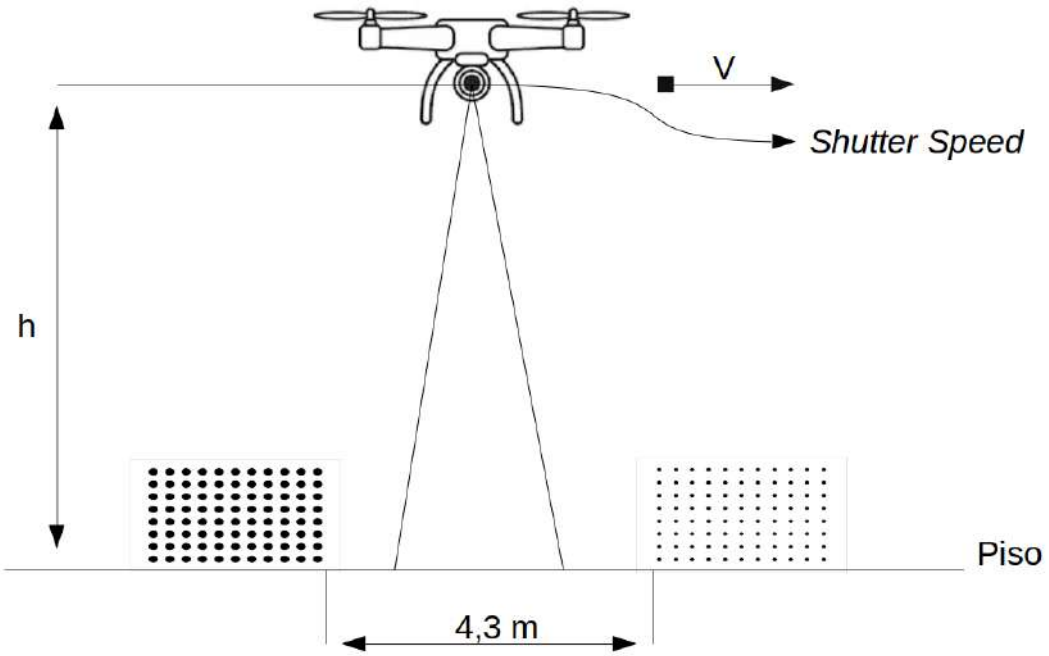


Figura 8: Disposición de los patrones en los videos

Se realizaron veintisiete vuelos variando entre tres alturas,tres velocidades y tres *shutter speed* (termino también conocido como velocidad de obturación) de la cámara:

- Alturas: 4 metros, 6 metros y 8 metros
- Velocidades: $1\frac{m}{s}$, $4\frac{m}{s}$ y $5\frac{m}{s}$
- *Shutter Speed*: $\frac{1}{60}s$, $\frac{1}{120}s$ y $\frac{1}{240}s$

Una vez obtenidos los vídeos experimentales, se procedió a su procesamiento utilizando *Python* (6) y sus librerías. Las librerías utilizadas son las siguientes:

- *OpenCV* (7)
- *Maplotlib* (8)
- *Numpy* (9)
- *OS* (10)
- *Pandas* (11)
- *Glob* (12)
- *Time* (13)

El análisis de dicho procesamiento se basó principalmente en algoritmos y teoría de visión artificial, por este motivo la librería de *OpenCV* (7) es la más utilizada por este trabajo. La misma cuenta con una gran cantidad de algoritmos sofisticados y clásicos de visión artificial.

4.1. Algoritmo de detección de patrones en videos.

El primer paso realizado para poder cuantificar el "borroneo", es encontrar las imágenes de los patrones utilizados dentro de los vídeos obtenidos. Se desarrolla un algoritmo utilizando las librerías mencionadas, que busca los *frames* en el video donde aparecen los patrones, y de estos *frames* guarda únicamente la parte de la imagen que corresponde a los patrones. De esta manera se restringe el problema al análisis en el patrón conocido. El programa analiza los archivos *DJIFlightRecord.CSV* para encontrar la altura, la velocidad y el *shutter speed* de cada video y poder almacenar los patrones encontrados en una carpeta discriminada por dichos parámetros. Todo el procesamiento de los *dataframes* de los archivos *"DJIFlightRecord.CSV"* se realiza con la librería *Pandas* (11) y sera explicado en la **sección 5**

Para comenzar con la búsqueda de los patrones, el programa hace uso de la librería *Glob* (12), que carga todos los vídeos generados para procesar. En un bucle *for* se recorren todo los videos, y según los parámetros de ese video se establece un umbral de binarización y de área que serán utilizados para encontrar los patrones. Con la librería de *OpenCV* (7), se sitúa el vídeo en un tiempo donde la cámara del *drone* apunta hacia abajo y se analizan los *frames* hasta que la cámara se levante. Esto se debe a que se uso un modo de vuelo denominado *Tap to fly*, donde la cámara del *drone* debe estar apuntando hacia el horizonte para poder tocar un punto de visualización de esta y el *drone* pueda volar manteniendo una altura y una velocidad configurada con anterioridad. Como los patrones están ubicados en el suelo, una vez que comienza el vuelo, la cámara baja 90 grados para poder capturar los *frames* que nos interesa.

Figura 9: Imagen de un *frame* del video donde aparece el patrón.

Durante esos *frames* aparecerán en el video las imágenes de los patrones que se deben guardar. Para esto, el algoritmo realiza una binarización utilizando el umbral correspondiente y luego se realizan operaciones morfológicas, tales como *erosiones* y *dilataciones*, que logran limpiar la imagen.

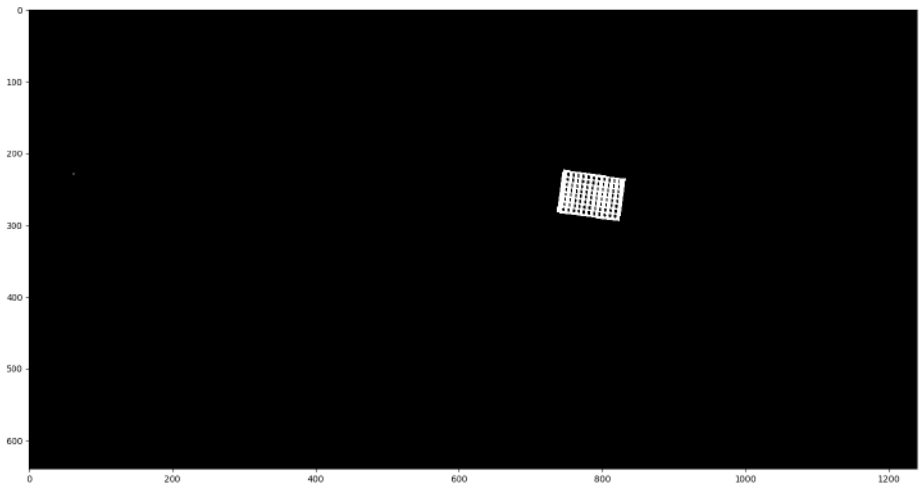


Figura 10: Imagen luego de aplicar una binarización y un conjunto de operaciones morfológicas

Continuando a partir de la imagen limpia, se encuentra el rectángulo filtrando por áreas.

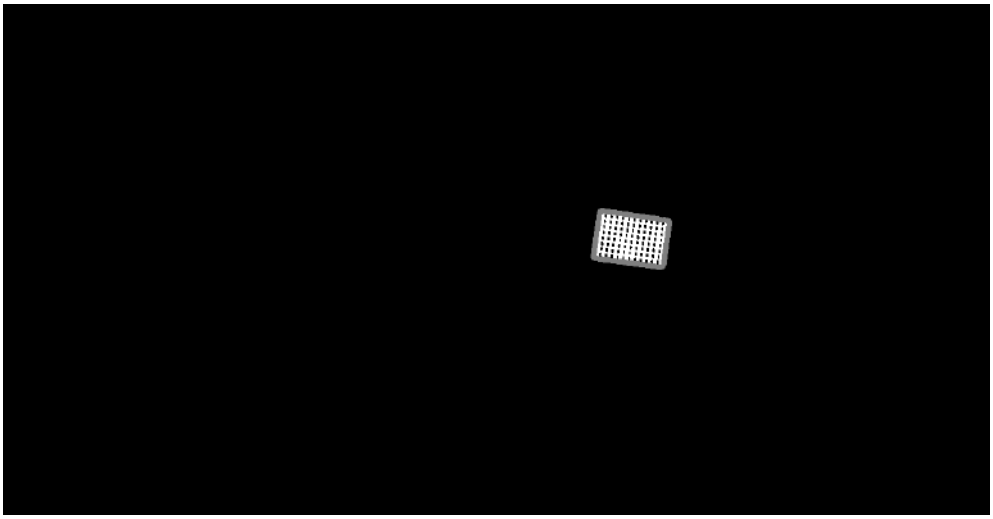


Figura 11: Área del patrón.

Con este umbral de área se asegura de tener los píxeles de las imágenes que corresponden a los patrones en el vídeo, y los guarda en una carpeta como se menciona anteriormente, para su posterior análisis.

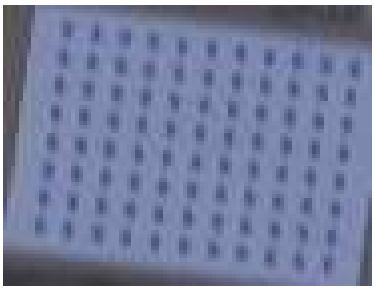


Figura 12: Patrón detectado luego de aplicar el algoritmo descripto

Este algoritmo se repite en todos los *frames* que aparece el patrón en el video y para cada uno de los videos generados. Esto quiere decir que en la carpeta de almacenamiento, se encontrarán una cierta cantidad de imágenes del patrón que va a depender, tanto de la altura como de la velocidad con la que se haya realizado el vuelo.

4.2. Análisis de borroneo

Luego de implementar el algoritmo de detección de patrones, se prosigue en realizar el análisis del *blur* para estas imágenes. Para dicho análisis, se utilizaron 3 criterios:

- **Modelo geométrico de *blur***
- **Ajuste de elipse**
- **Varianza del Laplaciano**

A continuación se explica como se implementaron dichos criterios.

4.2.1. Modelo geométrico de *blur*

El enfoque que se realiza con este modelo, es poder configurar la cámara y los parámetros físicos de vuelo bajo ciertas suposiciones del modelo. Se busca poder identificar cuantos píxeles necesito para poder tener una resolución de 1 cm de manera nítida y sin "borroneo", es decir, en la imagen cuantos píxeles representan 1cm. El *gráfico 13* muestra los parámetros de vuelo que se analizarán.

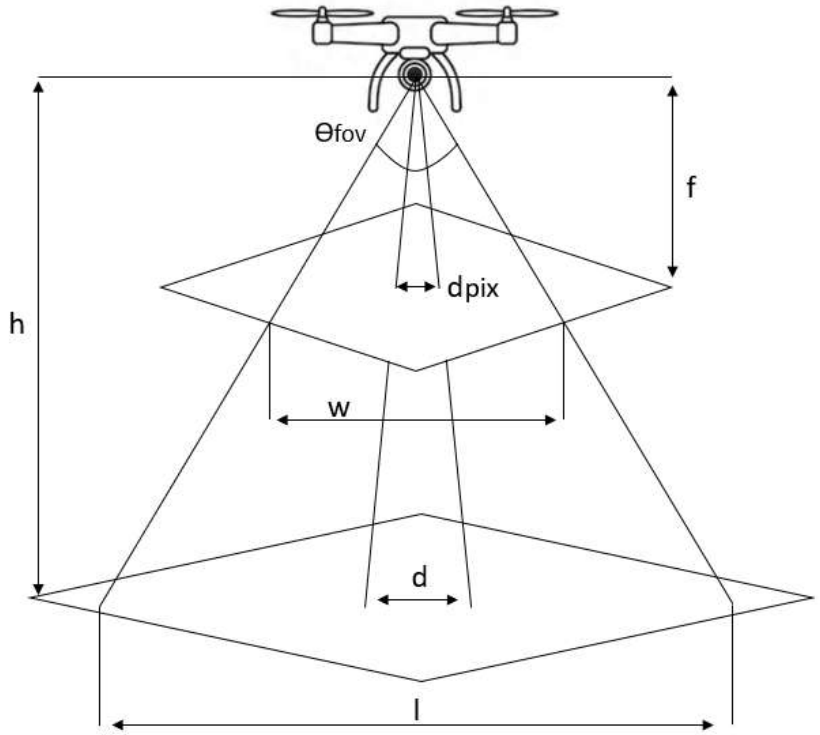


Figura 13: Esquema de parámetros físicos del vuelo

θ_{fov}: Apertura total de la cámara (Ángulo *field of view*)
f: Distancia focal [pixel]
h: Altura [m]
d_{pix}: tamaño de objeto [pixel]
d: tamaño de objeto [m]
l: Ancho de visión del *drone* [m]
w: Ancho de visión plano imagen [pixel]

Con las siguientes ecuaciones se relaciona los parámetros en el mundo real con los parámetros en el plano imagen. Para hacer el análisis mas sencillo, se divide la *figura 13* a la mitad. Teniendo en cuenta esto y relacionando los parámetros bajo funciones trigonométricas, se obtiene:

$$\begin{aligned} \frac{\theta_{fou}}{2} = \frac{\frac{w}{2}}{f} &= \frac{\frac{l}{2}}{h} \\ \frac{l}{2} &= h\frac{\frac{w}{2}}{f} \\ \boxed{l = \frac{hw}{f}} & \end{aligned} \tag{4}$$

Bajo esta relación podemos plantear cual es la altura correcta de vuelo sabiendo : el

w en píxeles , el l en centímetros o metros reales y la distancia focal medida en píxeles encontrada cuando se realizó la calibración de la cámara. Siguiendo con este razonamiento:

$$\frac{d_{pix}}{d} = \frac{w}{l} = \frac{w}{\frac{hw}{f}}$$
$$\boxed{\frac{d_{pix}}{f} = \frac{d}{h}} \tag{5}$$

Acomodando la *ecuación 5* y reemplazando con la *ecuación 4*:

$$\boxed{h = f\frac{d}{d_{pix}}} \tag{6}$$

A partir de la *ecuación 6*, podemos realizar un análisis de forma estática de la altura con la cual hay que volar para tener el tamaño en píxeles deseado de la medida del objeto real. El modelo se ajustó a los círculos del patrón, los cuales tienen 1cm de diámetro real, y midiendo los píxeles de la mejor imagen tomada en los vuelos anteriores, una buena resolución del círculo se da con 10 píxeles.

$$h = f\frac{d_{pix}}{d} = 4545[píxel]\frac{0{,}01[metros]}{10[píxel]}$$
$$\boxed{h = 4{,}545 metros} \tag{7}$$

Una vez fijada la altura, se procede a analizar como varía el "borroneo" según la velocidad y el *shutter speed*. Se propone alcanzar un ΔX_{pix} chico, o ~~lo mas aceptable posible~~. Se analiza la siguiente función:

$$\Delta X_{pix} = f_{pix}\frac{v\Delta t}{h} \tag{8}$$

Siendo Δt el *shutter speed*, v la velocidad de vuelo, h la altura teórica encontrada para poder tener 10 píxeles de resolución por centímetro y f la distancia focal en píxeles.

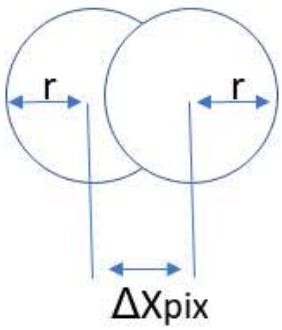


Figura 14: Estimación comportamiento del blur

La *figura* **??** muestra que producto de la velocidad por el *shutter speed* hay que tener para lograr un "borroneo" de X cantidad de píxeles en la imagen. Esto es útil para encontrar una buena relación entre estos dos parámetros y ,por ejemplo, tener un "borroneo" menor a 10 píxeles.

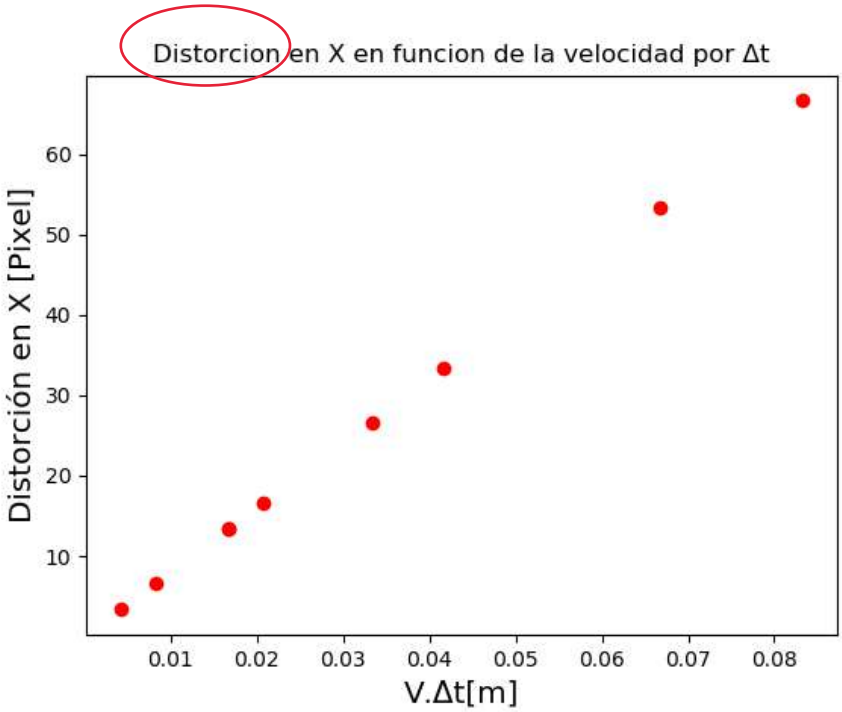


Figura 15: Gráfico la distorsión en X en función de la velocidad y el *shutter speed*

Si fijamos el *shutter speed* en la *ecuación (8* y se grafica la distorsión en el eje X :

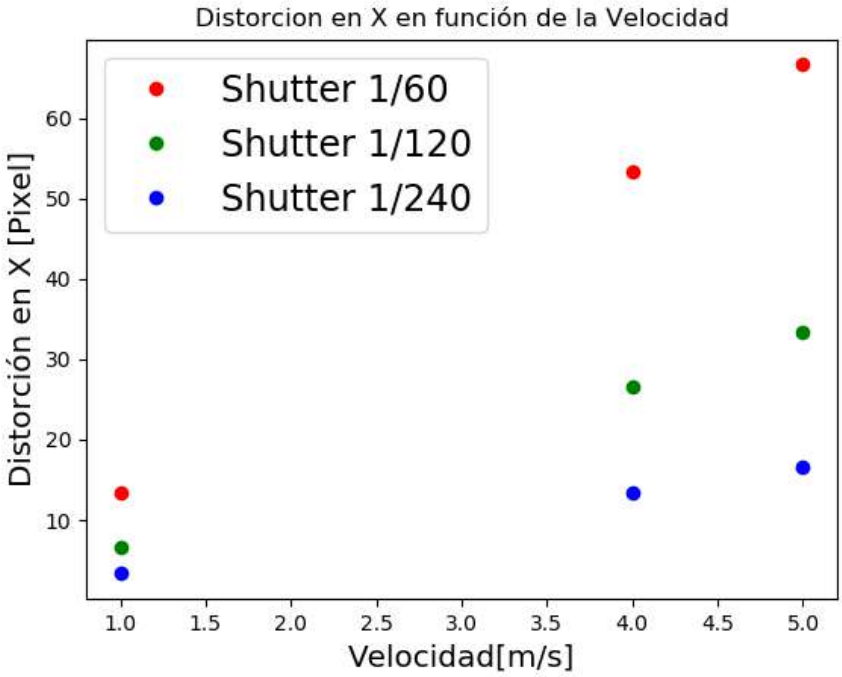


Figura 16: Gráfico la distorsión en X en función de la velocidad

El menor borroneo es de aproximadamente 4 píxeles y se da con el *shutter speed* de $\frac{1}{240}s$ a una velocidad de $1\frac{m}{s}$

4.2.2. Ajustes de elipse

Este método busca encontrar en la imagen de los patrones las circunferencias con el objetivo de ajustarle una elipse con centro y ejes conocidos. Este ajuste se realiza para cuantificar cuanto se borronea un círculo del patrón, durante los vuelos realizados, con respecto a la imagen original. Se sabe que cuanto mas borrosa la imagen, los círculos del patrón se deforman de manera elíptica. Teniendo en cuenta esto, suponemos que cuanto más parecidos sean el tamaño de los ejes de la elipse ajustada entre si, más parecido a una circunferencia son los píxeles que representan al círculo original del patrón en la imagen. Por ende, cuanto mas diferente sean, mayor será el borroneo.

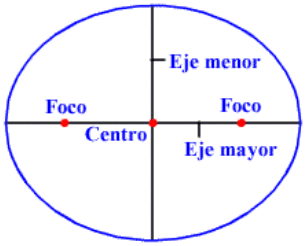


Figura 17: Característica de una elipse

Para este procedimiento, se realiza una serie de operaciones de pre-procesamiento sobre la imagen que consiste en realizar una ecualización por histograma y luego binarizar sobre la zona del patrón. La *figura 18* corresponden a los patrones binarizados para diferentes condiciones de vuelo:

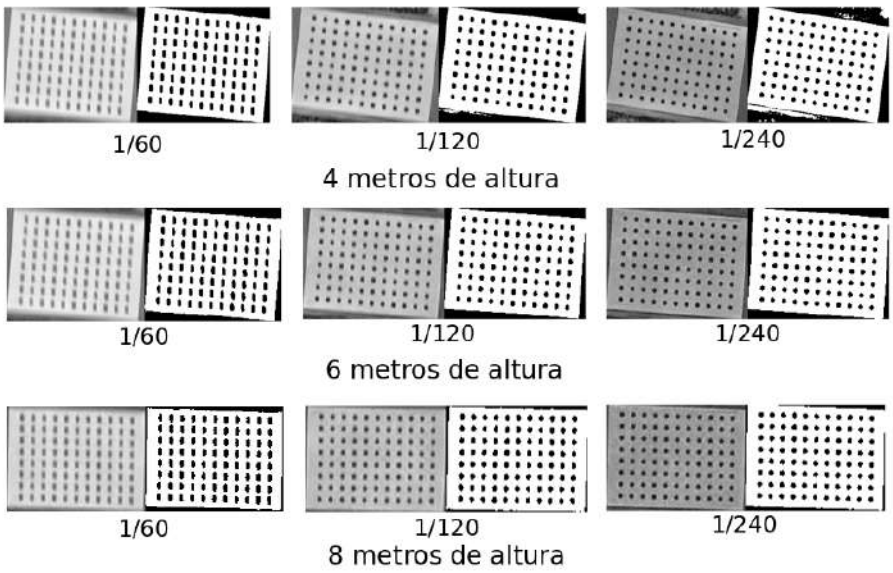


Figura 18: Volando a una velocidad de 1 m/s

Esta mismo proceso se realizó también para los vuelos de 4 $\frac{m}{s}$ y 6 $\frac{m}{s}$, para las 3 mismas alturas.

Al tener la imagen binarizada, ~~resultó sencil~~lo encontrar los contorno de las elipses usando la librería de *OpenCV* (7) con su algoritmo para ajustar elipses (22), para implementar la técnica explicada anteriormente. Este método pretende darle un enfoque mas geométrico a la medición del *blur*, y para esto se realizaron los gráficos que se muestran en las *figuras 19, 20* y *21*

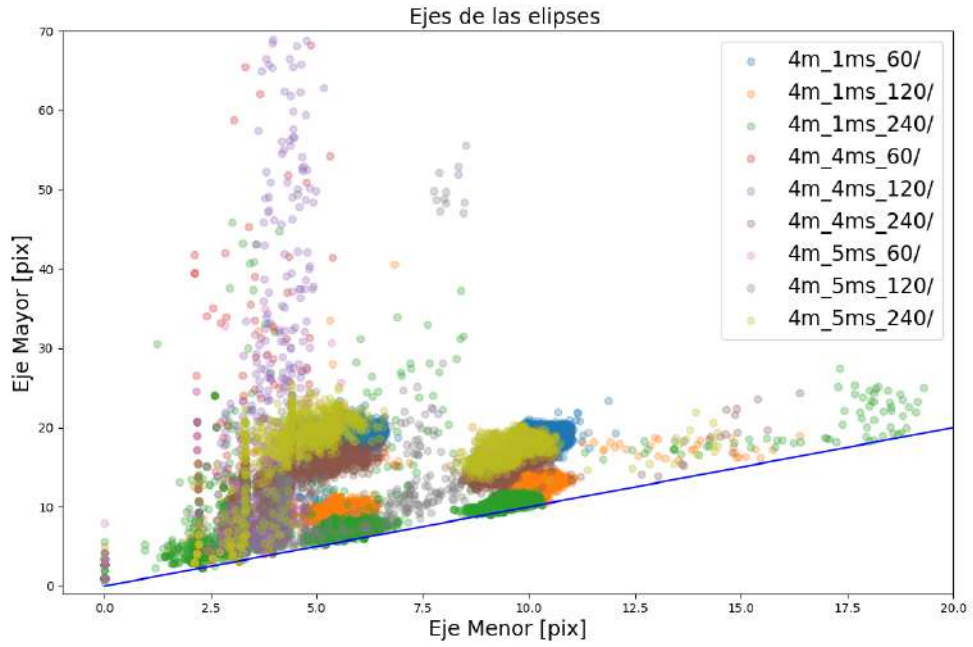


Figura 19: Gráfico de un eje de la elipse respecto del otro para patrones capturados a 4m

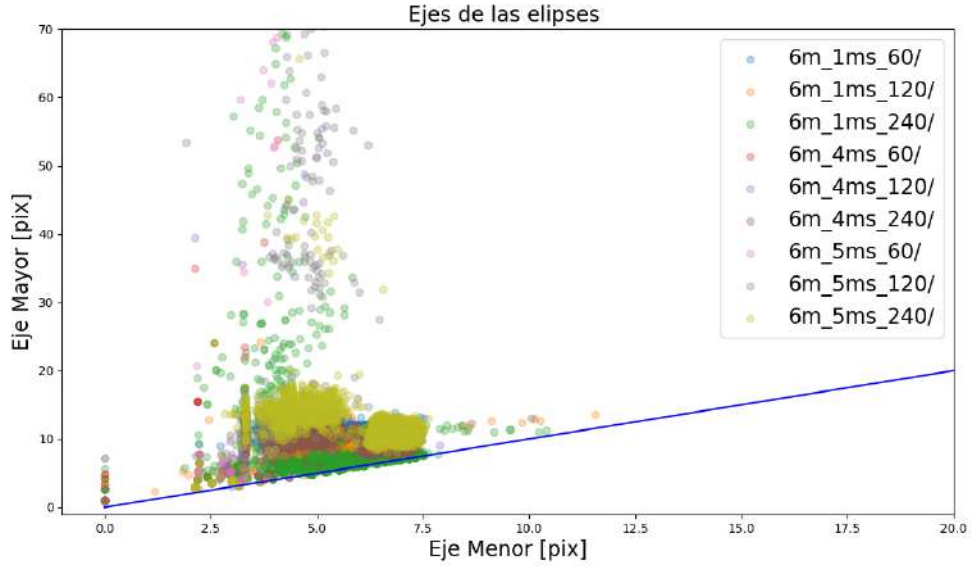


Figura 20: Gráfico de un eje de la elipse respecto del otro para patrones capturados a 6m

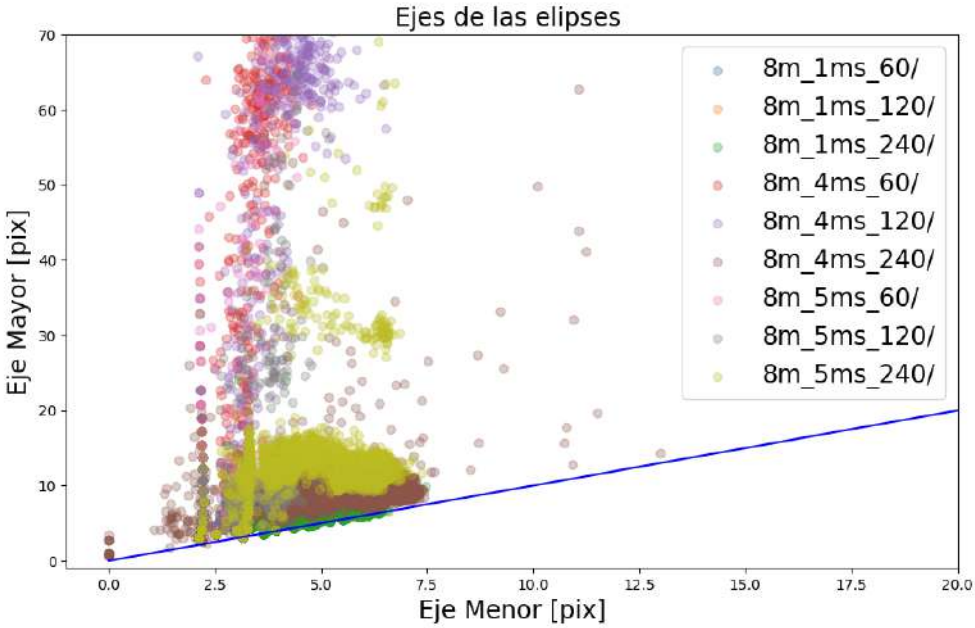


Figura 21: Gráfico de un eje de la elipse respecto del otro para patrones capturados a 8m

Estos gráficos representan al eje mayor de las elipses como la variable y, y al eje menor como la variable x. Las nubes de puntos representan las elipses ajustadas para distintas condiciones de velocidad y de *shutter speed*. Los gráficos contienen una recta identidad, que representa cuando los ejes son iguales en tamaño. Esto último quiere decir que los puntos representados mas cercanos a la recta, son los círculos de los patrones que sufrieron menos borroneo. En el caso de la *figura 19*, vuelo a 4 metros, se puede apreciar una

diferencia de dos sectores de puntos, que representan los dos patrones de círculos. Esto se debe a que a baja altura de vuelo la diferencia de tamaño se puede apreciar de manera correcta y, cuando el vuelo se realiza a mayor altura, esta diferencia no se distingue de manera notoria. No obstante es bueno tener los tres gráficos ya que en los mismos, se ve como tendencia a los puntos verdes acercarse de manera uniforme a la recta identidad. Estos puntos verdes identifican a los vídeos tomados a $1\frac{m}{s}$ con un *shutter speed* de $\frac{1}{240}s$.

En la *figura 19*, donde se distinguen dos *clusters* de color verde asignados a los vuelos con *Shutter speed* de $\frac{1}{240}s$ podemos realizar un análisis extra con lo explicado en la **sección 4.2.1**.
Los centros de estos *clusters* determinan el diámetro promedio aproximado de los círculos del patrón. Según el gráfico, estos centros se encuentran aproximadamente en 5,7 píxeles y 9.5 píxeles sobre la recta identidad respectivamente de los patrones de 0.5 cm y 1 cm. Con la propuesta de que 10 píxeles equivalen a 1 cm, se analiza la *ecuación 5* con los parámetros de altura del gráfico *figura 19* y la distancia focal de la calibración realizada.

$$d = \frac{d_{pix}.h}{f} = \frac{5[pixel]4[metros]}{4545[pixeles]} = \boxed{0{,}44cm} \tag{9}$$

$$d = \frac{d_{pix}.h}{f} = \frac{10[pixel]4[metros]}{4545[pixeles]} = \boxed{0{,}88cm} \tag{10}$$

Con las *ecuaciones 9* y 10 sabemos que el error promedio de la digitalización de los patrones de 0.5 cm y de los patrones de 1 cm es del 12 %. El borroneo para estas condiciones de vuelo se puede comparar con la *figura 16*, donde el borroneo para un *Shutter speed* de $\frac{1}{240}s$ y una velocidad de $1\frac{m}{s}$ es menor a 4 píxeles. Los centros de los *clusters* están dentro de la consideración de borroneo del modelo de predicción propuesto.

4.2.3. Varianza del Laplaciano

El algoritmo consiste en convolucionar la imagen del patrón con el operador Laplaciano, en todo los píxeles de la imagen y luego tomar la varianza (es decir, la desviación estándar al cuadrado) de la respuesta.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Figura 22: Kernel Laplaciano

La razón por la que este método funciona se debe a la definición del operador Laplaciano en sí, que se utiliza para medir la segunda derivada de una imagen. El Laplaciano resalta las regiones de una imagen que contiene cambios rápidos de intensidad. El Laplaciano a menudo se usa para la detección de bordes. La suposición aquí es que si una imagen contiene una gran varianza, entonces hay una amplia cantidad de bordes en la imagen. Pero si hay una variación muy baja, entonces hay una pequeña cantidad de bordes en la imagen. Cuanto más borrosa es una imagen, menos bordes bien definidos hay.

Utilizaremos este criterio para cuantificar el "borroneo" (19). Este criterio implica calcular la varianza del Laplaciano sobre los dos patrones en cada video, bajo las condiciones de vuelos implementadas, y determinar en que circunstancia la varianza del Laplaciano fue mas alta en los dos patrones.

Se mostrarán los gráficos obtenidos en diferentes condiciones de vuelo, fijando algunos parámetros mencionados anteriormente para realizar el análisis en los gráficos. Se podrían mostrar los veintisiete gráficos que se generaron con este algoritmo pero no tendría relevancia ya que las tendencias que se muestran, en los gráficos que fijan los mismos parámetros, son las mismas.

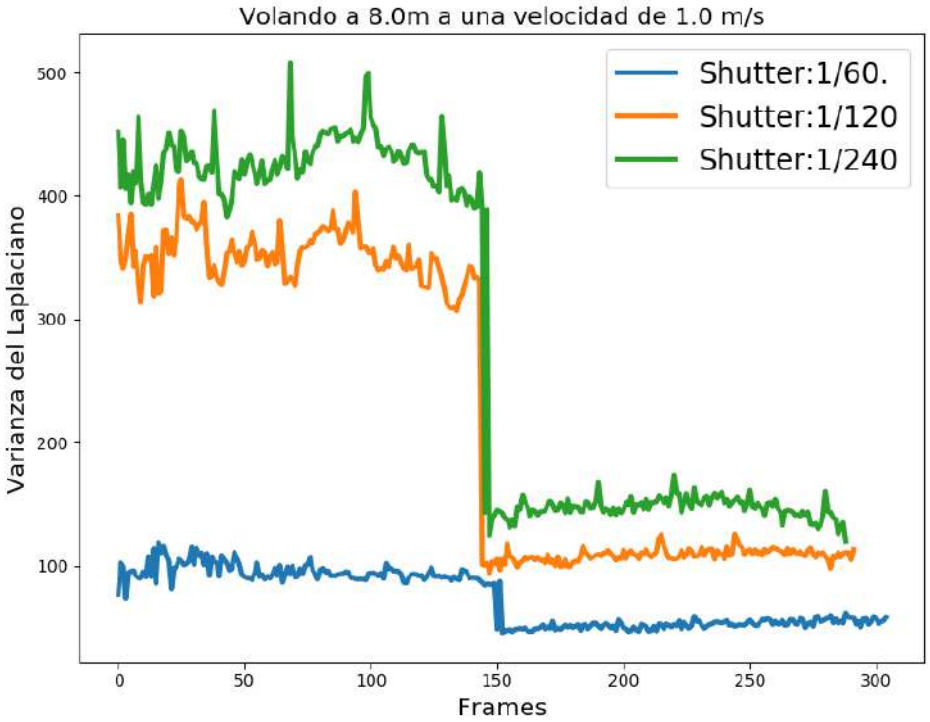


Figura 23: Variando el *shutter speed*.

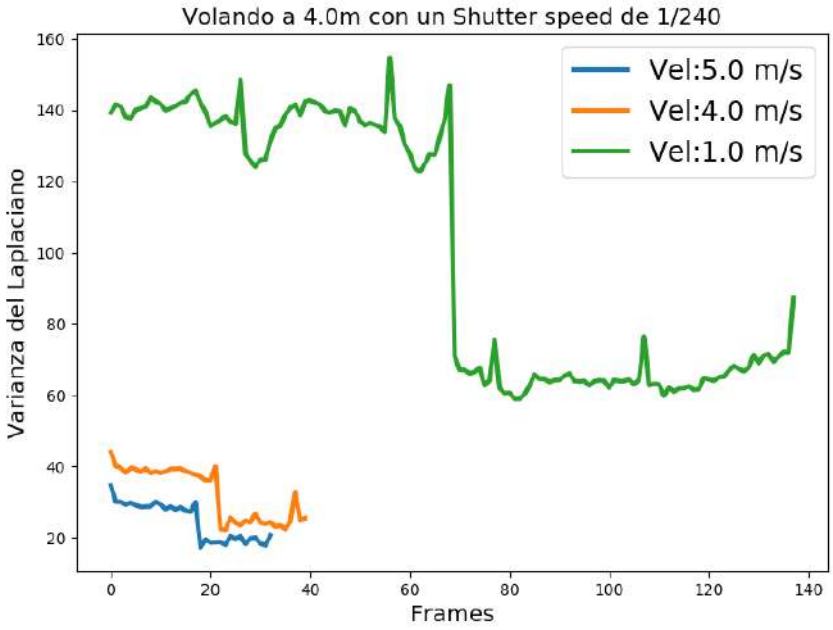


Figura 24: Variando la velocidad.

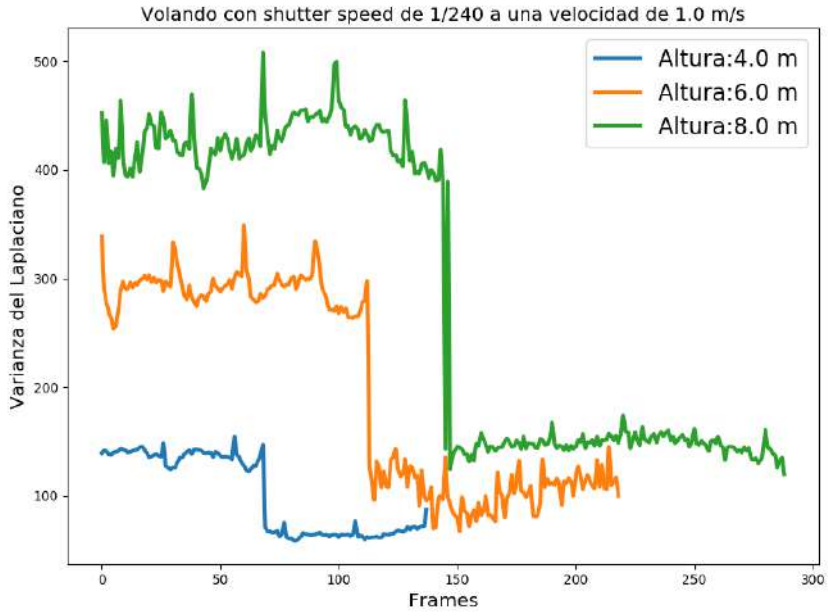


Figura 25: Variando la altura.

Con este enfoque, se puede ver en los *gráficos 23, 24 y 25* que los mejores resultados se da con una altura de 8 metros, a una velocidad de 1 $\frac{m}{s}$ y un *shutter speed* de $\frac{1}{240}s$

4.3. Conclusión

En las secciones anteriores se realizó un análisis de borroneo, teniendo en cuenta 3 métodos propuestos por este trabajo. Basándose en estos resultados, una propuesta para realizar los vuelos para la adquisición de imágenes de alta calidad es volar a una altura de 4 metros con una velocidad de 1 $\frac{m}{s}$ y un *shutter speed* de $\frac{1}{240}s$. Sin embargo, un detalle que se puede discutir es la influencia de la luz solar en el proceso de vuelo. Esto influirá con la elección del *shutter speed* ya que, a menor luz solar mayor tiene que ser el *shutter speed* y viceversa. Por lo tanto, la elección de un *shutter speed* como el de $\frac{1}{240}$ no es recomendado en días con poco sol. Para tener un margen de cobertura en cuanto a la grabación de vídeos de calidad en distintas condiciones de luminosidad, es recomendable elegir el *shutter speed* de $\frac{1}{120}s$, ya que el borroneo es de aproximadamente 5 píxeles y se pueden observar en los *gráficos* *16* y *19* .

5. Flight Record

En la actualidad, las aeronaves cuentan con un *Fligth Recorder* (conocido como *caja negra*) que se encarga de almacenar el monitoreo de los sensores en el vuelo incluyendo temperatura, velocidad, altitud y trayectoria, y de grabar todas las conversaciones entre los miembros de la tripulación y controladores aéreos. En el *drone DJI Mavic Pro 2*, utilizado para este proyecto, viene incorporado un dispositivo que cumple con la misma función que una caja negra. Si bien el motivo principal de este dispositivo es disponer de todos los registros para realizar una investigación completa en caso de un problema durante el vuelo, en este trabajo se utilizan estos datos con fines relacionados a la agricultura de precisión.

Figura 26: *Flight Records*

En esta sección se explicarán todos los conceptos relacionados con los *flight records*, desde la obtención de los mismos a través de los diferentes métodos que pone a nuestro alcance la marca DJI hasta el procesamiento de los mismos utilizando la librería pandas (11) para *Python* (6). Además, se hará una breve descripción de las secciones que utilizan estos archivos y de que modo se empleará.

5.1. Obtención de parámetros de vuelo

DJI fabrica sus equipos implementado redundancia en el almacenamiento de los *flight records*, de modo que el usuario puede obtener los mismos mediante el *drone* o el dispo-

sitivo de mando. Esto es útil en caso de que el *drone* realice alguna maniobra, impacte sobre un objeto y se averíe, debido a el dispositivo que almacena los *flight records* también puede sufrir daños que no permitan recuperar los datos del vuelo .De este modo, el operador tiene el mando para descargar la información de todos los vuelos que realizó.

5.1.1. Logfiles del drone

Uno de los métodos para extraer los datos del vuelo es utilizando los datos almacenados en el *drone*. Para realizar esta acción, el usuario tiene dos posibilidades, utilizar un cable que de un extremo tenga una conexión microUSB para el *drone* y del otro extremo una conexión USB para conectar en un equipo informático, o en caso de poseer insertada una memoria microSD, la misma se puede introducir en una PC que presente el *slot* compatible.

Luego de establecer la conexión con el *drone*, la aplicación detecta automáticamente el nuevo dispositivo y abre la pantalla principal del *DJI Assistant*, se seleccionan los archivos de interés para descargar y por último, se almacenan en la ruta seleccionada por el operador. Luego de finalizada la descarga, si bien estos archivos ya se encuentran en nuestro equipo local, estos no pueden ser leídos por ningún software debido a que se encuentran encriptados. Por este motivo, el fabricante ofrece una página web (23), que permite subir archivos a la nube de DJI, y que un algoritmo los procese y devuelva los parámetros. Como se muestra en la Fig.27 , esta plataforma web presenta información acerca de la trayectoria que realizó el *drone*, marcando en *Google Earth* la ubicación en cada instante de la trayectoria y una lista con los principales eventos y parámetros de vuelo.

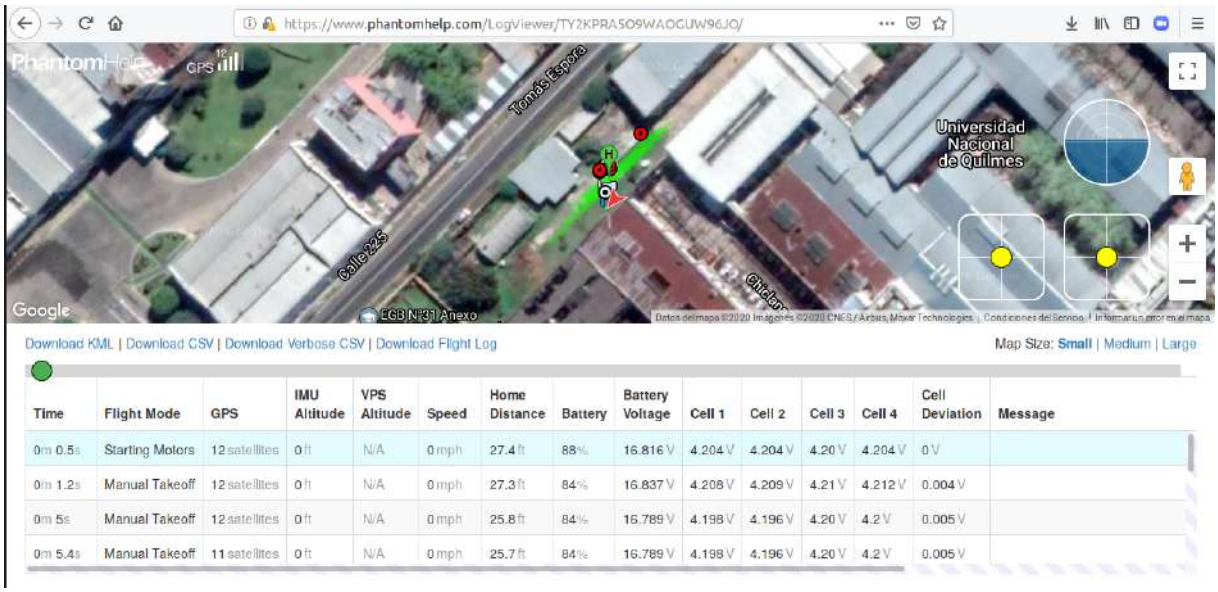


Figura 27: Plataforma web para descargar *logfiles*

En este proyecto es de gran necesidad poseer de un archivo, el cual posteriormente se pueda procesar con *Python* (utilizando la librería pandas ~~(11)~~). Para ello, esta plataforma

posibilita la descarga del archivo en cuatro formatos diferentes:

- **Archivo KML (*Keyhole Markup Language*):** formato basado en XML, que contiene los datos geográficos en tres dimensiones y permite importarlo en un software de geolocalización (como Google Earth (5)) para visualizar la trayectoria realizada por el *drone* en el vuelo.

- **Archivo CSV (*Comma-Separated Values*):** se visualiza con algún software capaz de representar datos en forma de tabla (como *Microsoft Excel*) y contiene mas de 40 parámetros del vuelo.

- **Archivo CSV Verbose:** similar al formato anterior, pero como su nombre lo indica, este contiene mas de 300 parámetros del vuelo.

(a) Formato KML

	A	B	C	D	E	F	G
1	Id	Time(seconds)	Time(text)	Latitude	Longitude	FlightMode	Altitude(feet)
2	1	0.5	0m	0.5s	-34.70665356	-58.27960291	Starting
3	2	0.6	0m	0.6s	-34.70665365	-58.27960298	Starting
4	3	0.7	0m	0.7s	-34.70665353	-58.27960302	Starting
5	4	0.8	0m	0.8s	-34.7066536	-58.27960294	Starting
6	5	0.9	0m	0.9s	-34.70665349	-58.27960297	Starting
7	6	1	0m	1s	-34.70665354	-58.27960305	Starting
8	7	1.1	0m	1.1s	-34.70665344	-58.27960308	Starting
9	8	1.2	0m	1.2s	-34.70665332	-58.27960316	Manual
10	9	1.3	0m	1.3s	-34.70665322	-58.2796032	Manual
11	10	1.4	0m	1.4s	-34.70665296	-58.27960328	Manual
12	11	1.5	0m	1.5s	-34.70665284	-58.27960333	Manual
13	12	1.6	0m	1.6s	-34.70665245	-58.27960339	Manual
14	13	1.7	0m	1.7s	-34.7066523	-58.27960344	Manual
15	14	1.8	0m	1.8s	-34.70665212	-58.2796035	Manual
16	15	1.9	0m	1.9s	-34.70665196	-58.27960354	Manual

(b) Formato CSV

Figura 28: Formatos de salida de *Flight Records*

5.1.2. Logfiles del celular

Como ya se mencionó anteriormente, los archivos de vuelo pueden descargarse a través del celular que se utilice para controlar el *drone*. Ante cualquier inconveniente físico con el dispositivo aéreo, esto permite obtener los datos para analizar su falla de todos modos. Si bien el proceso es muy similar al anterior, en este caso se utiliza la aplicación **DJI GO App** , que se puede descargar desde el *Play Store* (en caso de poseer un celular con Android), o desde el *App Store*, (en caso de poseer un celular con iOS).

Luego de instalar el software, se procede a iniciar la aplicación y desde el menú principal ingresar a la opción *Flight Records*. A continuación, aparecerá un cartel como el de la siguiente imagen, donde primero se debe seleccionar el rango de tiempo de los archivos para sincronizar, y luego hacer click en el botón *Start Synchronization* , como se muestra en la siguiente imagen.

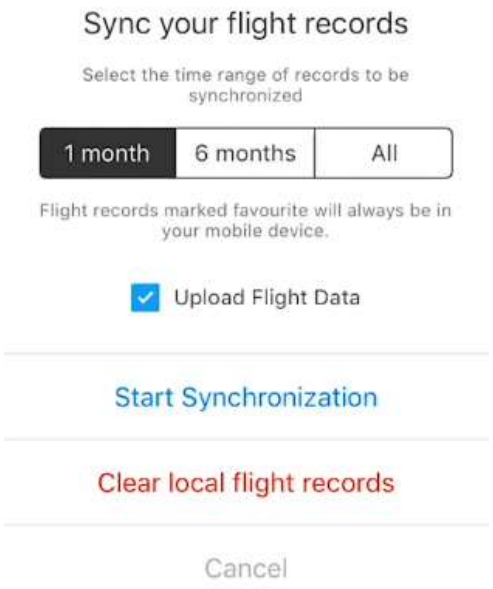


Figura 29: Sincronización de archivos con formato txt

Tomará unos segundos en subir los datos a la nube y luego ya estarán disponibles en formato txt para que el usuario los pueda descargar. A partir de este punto, el proceso es similar que utilizando *DJI Assistant 2*.

5.2. Descripción de parámetros

Independientemente de la elección del medio de descarga de los archivos *logfiles*, tanto el archivo CSV como el CSV Verbose presentan una gran variedad de datos de cada instante del vuelo. Si bien la mayoría de ellos son de gran utilidad, nos enfocaremos en este designio en particular, y detallaremos los campos que aportaron información valiosa para las diferentes partes del proyecto:

Función	Unidad	Descripción
CUSTOM.updateTime	AAAA/MM/DD HH:MM:SS.sss	fecha y hora de cada fila de datos
CUSTOM.isVideo	*Null/Recording/Stop*	indica si en ese instante se esta grabando un video
OSD.latitude	°	latitud en la que se encuentra el *drone* en ese instante
OSD.longitude	°	longitud en la que se encuentra el *drone* en ese instante
OSD.height	m	altura en la que se encuentra el *drone*
OSD.xSpeed	m/s	velocidad en el eje x con la que se mueve el *drone*
OSD.ySpeed	m/s	velocidad en el eje y con la que se mueve el *drone*
OSD.zSpeed	m/s	velocidad en el eje z con la que se mueve el *drone*
OSD.pitch	°	ángulo de *pitch* que presenta el *drone* en ese instante
OSD.roll	°	ángulo de *roll* que presenta el *drone* en ese instante
OSD.yaw	°	ángulo de *yaw* que presenta el *drone* en ese instante
GIMBAL.pitch	°	ángulo de posición de la catara del *drone*

Cuadro 5: Campos del archivo CSV mas utilizados

NOTA: El tiempo que se visualiza en la columna *CUSTOM.UpdateTime* se encuentra establecido en una zona horaria GMT 0. Por ende, este dato es de suma importancia a la hora de utilizar los logfiles para procesar las imágenes del drone, debido a que la hora se encuentra desfasada y algunos aspectos, como principalmente la iluminación solar, pueden variar ampliamente. Con la función *tseries.offsets.Hour(n)*, perteneciente al paquete de *pandas*, se corrigen los valores llevándolos a una zona horaria local (GMT -3 Buenos Aires en este caso).

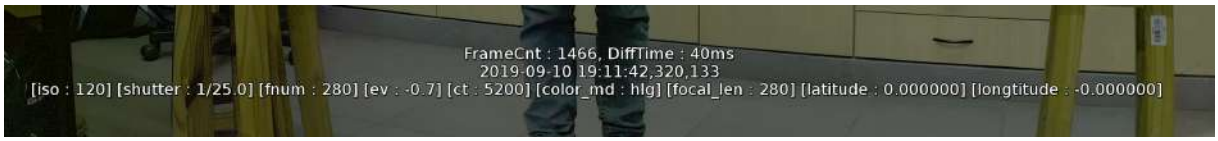
5.2.1. Flight records en subtítulos de videos

Complementando con los *flight records* que se pueden descargar y visualizar como se expresó recientemente, existe un método adicional para ver los registros de vuelo. Si bien los datos que se pueden visualizar son pocos, para este proyecto es de gran utilidad al momento de implementar imágenes y videos capturados con la cámara del *drone*. Cuando se descargan estos archivos en formato de video *AVI*, se pueden observar estos datos junto con la reproducción del mismo habilitando los subtítulos, los cuales vienen explícitos en un archivo STR. Estos proporcionan a información básica de vuelo, entre los cuales

se encuentran fecha y hora, algunos parámetros propios de la cámara como el ISO, la velocidad del obturador y la apertura de la cámara, y las coordenadas de latitud y longitud.

```
1
00:00:00,000 --> 00:00:00,033
<font size="36">FrameCnt : 1, DiffTime : 33ms
2019-11-06 08:23:14,277,447
[iso : 110] [shutter : 1/160.0] [fnum : 280] [ev : 3.0] [ct : 5200] [color_md : hlg] [latitude : -34.593164] [longtitude :
-58.484661] </font>
```

(a) Archivo STR



(b) Imagen con subtitulo

Figura 30: Visualización de archivo STR en subtítulos

5.3. Procesamiento de archivos con pandas

Como consecuencia de la gran cantidad de datos existentes en los archivos *logfiles*, es necesario buscar alguna librería que facilite el trabajo de interactuar y filtrar datos en dichas tablas. La mejor solución fue utilizar *pandas* (11), una biblioteca de código abierto con licencia BSD (*Berkeley Software Distribution*) que originalmente su propósito principal era la gestión de datos financieros , como alternativa al uso de hojas de calculo (*Microsoft Excel*). Pero actualmente, se utiliza para el análisis de datos que cuentan con las estructuras de datos que necesitamos para limpiar los datos en bruto (*raw data*) y que sean aptos para el análisis (por ejemplo, tablas). Además, permite alinear datos para su comparación, fusionar conjuntos de datos, gestionar datos extraviados, etc. Pandas denomina *DataFrame* a la estructura de datos básica, es decir, a una colección ordenada de columnas con nombres y tipos, donde una fila representa un único caso (un instante del vuelo del *drone*) y las columnas representan atributos particulares (parámetros del vuelo).

Algunas de las funciones que se utilizaron en este proyecto son *pandas.readcsv*, *pantas.DataFrame*, *pandas.concat*, *pandas.to_datatimes* y *pandas.tseries.offsets.Hour(n)*.Todo el detalle de las funciones se puede hallar en el sitio oficial de la librería pandas (11).

5.4. Secciones del proyecto que utilizan Flight Record

En esta sección se explicó los conceptos fundamentales relacionados con los archivos *Flight Record*, desde la obtención de los parámetros a través de la aplicación **DJI Assistant** con los datos del drone o mediante la aplicación **DJI GO App** con los datos del mando del control, hasta el procesamiento de los mismos con la librería *pandas* de *Python* (6). En cada segmento del proyecto, el uso de estos archivos es de suma importancia, debido a que facilita una gran cantidad de datos necesarios para desarrollar cada parte del proyecto, y permite llegar a resultados con una resolución aceptable. La utilización de los mismos en casa segmento fue:

- **Condiciones de vuelo**

  Establecer las condiciones de vuelo previo a capturar las imágenes de los campos experimentales permite obtener las capturas con el menor "borroneo" posible. Para poder cuantificarlo, se utilizaron dos patrones, los cuales fueron capturados por la cámara del drone y posteriormente analizados. El algoritmo que analiza el *blur* en las imágenes requiere de los archivos *logfiles* para obtener la altura, la velocidad y el *shutter speed*(o velocidad del obturador) de cada uno de los videos para determinar los parámetros óptimos de vuelo. Además, un campo que se utilizó es el *GIMBAL.pitch*, que indica el ángulo de posición de la cámara, debido a que la sección del video que interesa analizar con el algoritmo tiene como condición necesaria que la cámara este direccionada hacia el suelo (90°).

- **Geolocalización**

  Con respecto a esta sección, el uso de los logfiles tuvo un enfoque hacia las coordenadas geográficas. Tanto el método **Geodetic** como el método **Elipsoidal** (ambos explicados detalladamente en el Cap.7, utilizan los campos latitud y longitud para hallar la distancia entre un punto seleccionado por el operador y un punto de referencia.

- **Control de Crecimiento**

  El capítulo 8 explica un algoritmo mediante el cual el operador tiene la posibilidad de seleccionar un campo y ver las condiciones de esa fracción de cultivo a lo largo del tiempo (Control de Crecimiento). Si bien el detalle de su funcionamiento no forma parte de esta sección, se puede resaltar el uso y procesamiento de los *logfiles* que presenta. Como ya vimos, los videos presentan un *logfile* muy acotado de información donde solo se ven los parámetros más relevantes. Pero para la creación de este algoritmo, se necesita información adicional, motivo por el cual se fusionó los archivos STR de las imágenes con los registro de vuelo en formato CSV (Verbose).

6. Clasificación de parcelas

Los videos e imágenes de los campos experimentales, que fueron tomados en la *Facultad de Agronomía de la Universidad de Buenos Aires (FAUBA)* en los diferentes días del año 2019, presentan a simple vista una sección de campos de cultivos, que se separan en figuras rectangulares semejantes mediante líneas (suelo) en dirección horizontal y vertical. A estas secciones se las denominará como **parcelas**.

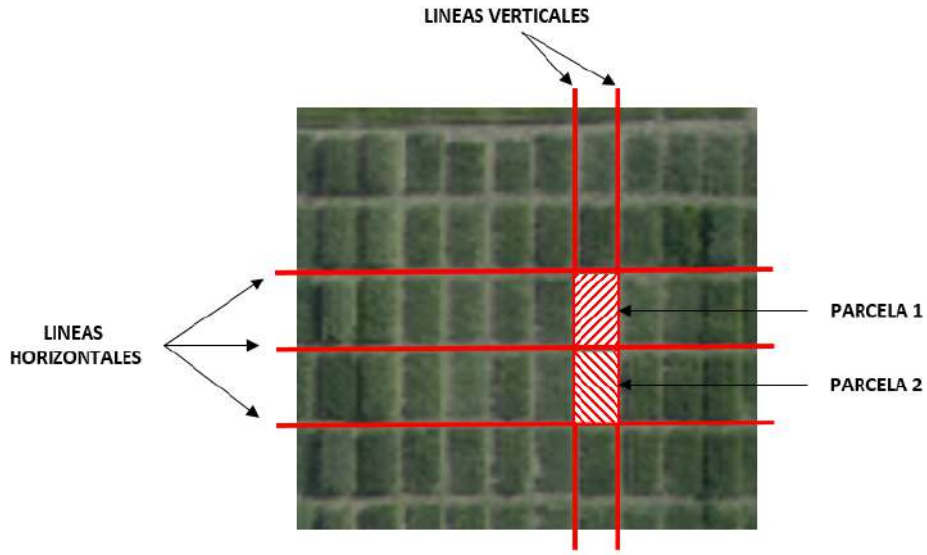


Figura 31: Parcelas y líneas de segmentación

Como su nombre lo indica, en esta sección se explicarán los diferentes métodos que se utilizaron para lograr la clasificación de las parcelas. El objetivo es darle al operador agropecuario todas las herramientas necesarias para que pueda seleccionar en una imagen una parcela dentro del campo, y que de la misma pueda obtener información necesaria para realizar posibles estudios. Para lograr este objetivo, se requiere de un método de segmentación eficiente. En esta sección se explicaran varios métodos de clasificación y se demostrara su grado de eficiencia.

6.1. Índice de verde

En una primera instancia, se realizó una clasificación utilizando un método que se basa en resaltar el canal verde (canal G) en una imagen definida en el espacio RGB llamado **índice de verde** (o en ingles *Green Index*). El procedimiento para realizar esta clasificación es el siguiente:

Se cargan las bibliotecas de *Python* (6) necesarias como *numpy* (9), *matplotlib* (8), *glob* (12) y *OpenCV* (7), y la imagen de los campos de cultivo (con la distorsión corregida) que se desea segmentar. Para comenzar , el índice de verde se calcula como:

$$I_{green} = 2G - R - B \tag{11}$$

donde G es el canal verde, R el canal rojo y B el canal azul. De esta manera, se logra resaltar el canal verde con respecto a los otros dos canales.

Luego para mejorar el resultado de la clasificación, se realiza una ecualización del histograma del tipo *clahe* (*Contrast Limited Adaptive Histogram Equalization*).Cuando se realiza una ecualización de histograma, se considera el contraste global de la imagen. Si bien en muchos casos los resultados son aceptables, en otros no se aproximan a lo esperado. Para solucionar esto, se divide la imagen en pequeños bloques y se ecualiza el histograma en cada uno de estos, y en caso de que haya ruido y para evitar que se amplifique, se aplica la limitación de contraste, donde cualquier contenedor de histograma que este por encima del limite de contraste doselecciona, los píxeles sufrirán un recorte y se distribuirán uniformemente a otros contenedores antes de aplicar la ecualización de histograma.

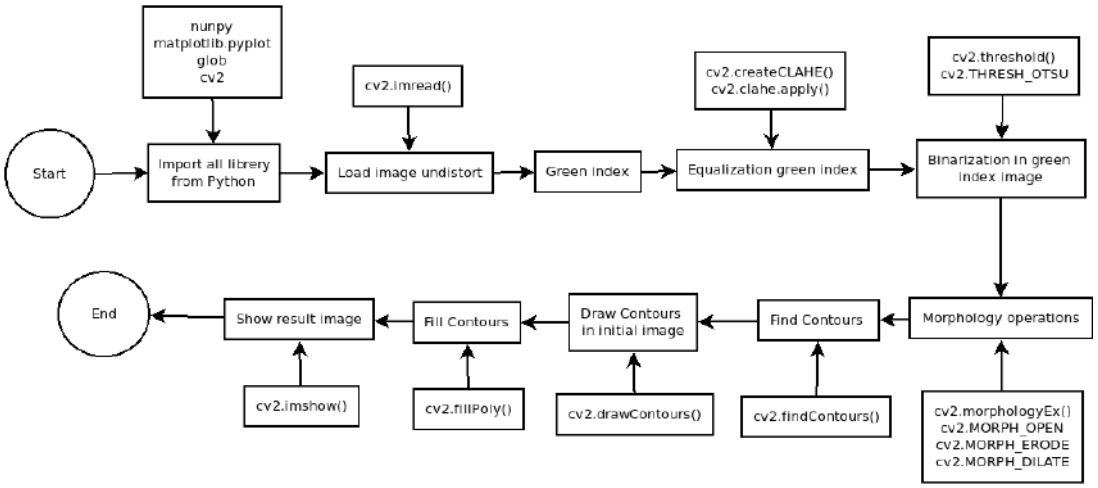
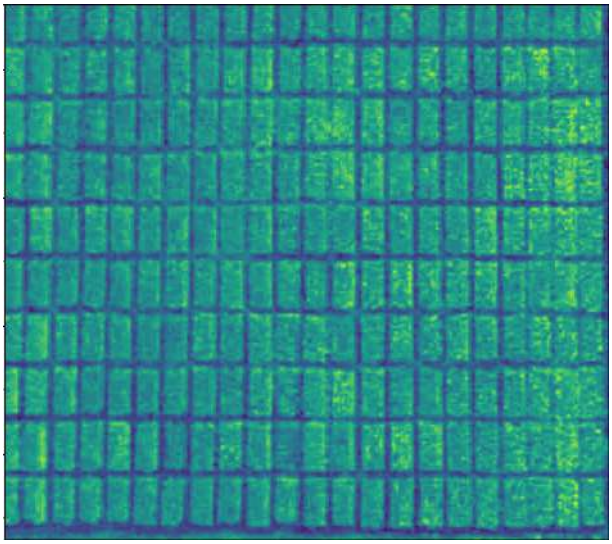


Figura 32: *Flowchart* del método Índice de verde

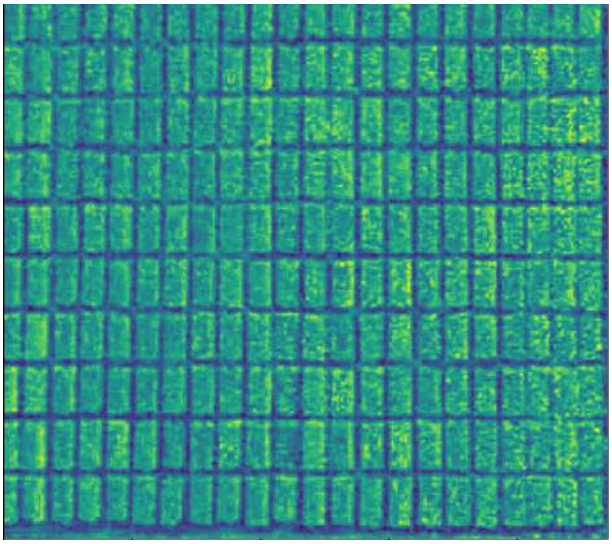
Usando la función *threshold* (30) de OpenCV (7), se realiza una binarización de la imagen donde todos los píxeles cuya intensidad sea menor que el umbral seleccionado, tomarán el valor 0, y los valores de intensidad mayores que el umbral seleccionado, tomarán el valor 1. La selección del umbral se realiza utilizando un método llamado **otsu**, que calcula automáticamente el valor de umbral del histograma de la imagen a binarizar. Además, para mejorar los resultados, se utilizan algunas operaciones morfológicas, como *Opening*, *Erode* y *Dilate*.

Con la función *findContours* (24) es posible detectar el borde de los gráficos de los campos de recorte y con la función *drawContours* (25) se dibujan en la imagen original.

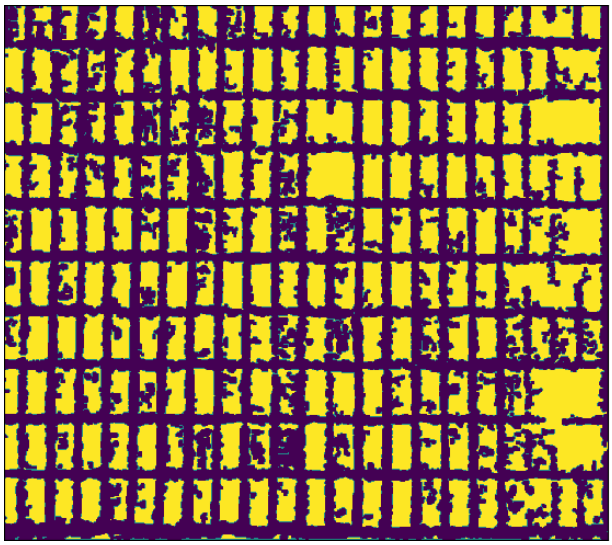
Finalmente, se realiza un relleno de los contornos con la función *fillPoly* (26) y se muestra la imagen.



(a) Imagen en índice de verde



(b) Imagen en índice de verde con filtros



(c) Bordes encontrados

(d) Interior de bordes rellenados

Figura 33: Procedimiento de segmentación por índice de verde

La clasificación de índice de verde permite segmentar las parcelas con un resultado aceptable, pero no tiene una gran eficiencia cuando las parcelas se aproximan, como se puede observar en la *figura 33 d*. La conclusión de este algoritmo es que no se puede utilizar como método de segmentación, pero si nos permite dar una primera clasificación para los próximos algoritmos. Es decir, podemos utilizar índice de verde como un preprocesamiento de la imagen antes de realizar el otro método.

6.2. Watershed con marcadores

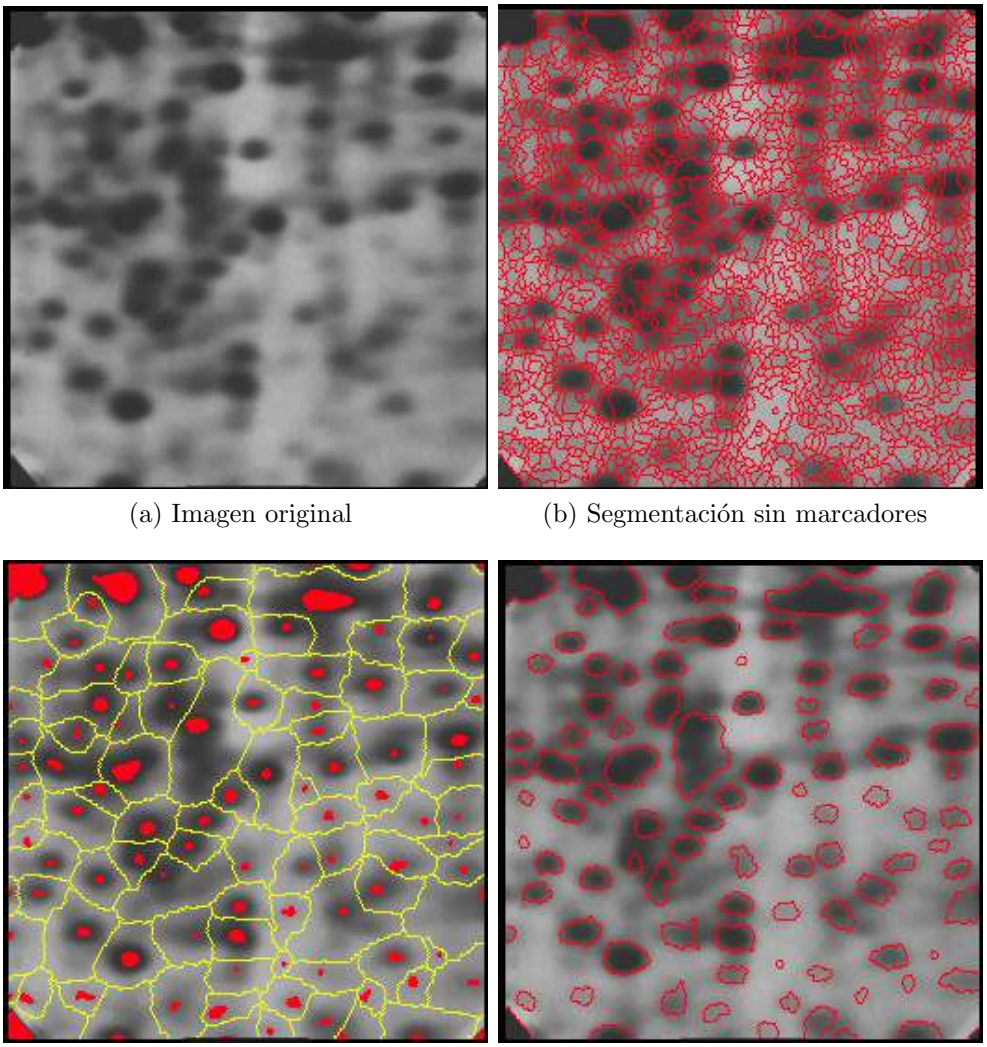
A partir de los resultados con el método *Índice de verde*, surge la necesidad de hallar otros modos de clasificación que puedan mejorar los resultados y segmentar correctamente los campos de la imagen. Uno de ellos fue el método de *watershed*, que para entender

su significado teórico, se utiliza una analogía geográfica. Cualquier imagen en escala de grises puede verse como una superficie topográfica donde la alta intensidad denota picos y colinas, mientras que la baja intensidad denota valles. Si se comenzaran a llenar todos los valles aislados (mínimos locales) con agua de diferentes colores (etiquetas), a medida que el agua sube, dependiendo de los picos (gradientes) cercanos, el agua de diferentes valles (con diferentes colores) comenzarán a fusionarse. Para evitar eso, se construyen barreras en los lugares donde se unirá el agua y se continúa el trabajo de llenar el agua y construir barreras hasta que todos los picos estén bajo agua. Las barreras dan el resultado de la segmentación.

(a) Imagen original (b) Superficie topográfica (c) Imagen segmentada

Figura 34: Procedimiento de segmentación por *watershed*

Pero en las imágenes existe ruido e irregularidades, por lo que el resultado seria excesivamente segmentado. Por este motivo, *OpenCV* (7) implementó un algoritmo basado en marcadores, donde la idea principal consiste en el uso de etiquetas, donde se etiquete con un color (o intensidad) la región que con seguridad será el primer plano u objeto, con otro color etiquetar la región que con seguridad sera fondo (o no sera objeto), y finalmente, rotular con 0 (cero) la región de la que se desconoce. Una vez establecidos los marcadores, se aplica el algoritmo de *watershed* ~~(27)~~. Luego, los marcadores se actualizarán con las etiquetas que introducimos en un principio, y los límites de los objetos tendrán un valor de -1. Esto se puede ver en la *figura 35*



(a) Imagen original (b) Segmentación sin marcadores

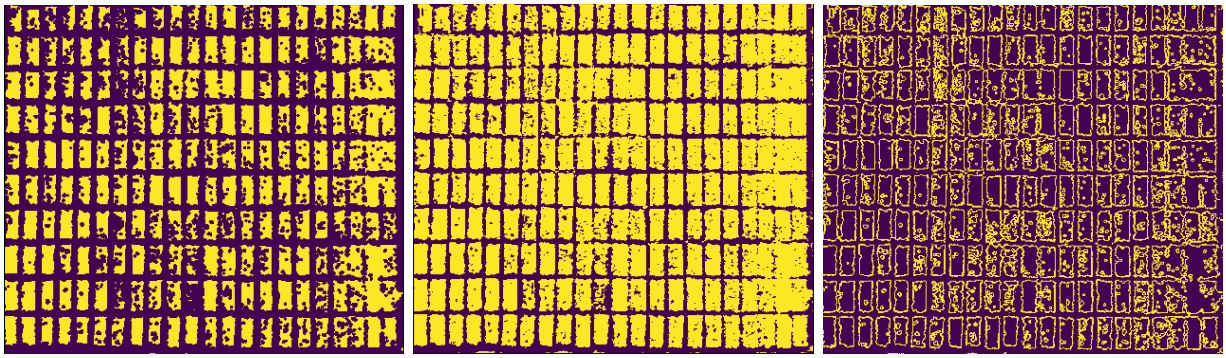
(c) Imagen con marcadores (d) Imagen segmentada con marcadores

Figura 35: Segmentación por *watershed* utilizando marcadores

A partir de la explicación del funcionamiento de *watershed* con marcadores, se detalla el procedimiento de uso en este proyecto.

En primer lugar, y como se mencionó en la sección anterior, el algoritmo de *watershed* utilizará una imagen que ya presenta cierto proceso de segmentación, que no es mas que el resultado del índice de verde.

Luego, se intenta eliminar ruido en la imagen utilizando un *Opening*. A su vez, para eliminar cualquier pequeño *agujero*, se utiliza la operación morfológica *Closeing*. Por ende, lo que se puede deducir es que la imagen con *Opening* puede considerarse con seguridad como *Foreground* (es parcela), y por otro lado, la imagen con el proceso de *Closing* puede considerar con seguridad *Background* (es decir, no es parcela) y la resta de ambas se considera como una región *Desconocida*. Además, la región restante, que en las imágenes se muestra con color violeta, es el suelo del campo de cultivos.



(a) *ForeGround* (b) *BackGround* (c) *Unknown*

Figura 36: Selección de secciones

Luego de conocer con certeza a que región pertenece cada píxel de la imagen, se procede a crear los marcadores. Las áreas *Foreground* y *Background* se etiquetan con diferentes enteros positivos y el área desconocida se le asigna el valor 0 (cero). Para realizar esto, se utiliza la función *connectComponents* (28), perteneciente a la librería de *OpenCV* (7). Pero como el método de *watershed* podría llegar a considerar al fondo como un área desconocida (por ser cero), se lo marca con un numero diferente y se establece un cero a la región de píxeles del área *Unknown*

Por último, ya con los marcadores creados, se utiliza la función **cv2.watershed** (27) para finalizar con la segmentación.

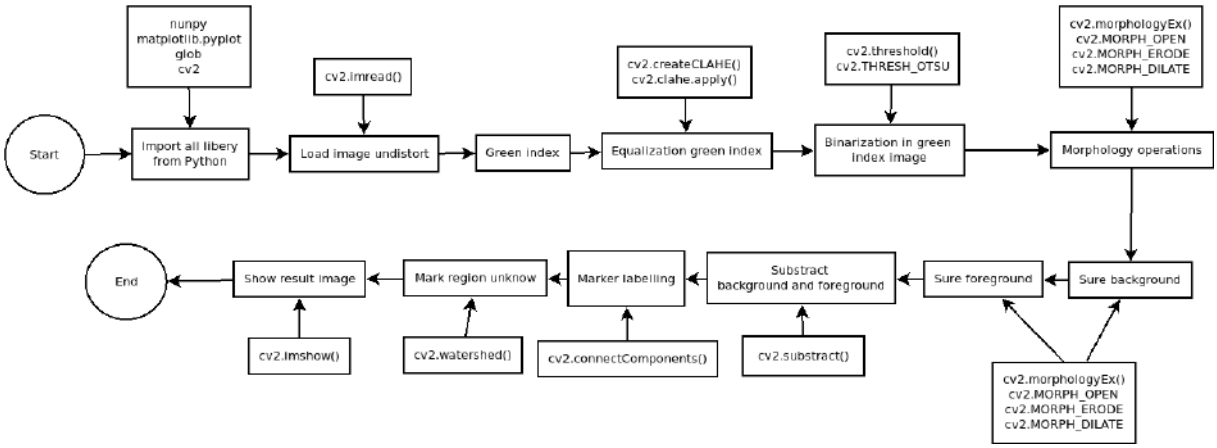
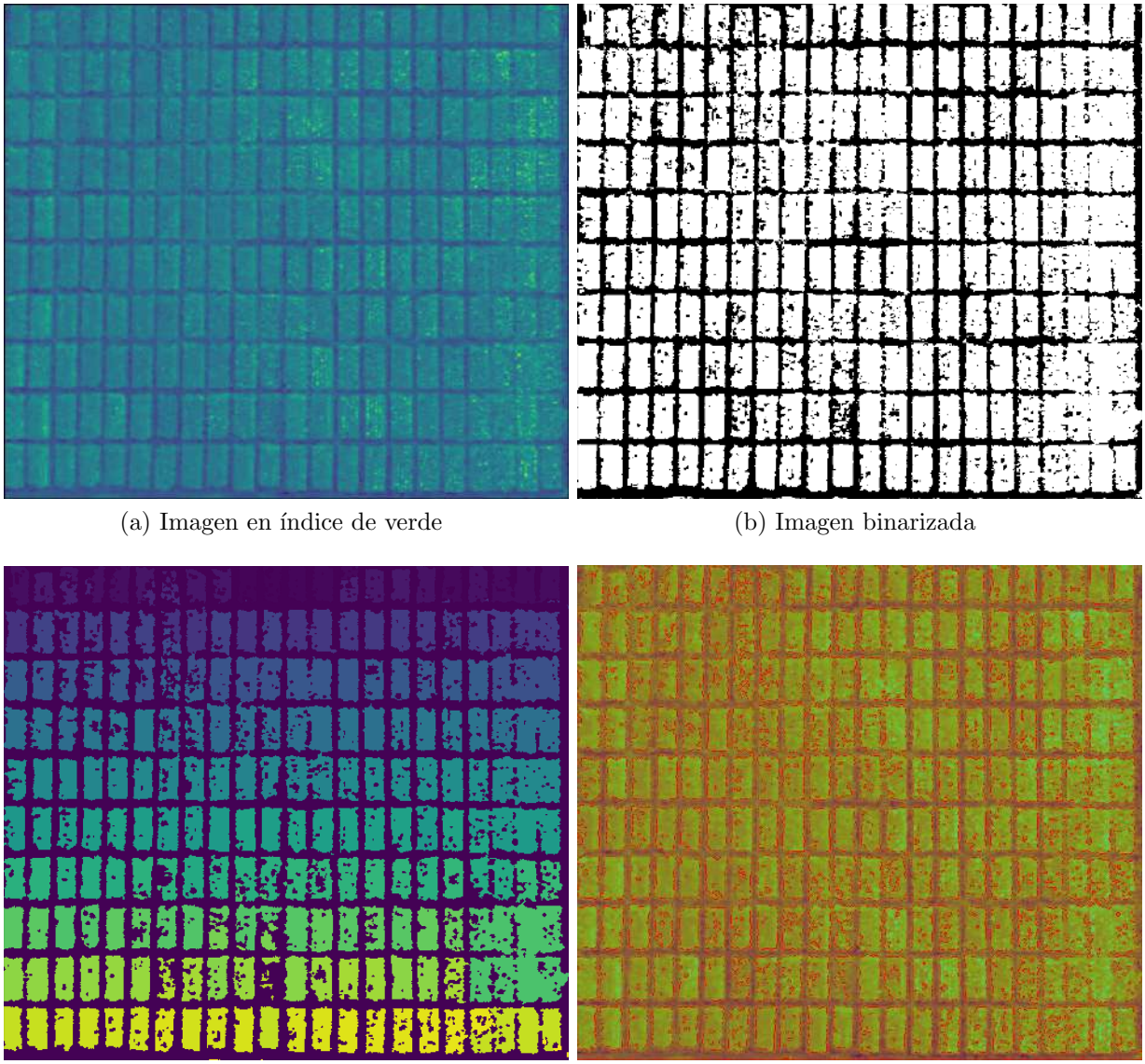


Figura 37: Flowchart del método de *Watershed*

Los resultados de este método se muestran en la Fig.38 :



(a) Imagen en índice de verde

(b) Imagen binarizada

(c) Marcadores

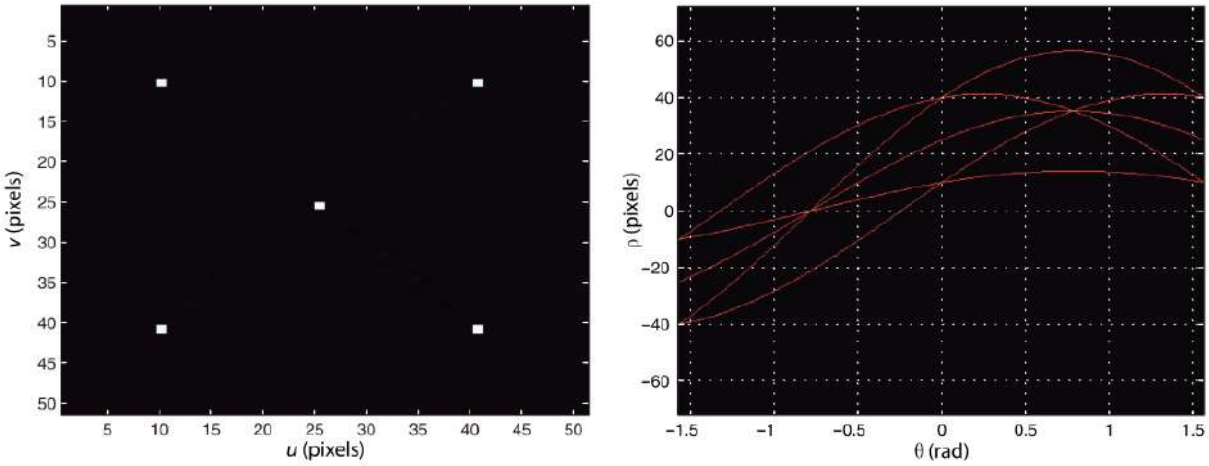
(d) Resultado de Watershed

Figura 38: Resultados con método *Watershed*

Si bien los resultados con este método superan a la clasificación por índice de verde, ocurre el mismo inconveniente con las parcelas que presentan una distancia reducida de separación. Si se observa las parcelas en la parte inferior derecha de la imagen, el algoritmo no detecta de manera certera la separación de las mismas, aun utilizando marcadores, como pueden apreciarse en la Fig.38 .Como conclusión, este método no presenta la suficiente eficiencia para separar las parcelas perfectamente.

6.3. Transformada de Hough

La transformada de Hough agrupa líneas en colecciones con puntos de fuga comunes mediante una representación en el espacio-ρ θ y la creación de una matriz acumuladora, como se explicará en los siguientes párrafos.



(a) Puntos en el espacio u, v (b) Rectas en el espacio ρ,θ

Figura 39: *Transformada de Hough* de cinco puntos

Una línea se puede representar como $y = mx + b$, pero muchas veces causa problemas para el caso de las lineas verticales donde $m = \infty$. Por este motivo, y describiendo cada línea en términos de un número mínimo de parámetros, se representa en su forma paramétrica como:

$$\rho = x \cos\theta + y \sin\theta \tag{12}$$

donde ρ es la distancia perpendicular desde el origen a la línea, y θ es el ángulo formado por esta línea perpendicular y el eje horizontal medido en sentido antihorario, que tal dirección va a depender en la forma en el que se representa el sistema de coordenadas.

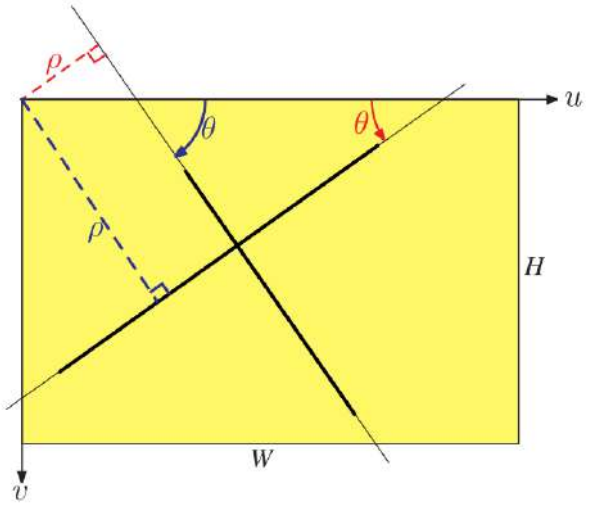


Figura 40: Parametrización (ρ,θ) de dos líneas. Valores positivos en azul y negativos en rojo

Ya con las líneas representadas en su forma paramétrica, se procede a crear una matriz **A** (que llamaremos acumuladora) de ρ filas por θ columnas e inicialmente, todos los valores se colocan en 0 (cero). El tamaño de la matriz depende de la precisión que se requiera para el resultado. Por ejemplo, si se necesitara una precisión de ángulo de 1°, se necesitarían 180 columnas. En cuanto al tamaño de ρ, la distancia máxima posible es la longitud de la diagonal de la imagen seleccionada debido a que no puede existir en la imagen otra línea de mayor longitud.

Para comprender el procedimiento del método con la matriz de acumulación, se realizará un ejemplo aclaratorio. Se considera una imagen de 10 x 10, y se exige una resolución de ángulo de 45°, que si bien es una resolución baja, facilitará los cálculos. Se supone que en la imagen solo se encuentra una línea horizontal, de la cual se saben todas las coordenadas x e y, de modo que se utilizará la Ec. 12 , y se obtendrá el valor de ρ para $\theta = 0°$, 45°, 90°y 135°.

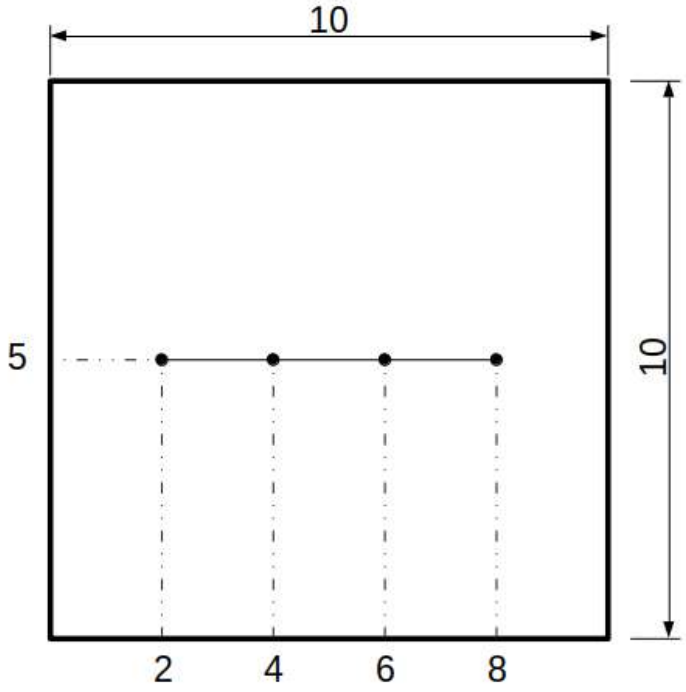


Figura 41: Imagen de 10cm x 10cm con linea horizontal

Por cuestiones de simplificación, solo se trabajará con cuatro puntos de la recta: $(2, 5)$,$(4, 5)$,$(6, 5)$ y $(8, 5)$. Los resultados son los siguientes:

(x, y)	0°	45°	90°	135°
(2, 5)	2	4	5	2
(4, 5)	4	6	5	0
(6, 5)	6	7	5	0
(8, 5)	8	9	5	-2

Cuadro 6: Resultados de ρ y θ

NOTA: Se toman solo valores enteros (se trunca parte decimal).

Para continuar, se adiciona un 1 en el espacio correspondiente de la matriz acumuladora para cada punto. En la siguiente imagen, se observa el escenario punto a punto, y además, como se van agregando a la matriz.

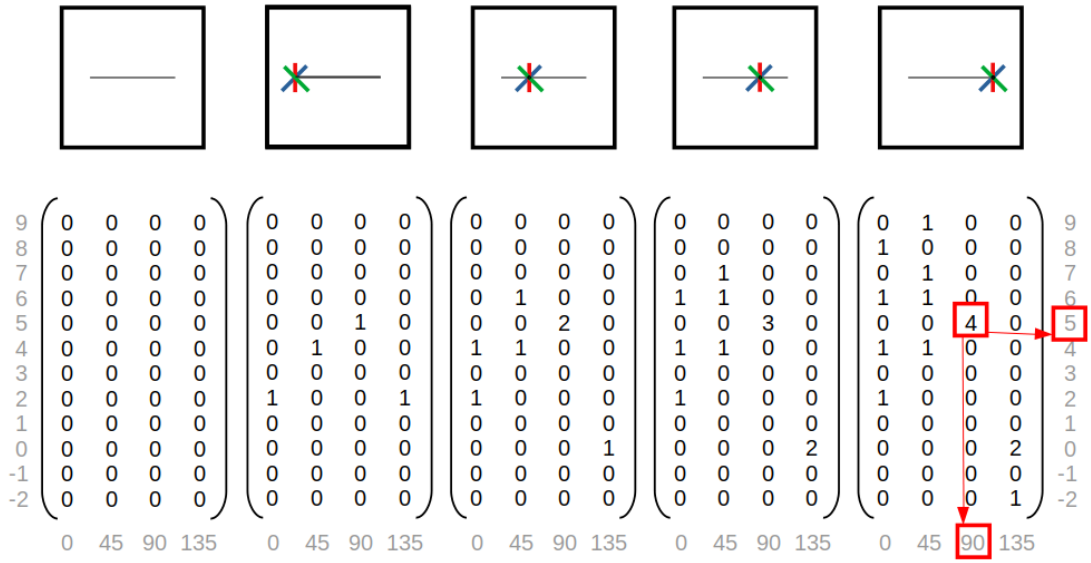


Figura 42: Evolución de matriz de acumulación punto a punto

Si se detiene a ver la matriz **A**, se observa que la posición (ρ,θ)= (5, 90) lleva el valor 4, que con respecto al resto de la matriz, es el máximo de acumulaciones. Esto nos lleva a la conclusión de que hay una línea en esta imagen a una distancia de 5 desde el origen de coordenadas y en un ángulo de 90°que une todos los puntos.

Una vez explicado el funcionamiento de la *transformada de Hough*, se procede a su implementación práctica. En principio el procedimiento es el mismo que el método de *Watershed* (27), debido a que utiliza las imágenes con la clasificación de *índice de verde* y las operaciones morfológicas de *Opening*, *Erode* y *Dilate* para mejorar la imagen. A partir de este punto, se utiliza el método de la *transformada de Hough*. *OpenCV* (7) contiene una función **HoughLines** (29) , que presenta los siguientes parámetros de entrada:

- imagen de entrada binaria. Por lo tanto, se requiere un preprocesamiento aplicando

un umbral de binarización correspondiente o utilizando una detección de borde correcta.

- precisión requerida del parámetro ρ
- precisión requerida del parámetro θ
- valor mínimo que debe obtenerse en la matriz acumuladora para que se considere como una linea. En otras palabras, es la mínima longitud de lineas deseables

Y la única salida es la matriz acumuladora **A**, con ρ medido en *píxeles* y θ en *radianes*.

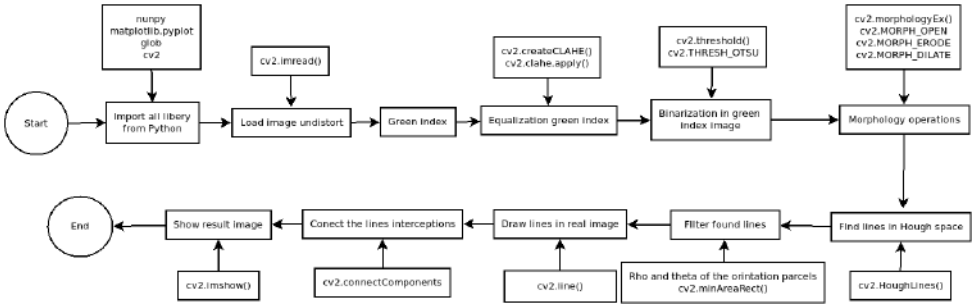


Figura 43: *Flowchart* de *Hough*

Si bien los resultados son acertados porque se encuentran una gran cantidad de líneas, estos no apuntan a nuestro objetivo. En todas las imágenes existe ruido, y esto genera que el algoritmo interprete esto como líneas, como se muestra en la Fig.44.

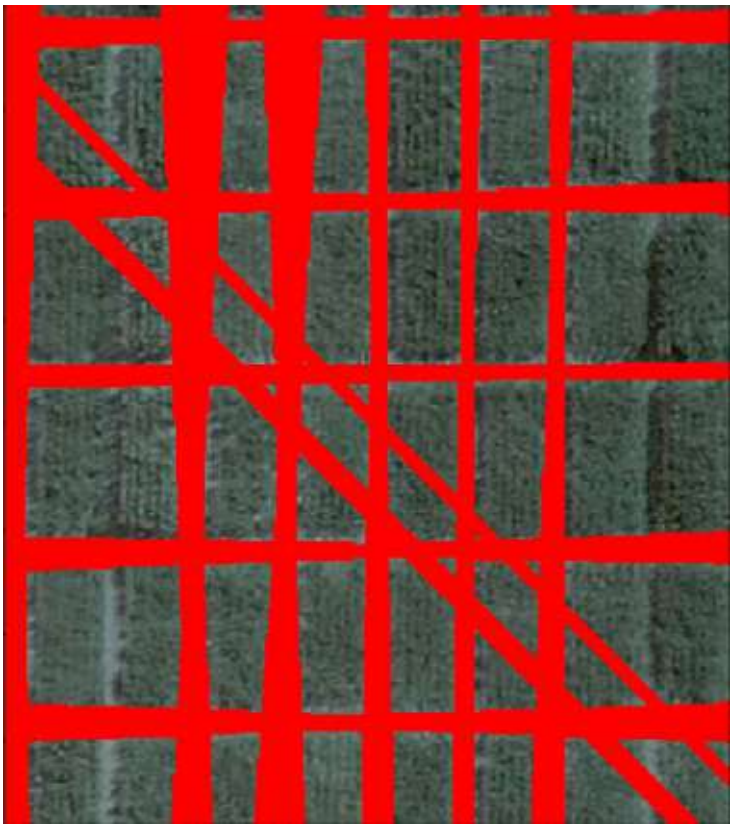


Figura 44: Resultado de *Hough* sin filtro

Para evitar este inconveniente, se utiliza el parámetro de resolución de θ para exigir al algoritmo que retorne solo las lineas verticales y horizontales que coinciden con la orientación de las parcelas de los campos de cultivo. En el histograma de la Fig.45, se observa como se concentran las los ángulos de las líneas en tres lugares de θ , correspondientes a los ángulos requeridos para seccionar las parcelas.

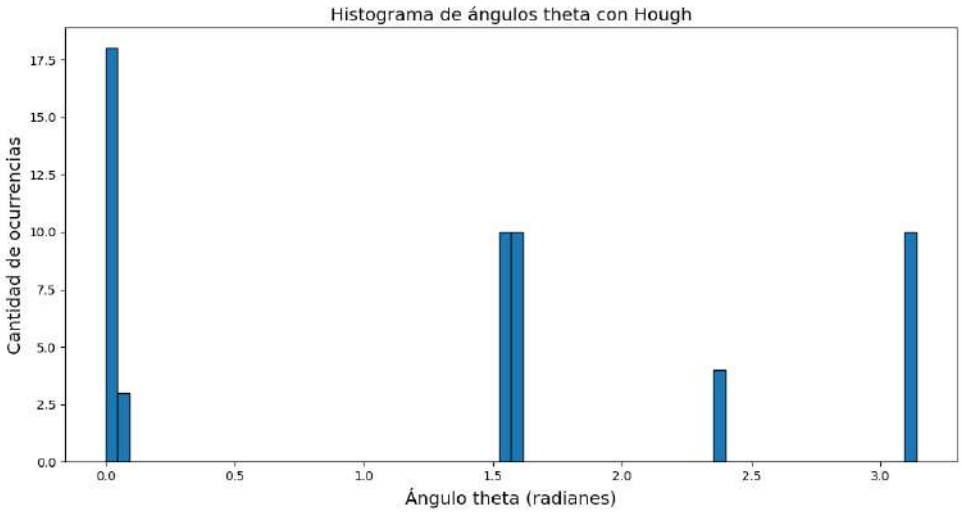
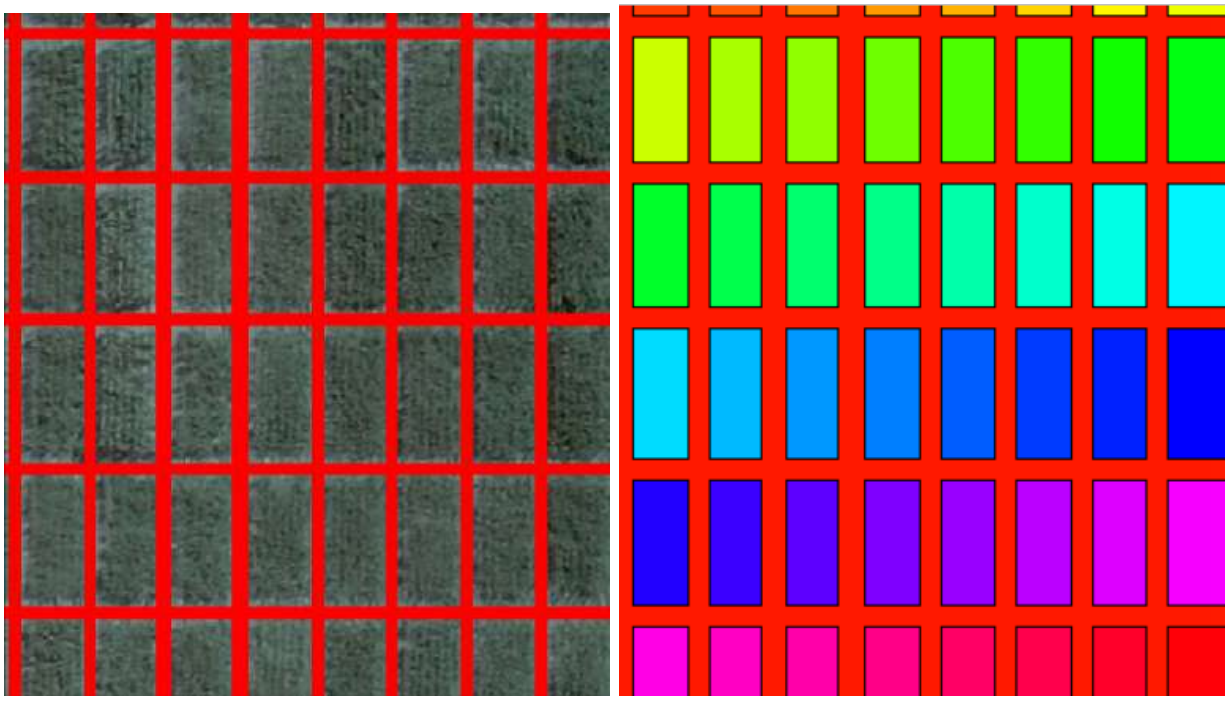


Figura 45: Histograma de ángulos de lineas

Luego, con la función *connectedComponents* (28), utilizada en el método anterior, se etiquetó a cada parcela de cultivo por separado.



(a) Segmentación de parcelas (b) Etiquetado de parcelas

Figura 46: Parcelas etiquetadas con *Hough*

Los resultados con la*transformada de Hough* superan a los que se obtuvieron tanto por Índice de verde, como por el método de *Watershed*. Las líneas separan los campos y se pueden identificar todas las parcelas de la imagen. Pero sigue existiendo un problema, el filtro de líneas. Como se explicó anteriormente, se requiere hacer un filtro de los ángulos de líneas que aparecen en la imagen debido a que el ruido o las imperfecciones de las parcelas pueden interpretarse mal por el algoritmo y definirla como líneas (como se vio en la *figura 44*). Por ende, el funcionamiento del método es eficiente, pero requiere la constante modificación del filtro debido a que no todos los campos presentan la misma estructura geométrica. A partir de aquí, surge la necesidad de encontrar un método que iguale los resultados del presente, pero que no requiera modificación constante del operador.

6.4. Método de la convolución

Para clasificar las parcelas, primero se requiere hallar los bordes (líneas) de las mismas y luego segmentarlas mediante algún método. Si se observa las imágenes de la Fig.47 , a la izquierda se puede identificar un cartel de alguna ciudad del mundo que ofrece información para los peatones con una línea horizontal verde que cruza toda la imagen, y al lado derecho, se grafica la intensidad de los píxeles de dicha línea verde. Los picos altos visibles corresponden a las letras blancas y otras marcas en el letrero.

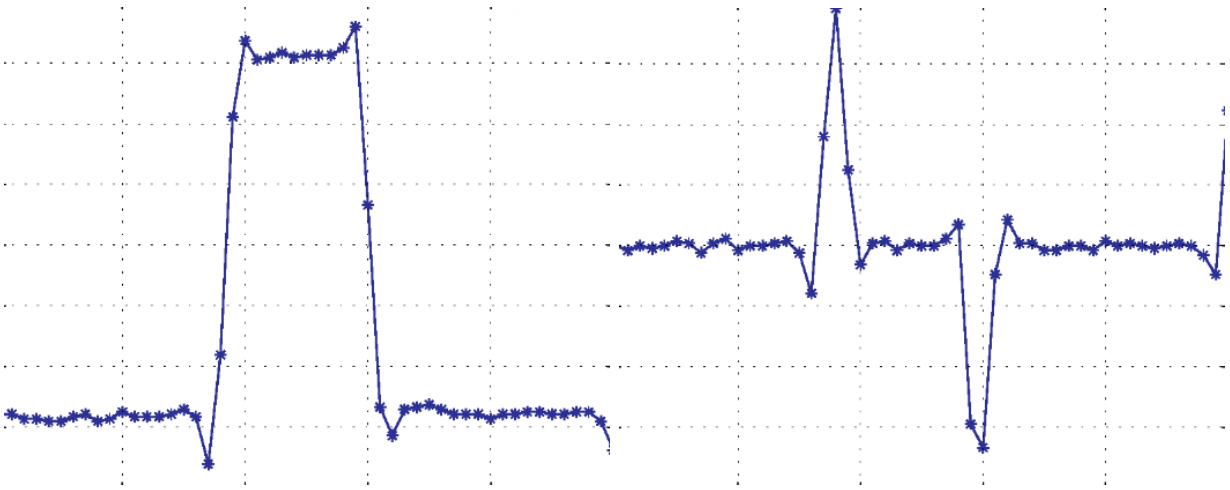
(a) Imagen de cartel (b) Gráfico de intensidad de gris de una linea de la imagen

Figura 47: Intensidad de grises en linea del cartel

La intensidad de fondo se aproxima al valor 0.3 y la intensidad brillante al valor 0.9 pero dependerá de los niveles de iluminación. Sin embargo, el aumento veloz y abrupto de solo unos pocos píxeles es distintivo y determina la existencia de un borde. La derivada de primer orden a lo largo de esta sección transversal es:

$$p^{'}[v] = p[v] - p[v-1] \tag{13}$$

que gráficamente nos indicarían que existe un borde debido a la existencia de un pico, como se muestra en las *imágenes* 48 que corresponden a la T del tablero de la Fig. 47.



(a) Gráfico de intensidad sobre letra T (b) Derivado de intensidad del gráfico de la izquierda

Figura 48: Derivada de letra T del tablero

La señal se mantiene en un valor cercano a cero con claras respuestas diferentes de

cero en los bordes de un objeto, en este caso los bordes de una parte vertical de la letra T.

La derivada en el punto v también se puede escribir como una diferencia de primer orden

$$p^{'}[v] = \frac{1}{2}(p[v+1] - p[v-1]) \tag{14}$$

que es lo mismo que realizar la convolución utilizando un **núcleo** (o también conocido como su nombre en ingles *kernel*) de una dimensión:

$$K = (-\frac{1}{2}; 0; \frac{1}{2}) \tag{15}$$

y que finalmente, el resultado muestra los bordes verticales del cartel

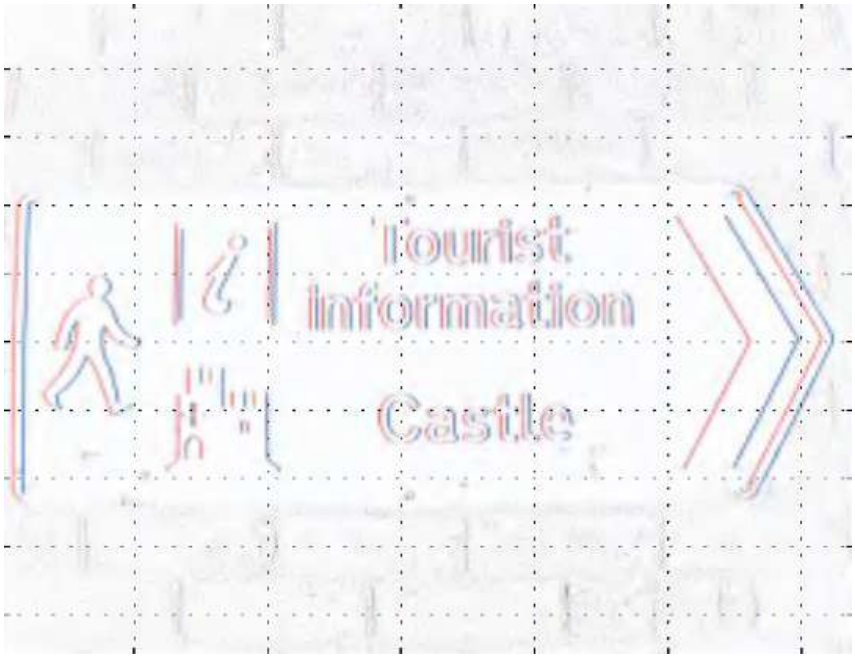
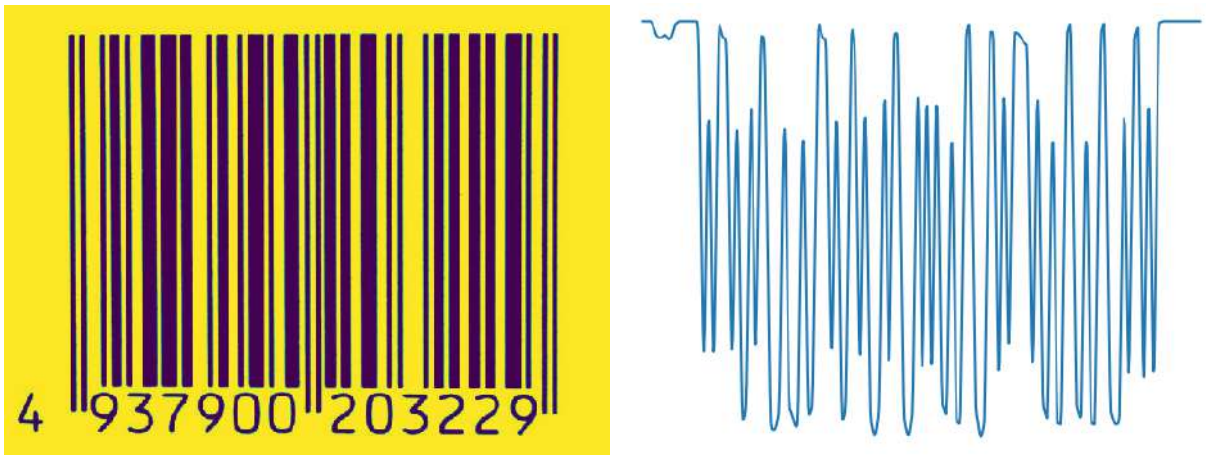


Figura 49: Bordes verticales del cartel

Si bien existen otros núcleos para obtener bordes en una imagen (como por ejemplo el kernel de *Sobel*), en este algoritmo se utilizó este concepto de la convolución de imágenes, pero la misma se realiza entre la imagen de los campos y líneas en diferentes direcciones. Hay que recordar que el objetivo es clasificar las parcelas, por ende si se realiza la convolución vertical de solo una línea recta horizontal del largo de la imagen con la fotografía de las parcelas, en las regiones donde se presenta el suelo los valores de la convolución serán elevados, mientras que en las regiones donde existe cultivo (sección parcela), los valores de la convolución disminuirán. Lo mismo ocurre si realizo la convolución horizontal de la imagen con una línea vertical. Para demostrar lo anterior, en la Fig.50 se muestra un ejemplo de la convolución horizontal de una línea recta vertical con un código de barra

genérico, donde en el gráfico de la derecha se muestra como los valores se elevan en las líneas blancas.



(a) Código de barra de muestra (b) Convolución horizontal con linea vertical

Figura 50: Ejemplo de convolución sobre código de barra

Comenzando con la explicación del algoritmo, en una primera instancia este permite seleccionar dos puntos de una imagen: la esquina superior izquierda y luego la esquina inferior derecha. A partir de estos puntos, se recorta el segmento de imagen que queda dentro del rectángulo imaginario creado con ambos puntos. A dicha imagen seleccionada, se le realiza la clasificación de *índice de verde*, como en los métodos *Watershed* y *Hough*.

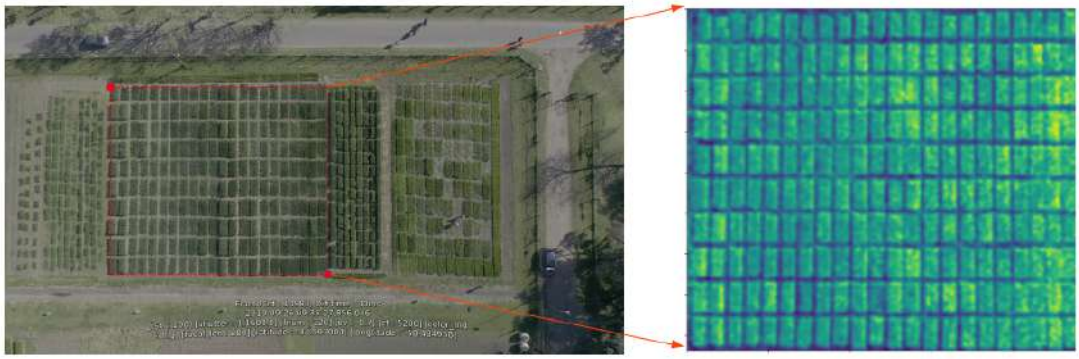
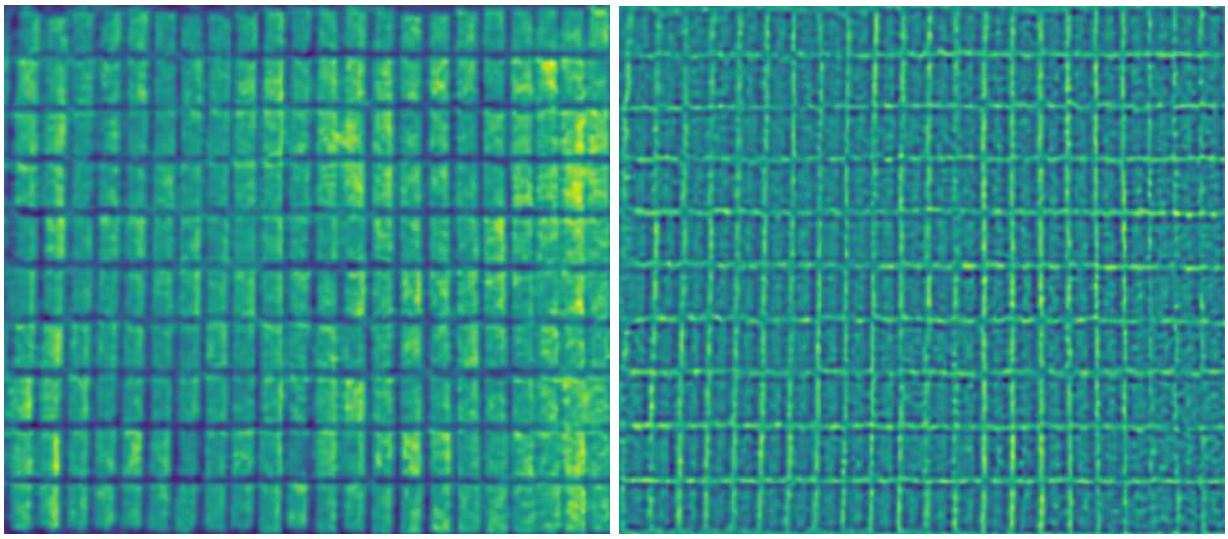


Figura 51: Selección de región de imagen

Luego, se realiza la normalización de la imagen, donde los valores de la matriz se reducen a un rango de $[0, 255]$, y se pasa la imagen por filtros Gausseanos y Laplacianos utilizando las funciones *GaussianBlur* (31) y *Laplacian* (32) , pertenecientes a la biblioteca de *OpenCV* (7)



(a) Imagen inicial

(b) Imagen normalizada con filtros Gausseanos y Laplaciano

Figura 52: Proceso de normalización y filtrado

Luego se procede a realizar la convolución. La función que se utilizó en esta etapa es *findLines*, que pertenece a la librería *imgTools*(18). A su vez, esta función presenta dos partes. En una primer parte,mediante una función denominada *_getLineImg* (perteneciente a la misma librería) se crean las lineas horizontales y verticales con las cuales se realizara la convolución. Y luego, como segunda parte, se utiliza una función *findBestDirection* , que realiza las siguientes tareas:

- Las parcelas e imágenes de los campos experimentales pueden presentar imperfecciones y esto impacta en la detección de lineas perfectamente horizontales (o verticales). Por ende, se realiza una rotación de la imagen en un rango de ángulos seleccionables (-5°a 5°por defecto) para verificar (dos pasos mas adelante) cual es el ángulo mas eficiente.
- Se realiza la convolución en ambas direcciones con todas las imágenes rotadas y se le resta la media.
- Con la ayuda de *find_peaks* (paquete de la librería **Scipy** (14)) , se encuentran los picos del gráfico de la convolución y se selecciona la imagen rotada que posee mayor intensidad de picos en promedio.

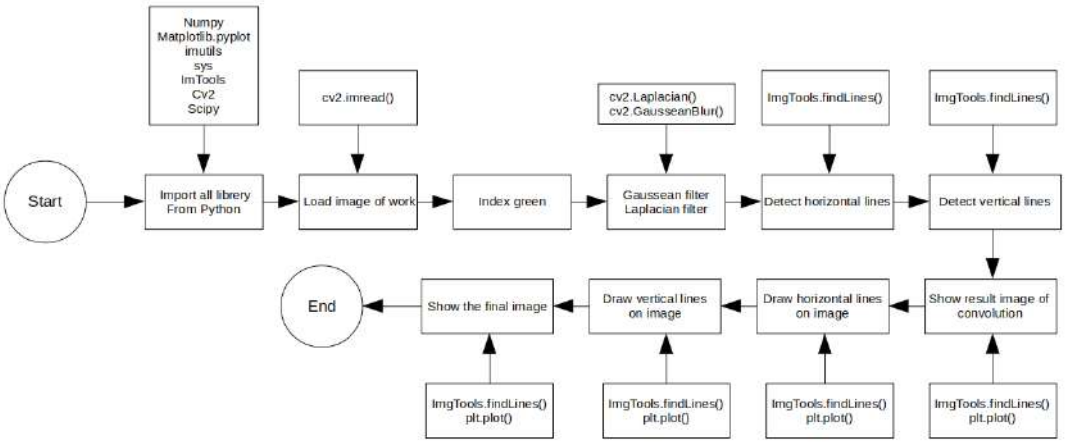
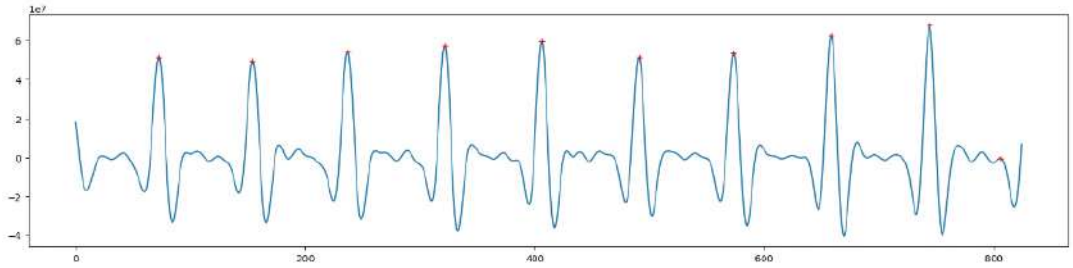


Figura 53: *Flowchart* del método de la convolución

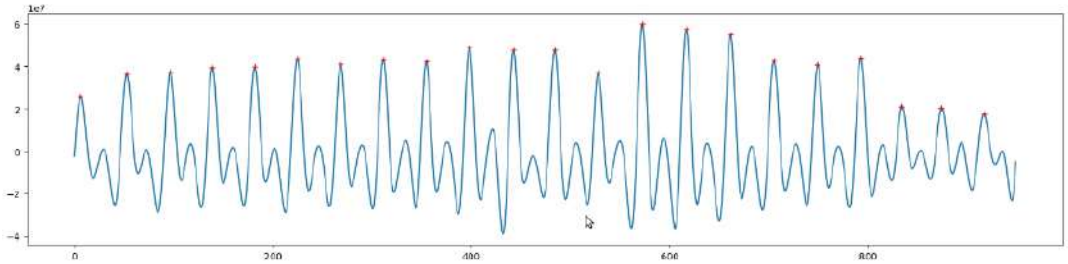
La función devuelve un diccionario con los siguientes elementos, que luego se utilizara para graficar las líneas sobre la imagen final

- **peaks:** picos de la imagen
- **convRes:** valor de la convolución
- **rotVal:** valor de la rotación en grados
- **maxIdx:** índice de mayor valor en la lista de ángulos seleccionables

En primer lugar, se visualizan los gráficos que se obtienen de realizar la convolución en ambas direcciones, en los cuales los **puntos rojos** indican que en esa posición de la imagen se detectó una línea.



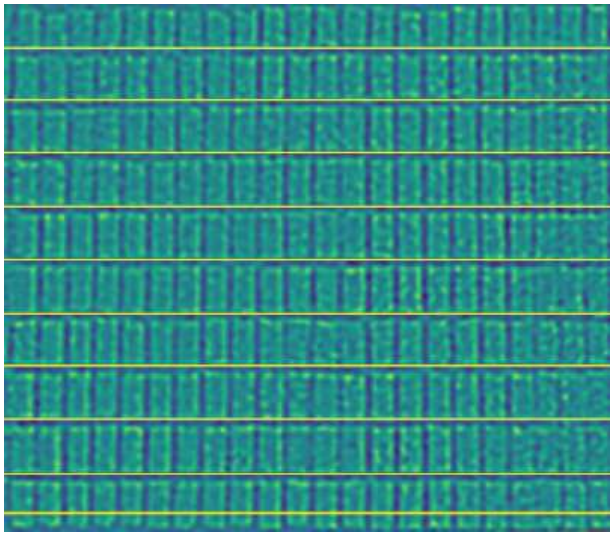
(a) Gráfico de lineas horizontales



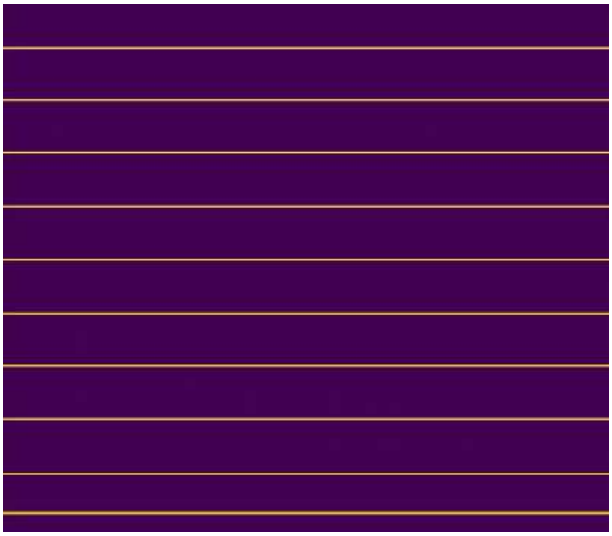
(b) Gráfico de lineas vertical

Figura 54: Gráficos de convolución

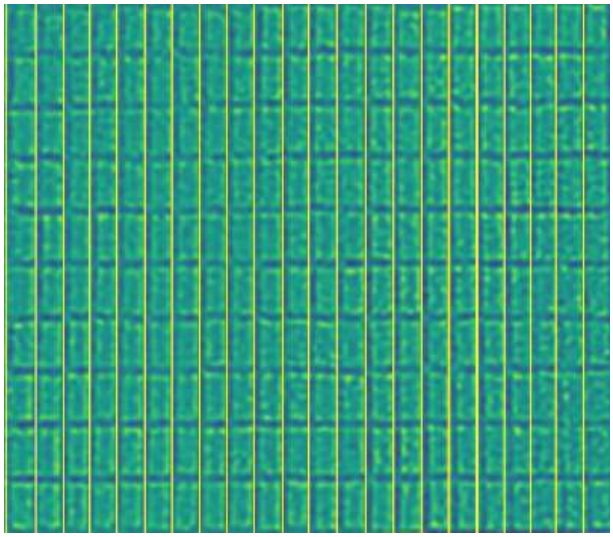
Luego, se muestran cuatro imágenes, de las cuales dos representan solo las líneas (horizontales y verticales) que se graficaron a partir de los gráficos recientes, y las imágenes restantes fusionan las líneas con la imagen original del campo experimental.



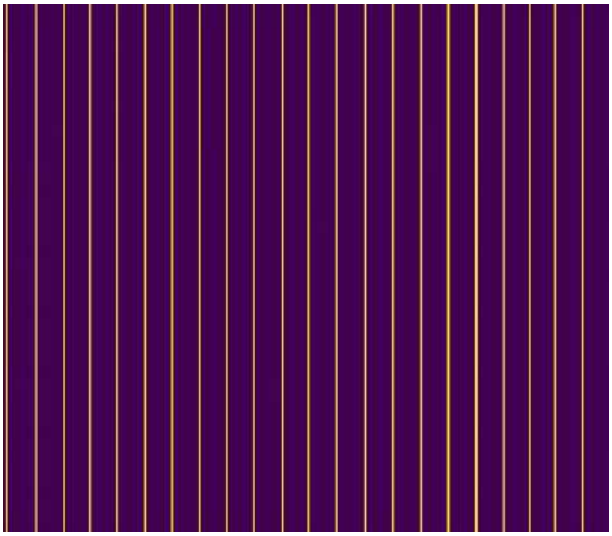
(a) Imagen de campos de cultivo



(b) Imagen de líneas horizontales



(c) Imagen de campos de cultivo



(d) Imagen de líneas verticales

Figura 55: Resultados de líneas por método de convolución

Y por último, como resultado final, se grafican las líneas en ambas direcciones sobre la imagen original.

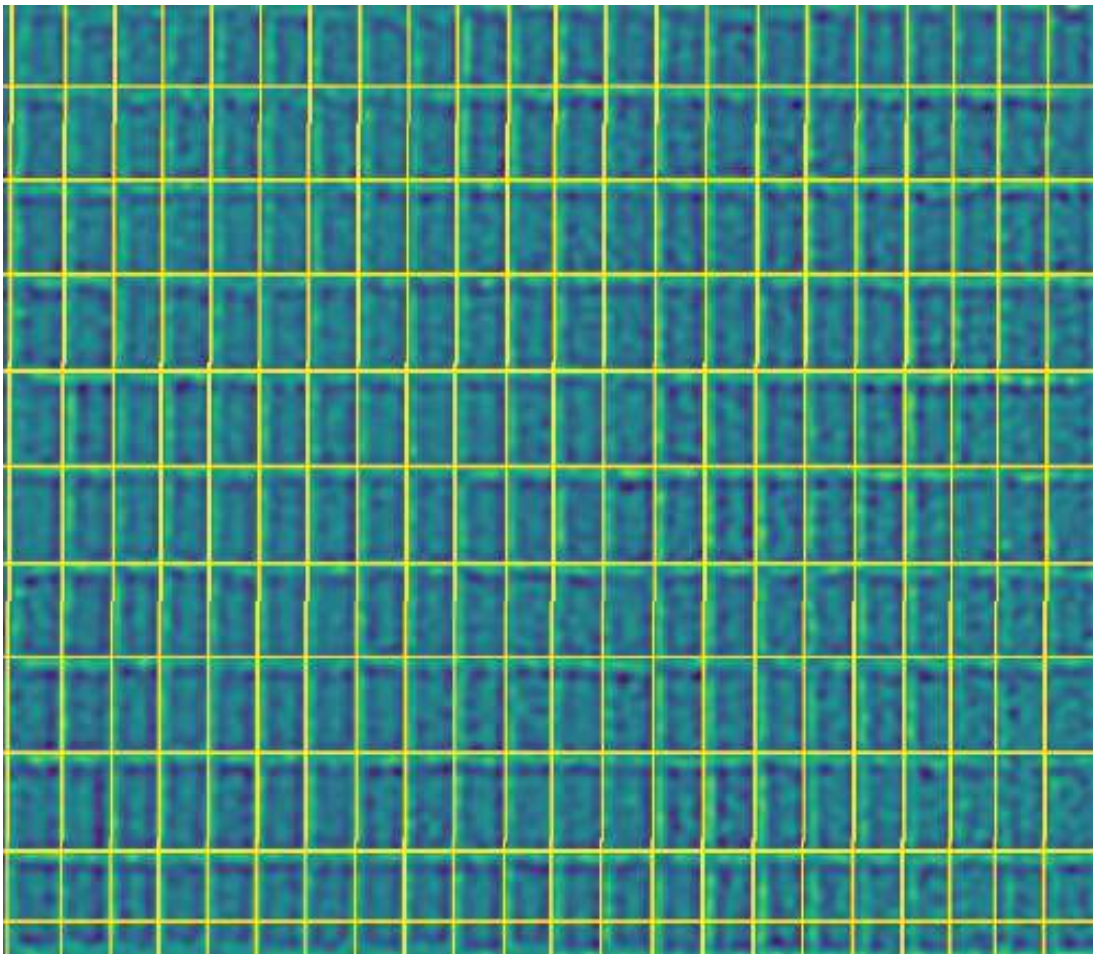


Figura 56: Imagen final con líneas marcadas

En este último método, los resultados son efectivos debido a que encuentra las rectas horizontales y verticales de forma acertada. Pero, como ventaja adicional, el algoritmo rota automáticamente las imágenes en varios ángulos (que se determinan en el algoritmo) posicionando los campos de cultivo de forma que las líneas del suelo queden en una dirección horizontal y vertical, analiza todas las imágenes localizando los picos , y luego, genera el mejor resultado.

7. Geolocalización píxeles del campo

A medida que se avanzo en el proyecto, surgió la necesidad de poder distinguir cada píxel de la imagen de manera única. La georeferenciación de cada píxel es una forma eficiente de poder identificar cada elemento de la matriz de datos que forma la imagen. Dicha identificación se realiza mediante una matriz de roto-traslación que relaciona el plano imagen (u, v) con respecto a la latitud y longitud de dicho punto en la tierra (x, y). Para encontrar la matriz que relaciona el plano imagen con el mundo real, se procedió a utilizar el método de la *Pseudoinversa de Moore-Penrose* (20). Este método consiste en una alternativa para la resolución de sistemas de ecuaciones lineales, se propone encontrar la solución del sistema de ecuaciones lineales $Y = AX$, donde A es una matriz de transformación con elementos reales de $2\,x\,4$, Y una matriz de $2\,x\,20$ que contiene datos del punto en el mundo real y X una matriz de datos del plano imagen de $4\,x\,20$. En una primera aproximación se podrían tomar 4 puntos reales del campo con su respectivas latitud y longitud para encontrar la transformación, pero para tener una mejor aproximación de la posición real de cada píxel se decidió realizar un vuelo alrededor de todo el contorno del campo experimental, para luego recoger los datos de latitud y longitud de los puntos que nos interesen. Para la obtención de esta matriz se utilizaron 20 datos de los cuales 4 pertenecen a las esquinas del campo. El sistema a resolver es el siguiente:

$$Y_{(2,20)} = A_{(2,4)} X_{(4,20)} \tag{16}$$

Matriz de transformación del plano imagen al mundo real:

$$A_{2,4} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,4} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,4} \end{pmatrix}$$

Matriz de datos del plano imagen:

$$X_{4,20} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,20} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,20} \\ \vdots & \vdots & \ddots & \vdots \\ a_{4,1} & a_{4,2} & \cdots & a_{4,20} \end{pmatrix} = \begin{pmatrix} u_1 & u_2 & \cdots & u_{20} \\ v_1 & v_2 & \cdots & v_{20} \\ u_1 v_1 & u_2 v_2 & \cdots & u_{20} v_{20} \\ 1 & 1 & \cdots & 1 \end{pmatrix}$$

Siendo el par (u, v) los puntos de interés en la imagen medidos en píxeles.

Matriz de datos del mundo real:

$$Y_{2,20} = \begin{pmatrix} a_{1,10} & a_{1,2} & \cdots & a_{1,20} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,20} \end{pmatrix} = \begin{pmatrix} lat_1 & lat_2 & \cdots & lat_{20} \\ lon_1 & lon_2 & \cdots & lon_{20} \end{pmatrix}$$

Estas medidas se encuentran expresadas en grados.

De esta manera asociamos a cada punto de la imagen, una coordenada en el mundo real para relacionarla con alguna coordenada del *set* de datos generado por el *drone*.

El método de la pseudo-inversa (20) se aplica a la matriz X de datos de la imagen, de la siguiente manera:

$$\begin{aligned} X_{pinv} &= (X^T X)^{-1} X^T \\ X_{pinv} &\in \mathbb{R}^{20,4} \end{aligned} \tag{17}$$

Al poder calcular la pseudo-inversa (20) de esta matriz, podemos aplicar las propiedades matriciales conocidas para resolver el sistema de la *ecuación 16*:

$$\begin{aligned} Y &= A * X \\ Y * X_{pinv} &= A * \underbrace{X * X_{pinv}}_{I} \end{aligned} \tag{18}$$

De esta manera la matriz A queda expresada como se muestra en la *ecuación 19*:

$$\begin{aligned} A &= Y * X_{pinv} \\ \boxed{A &= Y * (X^T X)^{-1} X^T} \\ A &\in \mathbb{R}^{2,4} \end{aligned} \tag{19}$$

La *figura 57* muestra la distribución de puntos que se utilizaron para adquirir la matriz A. Como se explica anteriormente, los puntos P_n representan un $(u, v)_n$ en el plano imagen y un $(lat, lon)_n$ en el mundo, con $n = 1, 2, ..., 20$.

Figura 57: Disposición de puntos en el campo

7.1. Proyección Geodetic de la tierra a un plano

Para poder medir distancias entre dos coordenadas del mundo, es necesario tener un sistema de georeferenciación que aproxime la distancia que hay entre un punto de origen y punto destino. Este sistema se implemento basándose en la teoría explicada en ~~el *paper* de Ulises Bussi~~ (2). El punto de origen fue propuesto sobre el extremo inferior izquierdo del campo experimental (P_{12} de la *figura 57*), y la coordenada de destino puede ser cualquier píxel de la imagen que es extrapolado a una coordenada del mundo por la matriz A. La georeferenciación se implementa de la siguiente manera:

- $a = 6378.137$ radio ecuatorial en km
- $b = 6356.7523$ radio polar en km
- lat_0 = latitud punto origen en radianes
- lon_0 = longitud punto origen en radianes
- lat_1 = latitud punto destino en radianes
- lon_1 = longitud punto destino en radianes

$$R_m = \frac{(a*b)^2}{\sqrt{((a*cos(lat_0))^2 + (b*sin(lat_0))^2)^3}} \tag{20}$$

$$R_n = \frac{(a)^2}{\sqrt{(a*cos(lat_0))^2 + (b*sin(lat_0))^2}} \tag{21}$$

Con estos parámetros, el calculo de los ΔX y ΔY se realiza de esta forma:

$$\begin{aligned} \Delta X &= \frac{((lon_1 - lon_0)*cos(lat_0)*Rn)}{sin(atan2((lon_1 - lon_0),(lat_1 - lat_0)))} \\ \Delta Y &= \frac{((lat_1 - lat_0)*Rm)}{sin(atan2((lon_1 - lon_0),(lat_1 - lat_0)))} \\ &\boxed{D = \sqrt{(\Delta X)^2 + (\Delta Y)^2}} \end{aligned} \tag{22}$$

Estas distancias se encuentran en kilómetros, pero para fines prácticos, se realizó una multiplicación por mil esta cantidad y se obtuvieron los metros de distancia entre las dos coordenadas.

Punto	Latitud	Longitud
P_0	-34.593144	-58.485319
P_1	-34.592962	-58.484660

Cuadro 7: Latitud y longitud de los puntos

Figura 58: Sistema *geodetic* entre dos puntos.

7.2. Proyección elipsoidal de la tierra a un plano

Este método es una abstracción para la superficie entre dos puntos geográficos que se utiliza para medir la distancia aproximada entre estos. Calcular la distancia entre coordenadas geográficas se basa en algún nivel de abstracción, en nuestro caso las ecuaciones elipsoidal (4), y no proporciona una distancia exacta ya que es inalcanzable dar cuenta de cada irregularidad en la superficie de la tierra. Las elevaciones del terreno no son tenidas en cuenta por esta abstracción. Si bien en nuestro campo experimental no es un problema ya que se encuentra todo sobre un mismo nivel , es un detalle valorable si se decide utilizar este método para calcular distancia entre puntos con cambio de elevación en el terreno. La validez de esta estrategia sirve para distancias que no excedan los 475 Km. Se implementa de la siguiente manera:

$$\begin{aligned} \boxed{D = \sqrt{(K_1 \Delta\phi)^2 + (K_2 \Delta\lambda)}} \\ \Delta\phi = \phi_1 - \phi_0 \\ \Delta\lambda = \lambda_1 - \lambda_0 \end{aligned} \tag{23}$$

Donde ϕ son las latitudes y λ son las longitudes expresada en radianes. D es la distancia total en Km.

$$\begin{aligned} \phi_m &= \frac{\phi_0 + \phi_1}{2} \\ K_1 &= 111{,}13209 - 0{,}56605\cos(2\phi_m) + 0{,}00120\cos(4\phi_m) \\ K_2 &= 111{,}41513\cos(\phi_m) - 0{,}09455\cos(3\phi_m) + 0{,}00012\cos(5\phi_m) \end{aligned} \tag{24}$$

Las distancias en los ejes X e Y se calculan como se muestra en la *ecuación 25*:

$$\begin{aligned} \Delta X &= (K_1 * (\Delta\phi) * \frac{\pi}{180})^2 \\ \Delta Y &= (K_2 * (\Delta\lambda) * \frac{\pi}{180})^2 \end{aligned} \tag{25}$$

Estas distancias están en Km pero las mismas, por fines prácticos, se llevan a metros como el caso anterior.

Se utilizan los mismos puntos de latitud y longitud anterior, para lograr una comparación de los métodos al final de esta sección.

7.3. Distancia por *Google Maps*

Para tener una referencia de distancia mas certera de los puntos que decidimos comparar con los métodos explicados y utilizados anteriormente, se decidió medir la distancia con la aplicación desarrollada por *Google Maps* (5). Esta aplicación lleva muchos años vigente y con actualizaciones que mejoran su rendimiento. Esto nos garantiza una cierta confiabilidad de la medida de distancia que arroja *Google* para compararla con los métodos implementados. Utilizando los puntos de la *tabla 7* , se los ubica dentro de *google maps* (5) y se usa una herramienta interna para medir la distancia.

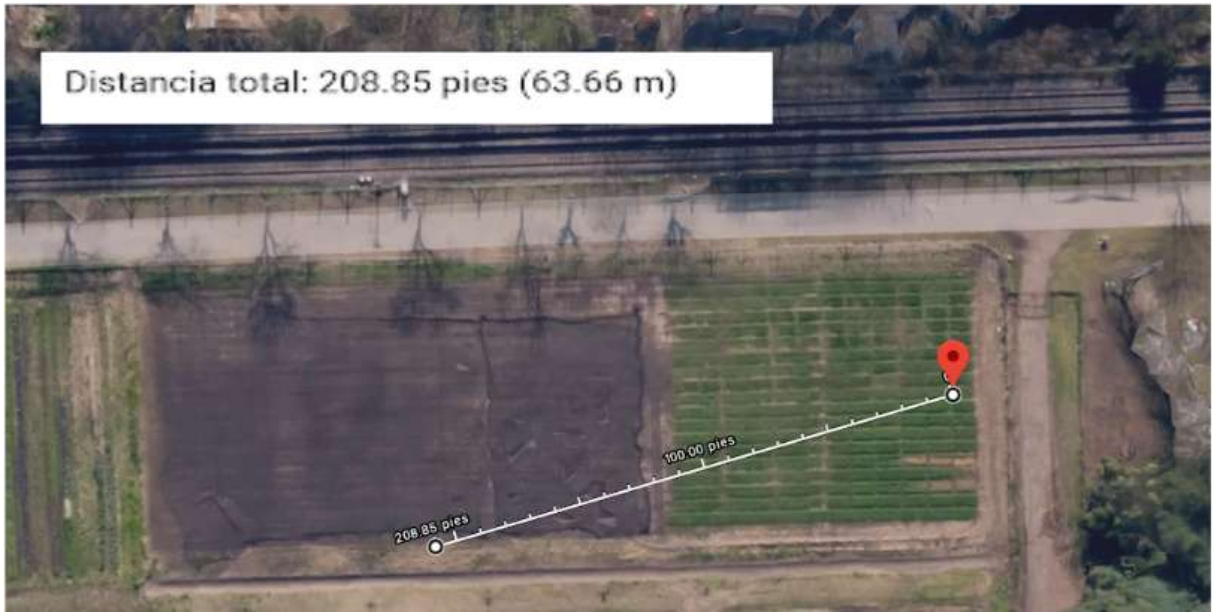


Figura 59: Medición por *google maps*

El resultado de la distancia total entre los dos puntos es de 208.85 pies, que son aproximadamente 63.66 metros.

7.4. Comparación de los métodos

La siguiente tabla muestra los resultados obtenidos por los dos métodos implementados y el resultado de la medición estimada por la aplicación *Google Maps* (5):

	Proyección Geodetic	Proyección Elipsoidal	*Google Maps*
Distancia X	63.28 m	61.45 m	-
Distancia Y	18.82 m	18.28 m	-
Distancia Total	66.02 m	64.11 m	63.66 m

Cuadro 8: Resultados de los métodos de medición de distancias referenciando 20 puntos de la imagen.

Se realizó una comparación referenciando solo las 4 esquinas de la *figura 57*. Esto generó una nuevo mapeo y una nueva matriz $A \in \mathbb{R}^{2,4}$. Utilizando los mismos métodos de medición, se obtuvieron los resultados de la *tabla* 9.

	Proyección Geodetic	Proyección Elipsoidal	*Google Maps*
Distancia X	58.87 m	57.48 m	-
Distancia Y	15.76 m	15.38 m	-
Distancia Total	60.94 m	59.50 m	63.66 m

Cuadro 9: Resultados de los métodos de medición de distancias referenciando las 4 esquinas de la imagen.

Si comparamos los resultados arrojados por *Google Maps* (5) con los métodos anteriores, ambos tienen un error menor a 5 metros, que es el error aproximado de un satélite *GPS*. En cuanto a los porcentajes de error entre generar una matriz de mapeo lineal con 4 y 20 puntos respectivamente, y comparando los resultados con *google maps* (5) es el siguiente:

Puntos	Proyección Geodetic	Proyección Elipsoidal
4	-4.27 %	-6.53 %
20	+3.70 %	+0.7 %

Cuadro 10: Porcentaje de error comparando con *google maps* y teniendo en cuenta la matriz de referenciación

Si bien los errores son bajos utilizando ambas matrices, cuando se utilizó la matriz generada por veinte puntos se obtuvieron mejores resultados. En este trabajo se decidió utilizar la matriz de transformación generada a partir de los veinte puntos.

8. Control de Crecimiento

Este algoritmo fue desarrollado con el objetivo de darle al agricultor una herramienta útil para llevar un control del proceso de crecimiento de sus cultivos. Su funcionamiento tiene como objetivo otorgarle al agricultor todas las imágenes recolectadas por el *drone* en distintos días de vuelo. El agricultor tiene como tarea elegir un punto de la imagen del cual quiera ver el progreso del cultivo, y el algoritmo buscará en todos los videos que tiene en su base de datos los *frames* que correspondan a ese punto marcado en la imagen. El desarrollo de este algoritmo se baso fuertemente en la integración de los capítulos anteriores, por eso es uno de los algoritmos mas importantes de este trabajo. La calibración de la cámara fue necesaria para poder corregir las imágenes adquiridas por el drone y tener una mayor precisión en las distancias medidas dentro de la imagen. Las condiciones de vuelo fueron utilizadas para saber con que parámetros se deben realizar los vuelos para obtener imágenes de calidad. El estudio de los *flight record* es un punto importante, ya que al volar el *drone* se genera una matriz de datos muy grande y esta tiene que saber ser interpretado para poder extraer los datos que son necesarios para el desarrollo de los algoritmos. Y por último, la geolocalización de los píxeles en la imagen ayudo a poder identificar de forma única a cada píxel y en un formato en el cual se puede relacionar con la información brindada por el *drone*.

El inicio de este algoritmo consiste en seleccionar el punto de interés en la imagen, para poder acceder a las coordenadas del mismo:

Figura 60: Punto de interés para control de crecimiento

Las coordenadas de ese punto son las siguientes:

Punto	Latitud	Longitud
$P_{interes}$	-34.592970	-58.484668

Cuadro 11: Latitud y longitud del punto de interes

Al tener estos datos, se prosigue para cargar los videos y los *datasets* de los mismos. Estos se cargan con la librería *Glob* (12) y los *datasets* se procesan con la librería *Pandas* (11). Cabe aclarar que a lo largo de nuestro trabajo se realizaron aproximadamente 102 videos del campo experimental. Una vez cargados y ordenados los videos con sus *datasets* correspondiente, se realiza un búsqueda de los *frames* que correspondan a esa coordenada en cada video. Con la librería *Pandas* (11) se buscan distintos *flags* que son de interés por el algoritmo, como por ejemplo:

- *CUSTOM.updateTime* (Fecha y hora)
- *CUSTOM.isVideo* (*Flag* de grabación)
- *OSD.latitude* (Latitud)
- *OSD.longitud* (Longitud)
- *OSD.height* [m] (Altura del *drone*)
- *GIMBAL.pitch* (Posición de la cámara)

Al tener estos *flags* podemos limitar la búsqueda del video cuando el *drone* esta grabando (*CUSTOM.isVideo = Recording*) con la cámara enfocando el suelo (*GIMBAL.pitch* = -90). Luego, se realiza una resta entre la latitud y longitud real de cada *frame* con la coordenada del punto de interés, si la resta es menor a un umbral propuesto quiere decir que el *frame* del vídeo corresponde al punto de interés seleccionado por el usuario. Esta resta se hace utilizando la librería *Numpy* (9). Con esta vinculación, podemos distinguir el número de *frame* del video y almacenar todas las imágenes que correspondan a este punto. Esto se hace posicionando el video en los números de *frames* que cumplan con este umbral, utilizando la librería de *OpenCV* (7). Con la librería *OS* (10), se crea una carpeta distinguida por fecha donde guarda todas las imágenes del punto de interés de los vídeos tomados en el mismo día. De esta manera, se puede discriminar las imágenes para distinguirlas entre diferentes días.

A continuación se mostraran las imágenes devueltas por el algoritmo. Por simplicidad solo mostraremos 3 imágenes de cada día, ya que si uno no limita el algoritmo se tienen muchas imágenes del punto seleccionado por cada video. Esto se debe a que el *drone* captura una imagen cada 33ms y según la velocidad con la que se vuela se generan muchos datos.

Figura 61: Variación de las parcelas a través de los días.

Si bien en estas imágenes puede verse fácilmente que se trata del mismo punto, mas en las imágenes del 26 de septiembre 2019 y 07 de octubre de 2019, el siguiente objetivo es poder alinear estas imágenes para saber que efectivamente se tratan del mismo lugar. Para esto se desarrollo un algoritmo de *stitching* (21), para 'cocer' o 'pegar' imágenes subsecuentes y así poder generar una imagen mas grande para su alineación. Una imagen mas grande posee mayores *features* para comparar entre si y poder lograr una matriz de transformación que lleve una imagen a la otra. A continuación se procederá a realizar una explicación de estos métodos.

8.1. Stitching

El objetivo del *stitching* (21) es poder alinear imágenes subsecuentes mediante la detección de *features* en cada imagen que correspondan a los mismos *features* en la imagen siguiente. Mediante estos es posible encontrar una matriz de homografía (16) que los relacione. Para la alineación de las imágenes utilizamos la librería *imgtools* (18), dentro de la cual se encuentra un algoritmo de *stitching* (21) que a partir del método de flujo óptico resuelto por Lucas-Kanade (15) encuentra una matriz de homografía (16).

8.1.1. Flujo óptico

El flujo óptico (15) es el patrón de movimiento aparente de los objetos de la imagen entre dos fotogramas consecutivos causado por el movimiento del objeto o la cámara. Es un campo vectorial 2D donde cada vector es un vector de desplazamiento que muestra el movimiento de los puntos del primer *frame* al segundo. Siempre se supone que este flujo óptico funciona para movimientos lentos y bajo estos supuestos :

- Las intensidades de píxel de un objeto no cambian entre fotogramas consecutivos.
- Los píxeles vecinos tienen movimiento similar.

Considere un píxel $I(x, y, t)$ en el primer *frame*. Se mueve a una distancia (dx, dy) en el siguiente *frame* tomado después del tiempo dt. Como esos píxeles son los mismos y la intensidad no cambia, podemos decir:

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{26}$$

Realizando la aproximación de la serie Taylor del lado derecho, eliminando los términos comunes y dividiendo por dt para obtener la siguiente ecuación:

$$f_x u + f_y v + f_t = 0 \tag{27}$$

donde $f_x = \frac{\partial f}{\partial x}$, $f_y = \frac{\partial f}{\partial y}$, $u = \frac{\partial x}{\partial t}$ y $v = \frac{\partial y}{\partial t}$

La *ecuación 27* se llama ecuación de flujo óptico. En ella podemos encontrar f_x y f_y, son gradientes de imagen. Del mismo modo f_t es el gradiente a lo largo del tiempo. Pero (u, v) es desconocido. No podemos resolver esta ecuación con dos variables desconocidas. Así que se proporcionan varios métodos para resolver este problema y uno de ellos es Lucas-Kanade (15).

8.1.2. Método Lucas-Kanade

El método Lucas-Kanade (15) toma un parche de 3×3 alrededor del punto por lo cual cada punto de los 9, tienen la misma moción. Podemos encontrar (f_x, f_y, f_t) para estos 9 puntos. Esto se convierte en resolver un sistema de 9 ecuaciones con dos variables desconocidas que están predeterminadas.

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{9} f_{xi}^2 & \sum_{i=1}^{9} f_{xi} f_{yi} \\ \sum_{i=1}^{9} f_{xi} f_{yi} & \sum_{i=1}^{9} f_{yi}^2 \end{pmatrix}^{-1} \begin{pmatrix} -\sum_{i=1}^{9} f_{xi} f_{ti} \\ -\sum_{i=1}^{9} f_{yi} f_{ti} \end{pmatrix} \tag{28}$$

Este algoritmo se encuentra implementado dentro de la librería de *OpenCV* (7) entonces desde el punto de vista del usuario, la idea es simple, se dan algunos puntos a rastrear y recibimos los vectores de flujo óptico de esos puntos. Estos vectores son utilizados para encontrar la matriz de Homografía (16) que relaciona las imágenes.

8.1.3. Homografía

El método de transformación de la homografía (16) se ha extendido a los mapas cuando se han utilizado técnicas fotográficas aéreas. En este tipo de levantamiento, la homografía(16) se utiliza asumiendo que el mapa es una vista en perspectiva del suelo como se muestra en la *figura 62*.

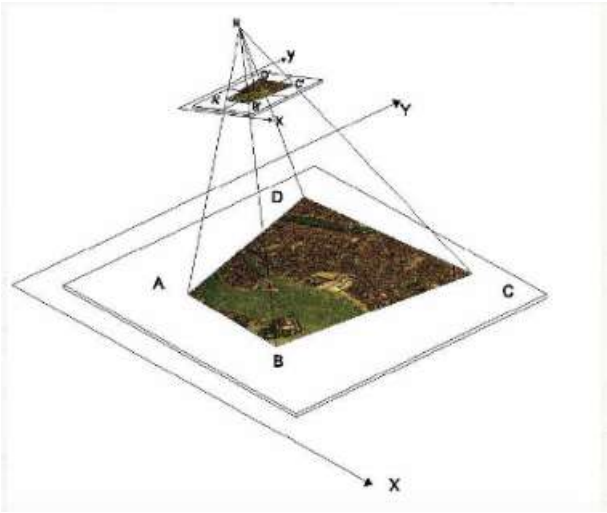


Figura 62: Ejemplo de adquisición de imágenes aéreas

La homografía esta basada en las siguientes fórmulas:

$$X = \frac{ax + by + c}{gx + hy + 1} \tag{29}$$

$$Y = \frac{dx + ey + f}{gx + hy + 1} \tag{30}$$

Donde X e Y son las coordenadas a calcular en el segundo sistema de referencia, dadas las coordenadas x-y en el primer sistema de referencia en función de 8 parámetros de transformación a, b, c, d, e, f, g, h. Entonces, teniendo estas 8 incógnitas, se requieren al menos 4 puntos conocidos en ambos sistemas. Estos puntos conocidos,requeridos para hacer la matriz de homografía(16), los obtuvimos con el resultado del método de Lucas-Kanade (15).

La librería de *OpenCV* (7) contiene un algoritmo diseñado para encontrar la matriz de Homografía (16), que utilizaremos para alinear las imágenes.

8.2. Alineación

El resultado de aplicar el algoritmo de *stitiching* (21) a las imágenes del 04-SEP-2019,26-SEP-2019 y 07-OCT-2019, para luego tratar de alinearlas entre si, es el siguiente:

Figura 63: *Stitching* punto de interés 04-SEP-2019

Figura 64: *Stitching* punto de interés 26-SEP-2019

Figura 65: *Stitching* punto de interés 07-OCT-2019

Se trato de implementar un método automático basado en lo anterior, para tratar de encontrar una matriz de Homografía (16) que relacione las imágenes pegadas y pueda realizar la alineación. Esto no es una tarea sencilla debido a que los *features* en la imagen son muy diferentes entre si, ya que las imágenes fueron tomadas con varios días de diferencia y el crecimiento fue muy notorio. Además, se intentó utilizar descriptores más fuertes como los de *SURF*(3), pero la alineación no fue satisfactoria. Se propuso realizar un algoritmo semiautomático utilizando una transformación afín (17), la cual es menos compleja que una homografía (16).

Una transformación afín (17) es una transformación lineal que contempla traslaciones, rotaciones y variaciones en la escala. Dicha transformación se puede expresar como la matriz M de 2 x 3 que se muestra en la *ecuación 31* :

$$A_{2,2} = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}$$

$$B_{2,1} = \begin{pmatrix} b_{1,1} \\ b_{1,2} \end{pmatrix}$$

$$M_{2,3} = \begin{bmatrix} A & B \end{bmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & b_{1,1} \\ a_{2,1} & a_{2,2} & b_{1,2} \end{pmatrix} \tag{31}$$

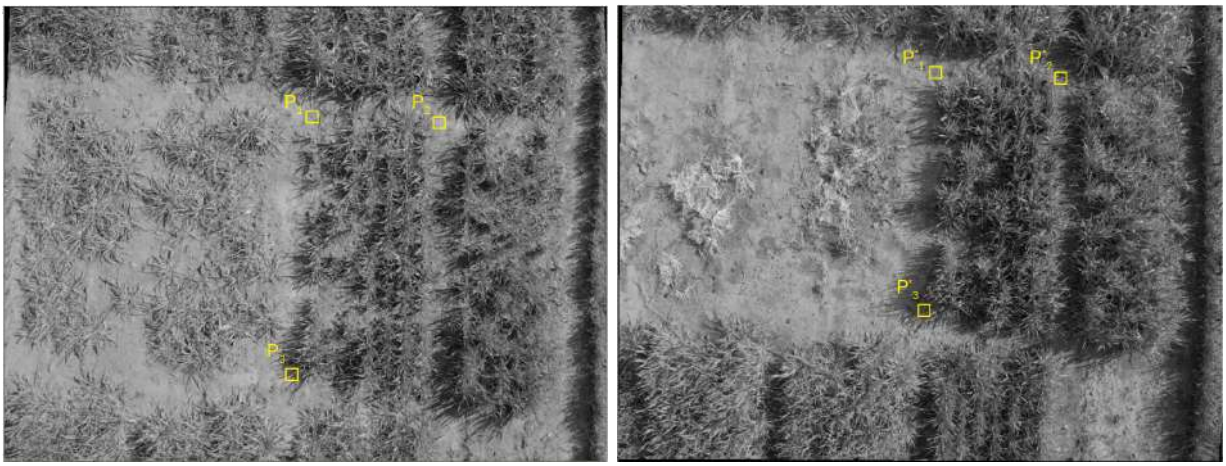
La transformación del vector 2-D $X = \begin{bmatrix} x \\ y \end{bmatrix}$, se puede escribir de la siguiente manera :

$$T = A \begin{bmatrix} x \\ y \end{bmatrix} + B \qquad o \qquad T = M \begin{bmatrix} x, y, 1 \end{bmatrix}' \tag{32}$$

Entonces la transformación se completaría de esta forma:

$$T = \begin{pmatrix} a_{1,1}x & a_{1,2}y & b_{1,1} \\ a_{2,1}x & a_{2,2}y & b_{1,2} \end{pmatrix} \tag{33}$$

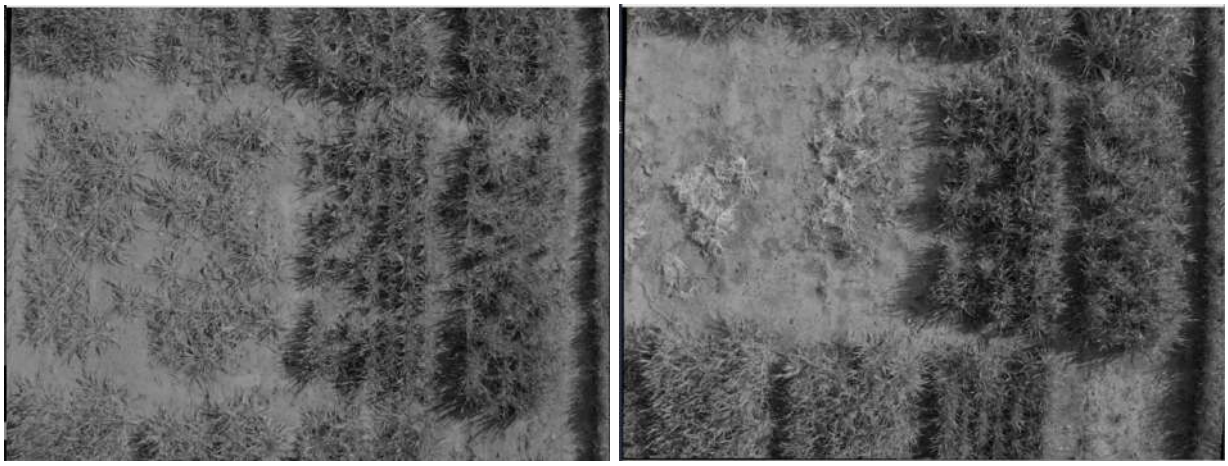
De esta manera, se propone que el agricultor que opera el programa pueda seleccionar los mismos tres puntos en las dos imágenes, de esta manera los primeros tres puntos de la primer imagen corresponderían al vector x e y y los segundos tres puntos de la segunda imagen al vector T de lo explicado anteriormente. Utilizando la librería de *OpenCV* (7), se puede encontrar la matriz M que relaciona estas imágenes. Las imágenes utilizadas son las devueltas por el algoritmo de *Stitching* (21) de los días 26-SEP-2019 y la del 07-OCT-2019.



(a) Puntos de interés *Stitching* 26-SEP-2019 (b) Puntos de interés *Stitching* 07-OCT-2019

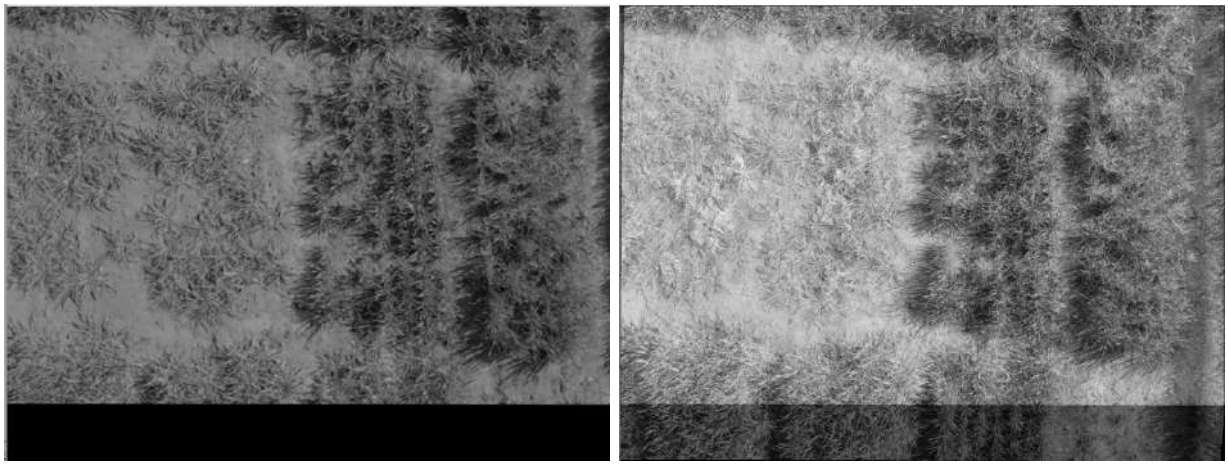
Figura 66: Alineación manual por usuario

Entonces $P1$ en la primer imagen es el punto P_1^* en la segunda y así sucesivamente. Con estos puntos se encuentra la matriz de transformación M y las imágenes se modifican de la siguiente manera:



(a) Imagen fuente (b) Imagen Destino

Figura 67: Imágenes a alinear



(a) Imagen fuente preparada para alinear (b) Imagen Alineada

Figura 68: Alineación devuelta por el algoritmo

La imagen fuente se modifica con los datos de la transformación M como se muestra en la *figura 68 a*, para ser alineada con la imagen de destino *figura 67 b*. El resultado de esta alineación se puede observar en la *figura 68 b*. De esta manera, se pueden alinear las imágenes y reconocer que efectivamente se tratan del mismo punto de interés en días diferentes.

9. Conclusiones y mejoras a futuro

9.1. Conclusiones

En el presente trabajo se implementaron varios algoritmos que facilitan el trabajo sobre campos de cultivos experimentales.

- Aprovechando la geometría de los campos de cultivos se diseñaron programas que dan herramientas para poder medir y clasificar parcelas de cultivos entre si. Esto da una gran utilidad a la hora de clasificar como fue tratada cada parcela para el crecimiento de cultivo. Los algoritmos se fueron perfeccionando a medida que se avanzó con el trabajo y los resultados fueron expuestos en cada sección.

- Otra herramienta importante para el agricultor, es el algoritmo de **Control de crecimiento** (Cap. 8). Con este algoritmo puede elegir cualquier punto de su campo y referenciarlo. Esto da una gran utilidad ya que si se adquieren imágenes del campo con cierta frecuencia, se puede utilizar este algoritmo para tener información de como fue creciendo el cultivo y a partir de esto tomar las decisiones pertinentes sobre este.

- Sin involucrar la agricultura, en el trabajo se realizó un análisis profundo de como realizar los vuelos con un cuadrotor para adquirir imágenes de alta calidad. Si bien la cámara 4K del *drone* que se utilizó es de gran ayuda, si los parámetros de vuelo no son ajustados con conocimientos las imágenes tomadas pueden contener un *blur* importante. Estos parámetros fueron determinados y puestos a prueba. Los resultados obtenidos se pudieron corroborar con los videos grabados por el *drone*.

9.2. Mejoras a futuro

- Una de las prácticas a futuro que se puede implementar para hacer un algoritmo automático seria generar un *datasets* de vídeos con más regularidad y tener imágenes del crecimiento de las plantas más detallado, es decir, vídeos de días más cercanos. De esta manera el algoritmo de **alineación 8.2** podría encontrar descriptores más fuertes entre imágenes de días diferentes y facilitarían la obtención de la matriz M de forma automática y sin la necesidad de que un usuario seleccione los puntos que coinciden en las imágenes.

- Fijar una metodología de vuelo. Esto serviría para que el *drone* realice siempre un mismo patrón de vuelo y adquiera imágenes sobre las mismas coordenadas del campo. De esta manera nos aseguramos de tener información más precisa de los puntos de interés.

- Perfeccionar y ampliar el estudio de *blur*, considerando una mayor cantidad de parámetros de la cámara, como por ejemplo el ISO y más variedades de *shutter speed*. También se podría realizar un análisis en detalle de como impacta la luminosidad del día con diferentes condiciones climáticas.

- Mejorar los algoritmos de segmentación para trabajar directamente sobre los videos tomados con el *drone* y optimizar la indexación de cada parcela con sus correspondientes coordenadas geográficas.

Bibliografía

[1] Especificaciones técnicas de DJI Mavic 2 Pro - Sitio web oficial: https://www.dji.com/mavic-2/info

[2] Ulises Bussi.(2018).Transformación mundo real a imagen,*Departamento de Ciencia y Tecnología, Universidad Nacional de Quilmes.*

[3] Corke, Peter (2017).Robotics, Vision and Control, Fundamental Algorithms in MATLAB®

[4] Ellipsoidal Earth projected to a plane :https://en.wikipedia.org/wiki/Geographical_distance

[5] Sitio web oficial de *Google Maps*: https://www.google.com.ar/maps/

[6] Sitio web oficial documentación *Python*: https://docs.python.org/3/

[7] Sitio web oficial librería *OpenCV*: https://docs.opencv.org/4.3.0/index.html

[8] Sitio web oficial librería *Matplotlib*: https://matplotlib.org/index.html

[9] Sitio web oficial librería *Numpy*: https://numpy.org

[10] Sitio web librería *OS*: https://docs.python.org/3/library/os.html

[11] Sitio web oficial de la librería *Pandas* :https://readthedocs.org/projects/pandas.

[12] Sitio web de la librería *Glob* :https://docs.python.org/3/library/glob.html

[13] Sitio web de la librería *Time* :https://docs.python.org/3/library/time.html

[14] Sitio web oficial de la librería *Scipy*: https://www.scipy.org/

[15] Algoritmo de Lucas-Kanade: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_lucas_kanade/py_lucas_kanade.html

[16] Algoritmo para encontrar Homografía: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html?highlight=findhomography

[17] Algoritmo para encontrar transformación afín: https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html

[18] Librería de visión artificial: https://github.com/ulisesbussi/imgTools

[19] Adrian Rosebrock.Blur detection with OpenCV.https://www.pyimagesearch.com/2015/09/07/blur-detection-with-opencv/

[20] Algoritmo Pseudoinversa Moore-Penrose: https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.pinv.html

[21] Algoritmo *stitching* de la librería *imgtool* (18): https://github.com/ulisesbussi/imgTools/blob/master/_utils/Stitching.py

[22] Algoritmo para ajustar elipses : https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=fitellipse

[23] Página oficial de *DJI Flight Log Viewe*: https://www.phantomhelp.com/logviewer/upload/

[24] Algoritmo para encontrar contornos :https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=drawcontours

[25] Algoritmo para dibujar contornos https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=drawcontours

[26] Algoritmo para rellenar contornos :https://docs.opencv.org/2.4/modules/core/doc/drawing_functions.html

[27] Algoritmo de *Watershead*: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_watershed/py_watershed.html

[28] Algoritmo para conectar componentes : https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html

[29] Algoritmo para dibujar líneas de *Hough*: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html

[30] Algoritmo para binarizar : https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html

[31] Algoritmo filtro *gausseano*: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html

[32] Algoritmo filtro *laplaciano*: https://docs.opencv.org/3.4/d5/db5/tutorial_laplace_operator.html