Seguimiento visual de líneas de cultivo

Proyecto Final de Carrera
Facultad de Informática
Universidad de Murcia
Septiembre 2009

Directores:
Alberto Ruiz García
Ángela Ribeiro Seijas

Alumno:
Juan Luis Barreda Sánchez

Agradecimientos:

Expresar mi más sincero agradecimiento al Instituto de Automática Industrial del CSIC y a mi tutora Ángela Ribeiro Seijas durante mi estancia allí sin cuya ayuda y apoyo este proyecto tal vez nunca hubiera sido concebido.

Índice

0. Resumen **1**

1. Introducción y marco del proyecto **2**

2. Análisis **3**

2.1. Descripción del entorno 3
2.2. Fases en el procesamiento de la imagen 6
2.2.1 Enumeración de operaciones a aplicar 7
2.3 Pre-procesamiento de la imagen 8
2.3.1 Binarización de la imagen 8
2.3.2 Supresión de información superflua 10
2.3.2.1 El algoritmo de Bresenham 11
2.4. La transformada de Hough 13
2.5. Proyección perspectiva y punto de fuga 17

3. Diseño y resolución del trabajo realizado **19**

3.1. Pre-procesamiento y segmentación en Matlab 19
3.2. Aplicación de la transformada de Hough en Matlab 25
3.2.1. Parámetros de la transformada de Hough 29
3.3. Selección de líneas óptimas usando el punto de fuga 33
3.4. Otros resultados experimentales 45

4. Alternativas de diseño **47**

5. Conclusiones y vías futuras **49**

6. Bibliografía **50**

Índice de Figuras

Figura 0: Vehículo tractor sobre el que se estudia el proyecto 2
Figura 1: Posición del dispositivo de captura de vídeo 4
Figura 2: Anclaje del dispositivo a la carrocería del tractor 4
Figura 3: Imagen capturada en condiciones óptimas 5
Figura 4: Imagen original 9
Figura 5: Imagen binarizada 9
Figura 6: Zonas de la imagen suprimidas 10
Figura 7: Resultado del algoritmo de Bresenham 11
Figura 8: Esquinas de la imagen limitadas por líneas de Bresenham 12
Figura 9: Plano xy 13
Figura 10: Espacio de parámetros 13
Figura 11: Parametrización de la línea en el plano xy 14
Figura 12: Curvas sinusoidales en el plano ρΘ 15
Figura 13: División en celdas del espacio ρΘ 15
Figura 14: Espacio de parámetros de la figura 5.(División en celdas) 16
Figura 15: Proyección perspectiva 17
Figura 16: Lineas de cosecha convergentes al punto de fuga 17
Figura 17: Mapa de puntos de corte y su correspondencia con la imagen 18
Figura 18: Segmentación con umbral 200 21
Figura 19: Segmentación con umbral 205 21
Figura 20: Segmentación con umbral 210 21
Figura 21: Imagen segmentada con esquinas sin recortar 24
Figura 22: Imagen segmentada y recortada en las esquinas 24
Figura 23: Celdas del espacio de parámetros ρΘ construido por Hough 25
Figura 24: Líneas detectadas por Hough 26
Figura 25: Zona de interés para detectar líneas (marcada en rojo) 27
Figura 26: Espacio de parámetros de Hough recortado a 40º 27
Figura 27: Líneas detectadas con ángulo incorrecto 28
Figura 28: Líneas detectadas tras aplicar el ángulo de corte 28
Figura 29: Picos detectados con valor de Threshold=0.5·max(H) 29
Figura 30: Líneas detectadas tras aplicar Houghpeaks con threshold 0.5·max(H) 30
Figura 31: Picos detectados con valor de Threshold=0.8·max(H) 30
Figura 32: Líneas detectadas tras aplicar Houghpeaks con threshold 0.8·max(H) 31
Figura 33: Resultado de usar el parámetro Fillgap 32
Figura 34: Líneas detectadas con un alto valor de FillGap 32
Figura 35: Líneas detectadas que pasan por los bordes superior e inferior 34
Figura 36: Mapa de puntos de corte de las líneas detectadas 34
Figura 37: Puntos de corte de líneas con ángulo 90º aprox. (En azul) 35
Figura 38: Puntos de corte según el ángulo de sus rectas 36
Figura 39: Puntos calculados mediante media aritmética 37
Figura 40: Mapa de puntos de corte para la figura 22 38
Figura 41: Línea que indica la posición del punto de fuga detectado 38
Figura 42: Zona de concentración de puntos de corte 39

Figura 43: Mapa de puntos de corte y líneas de cosecha 40
Figura 44: Líneas de punto de fuga ajustadas 42
Figura 45: Zona de puntos de corte inferiores de la líneas de cosecha detectadas 42
Figura 46: Puntos de corte superiores e inferiores de las líneas 43
Figura 47: Líneas de cosecha detectadas (en azul) 43
Figura 48: Resultado final de detección de líneas 44
Figura 49: Líneas detectadas incorrectamente 45
Figura 50: Imagen binarizada 45
Figura 51: Líneas detectadas por Hough 46

Índice de listados de código

Listado 1: Función segmentA de binarización de la imagen 20
Listado 2: Función myBresenham que calcula la línea de Bresenham en MATLAB 22
Listado 3: Función para recortar las esquinas de la imagen 23
Listado 4: Código para descartar las líneas cuyo ángulo sea incorrecto 26
Listado 5: Función de ajuste del punto de fuga ajustaPuntoFuga 41

0. Resumen

El uso de sistemas automáticos en el campo de la agricultura permite disminuir el tiempo de trabajo humano en la misma y reducir el gasto total que se invierte en cada plantación. Esta reducción de gastos se deduce por ejemplo de un menor consumo de herbicidas o de un menor consumo de agua de riego al optimizar el riego mediante sistemas automáticos que programen la cantidad de agua necesaria. Muchos de estos sistemas se apoyan en la visión por computador como por ejemplo sistemas que regulen la cantidad de herbicida a usar en cada parte del campo de cosecha dependiendo del nivel de malas hierbas que se encuentren en cada parte de la plantación.

El objetivo de este proyecto es diseñar un método automático de visión por computador para detectar líneas de cosecha y de este modo poder guiar a un vehículo tractor a través del campo de cosecha de forma que no cause daños mayores a la misma y más adelante poder prescindir de un agente humano que guíe el vehículo.

Se muestra la aplicación de una herramienta bastante conocida y usada en la detección de líneas como es la transformada de Hough sobre un caso real para ver que no todo está escrito en la teoría sino que al llevarlo a la práctica nos encontraremos con ciertas dificultades e inconvenientes específicas del caso sobre el que se aplica y que deberemos de solventar para alcanzar nuestro objetivo.

El método construido consiste en 2 fases principales: segmentación de la imagen y mejora de las líneas obtenidas usando las propiedades de la proyección en perspectiva. De esta forma combinaremos los datos obtenidos por la segmentación (segmentos detectados) con el punto de fuga calculado de las líneas de cosecha para obtener unas líneas que se aproximan bastante a las líneas reales.

1. Introducción y marco del proyecto

Este proyecto se ha realizado gracias a la realización de una beca de introducción a la investigación concedida por el CSIC (Consejo Superior de Investigaciones Científicas) en el Instituto de Automática Industrial donde se llevan a cabo varios proyectos e investigaciones en agricultura de precisión.

Uno de estos proyectos tiene como objetivo construir un sistema capaz de tomar la decisión de si una parte del campo de cultivo debe ser fumigada o no con herbicidas. Este sistema se implanta sobre un vehículo tractor el cual lleva acoplado en la parte delantera una cámara de vídeo que recoge las imágenes del campo de cosecha sobre las que está pasando y en la parte trasera esta equipado con una barra dispersora de herbicida diseñada por tramos de forma que se puedan activar unos tramos u otros en función de las zonas del campo que tengan más o menos densidad de malas hierbas. En la siguiente imagen se puede apreciar el vehículo tractor que se menciona.

Figura 0: Vehículo tractor sobre el que se estudia el proyecto.

Sobre este proyecto se han realizado gran cantidad de publicaciones por parte del instituto del CSIC como lo son los artículos [4], [5], [6] y [7] y que intentan hacer una exitosa tarea de fumigado selectivo de la cosecha. La idea es poder complementar este proyecto con un sistema de detección de las líneas de cosecha por las que se mueve el vehículo tractor y de esta manera conseguir que el tractor sea autónomo y no necesite de la mano humana para su guiado a través del campo de cosecha.

Todo este proyecto es posible gracias a la visión por computador y los avances que en este campo se han realizado ya que disponemos de métodos y algoritmos que nos permiten analizar las imágenes captadas por un dispositivo de vídeo para procesar y extraer la información que nos interesa.

2. Análisis

En esta sección se expondrán los objetivos que se pretenden alcanzar con este trabajo, así como dar una visión del entorno y las herramientas que se han usado en su desarrollo.

Como en cualquier sistema de visión artificial, nuestro objetivo es detectar sobre las imágenes obtenidas una propiedad que nos interese, en nuestro caso conseguir detectar las líneas de cultivo de las cosechas.

La dificultad de esta tarea depende esencialmente del grado de control que poseemos sobre las condiciones de trabajo. Para ello hemos de tener en cuenta la condiciones del entorno sobre el que se tomarán las imágenes y los dispositivos con los que trabajaremos (ver sección 2.1). Una vez obtenidas las imágenes en formato de vídeo (secuencia de imágenes) pasaremos a la fase de tratamiento de cada una de las imágenes. En esta fase trataremos de eliminar componentes de la imagen que no son necesarios, no aportan información o simplemente interfieren de forma negativa a la hora de la detección de las líneas de cosecha. Se explicarán los distintos procesamientos realizados sobre las imágenes (ver sección 2.2) para quedarnos con una representación de la misma que sea óptima a la hora de aplicar la detección de las líneas de cultivo mediante la Transformada de Hough (ver sección 2.3). Pero la aplicación de la transformada de Hough no asegura un reconocimiento perfecto de las líneas de cultivo debido a que detecta gran cantidad de segmentos y no todos los segmentos obtenidos son óptimos por ello se realiza un último procesamiento en el que se aproximará a partir de todos los segmentos obtenidos los segmentos reales de las líneas de cosecha de la imagen. Este último procesamiento consiste básicamente en aproximar de forma óptima el punto de fuga que forman las líneas de cosecha debido a las propiedades de la proyección en perspectiva(ver sección 2.4).

2.1. Descripción del entorno.

Uno de los mayores problemas en los sistemas de visión artificial son las condiciones en las que la adquisición de la imagen se realiza. La iluminación de la escena, las sombras, el calibrado del dispositivo de captura de imágenes, posibles movimientos o vibraciones del mismo son ejemplos de todas las condiciones del entorno que pueden afectar a la correcta adquisición de la imagen.

La adquisición de imágenes se realiza en este caso sobre un campo de cultivo a partir de una cámara de video instalada en el frontal superior de la cabina de un tractor. Concretamente los dispositivos usados son:

- Cámara Sony DCR PC110E
- Tractor John Deere 1140

Las secuencias de vídeo obtenidas serán a color y a 25 frames por segundo (fps).

Debido a que la cámara se encuentra sujeta a la carrocería del tractor, las imágenes obtenidas se ven afectadas por las propias vibraciones del tractor y por las vibraciones que se generan por el desplazamiento del tractor a través del campo de cosecha debido a que un campo de cosecha no es un terreno uniforme. En las siguientes imágenes se aprecia mejor la configuración del dispositivo de

captura de imágenes. Podemos ver recuadrada en rojo la posición de la cámara usada para la adquisición de imágenes:

Figura 1: Posición del dispositivo de captura de vídeo.

y en la siguiente imagen podemos ver el anclaje de la cámara a la carrocería del tractor:

Figura 2: Anclaje del dispositivo a la carrocería del tractor.

La iluminación también es un factor importante por ello las imágenes se obtuvieron en las mejores condiciones posibles de iluminación (días soleados, cuando el sol proporciona la mayor cantidad de luz y genera pocas sombras en el campo de cosecha.
En la siguiente imagen se puede ver una instantánea del campo de cosecha en condiciones óptimas.

Figura 3: Imagen capturada en condiciones óptimas.

En la imagen anterior podemos apreciar que otro factor importante para la correcta detección de las líneas de cosecha es la anchura de cada una de éstas ya que cuando la cosecha se encuentra en un estado avanzado de crecimiento las distintas líneas son más gruesas y frondosas y tienden a juntarse cada vez más debido a la perspectiva de la imagen, no llegando a distinguirse unas de otras en el tercio superior de la imagen.

Por último tenemos otro factor que son las zonas pobladas de malas hierbas entre las líneas de cosecha. Altas densidades de estas hierbas pueden hacer que a la hora de analizar la imagen se tomen como parte de las líneas de cosecha, por lo que deberán ser tenidas en cuenta a la hora de realizar la detección.

Debido a todos estos factores que deterioran la calidad de la imagen, es necesario procesar la imagen para aumentar su calidad y quedarnos con la información de la misma que más nos interesa. Estos procesamientos se explican de forma clara y precisa en el siguiente apartado.

2.2. Fases en el procesamiento de la imagen.

El procesamiento digital de imágenes es una herramienta fundamental en el campo de la visión por computador. Uno de sus objetivos es mejorar, modificar o restaurar imágenes para uso humano. En nuestro caso estamos interesados en automatizar, en la medida de lo posible, la detección de propiedades de una imagen que sean útiles para nuestro cometido que es el de detectar las líneas de cosecha.

Cuando se realiza el procesamiento de una imagen digital se pueden llevar a cabo uno o varios pasos, depende del resultado que se quiera obtener, así podemos distinguir varias etapas aplicables:

- Adquisición, como su propio nombre indica, la etapa de adquisición se encarga de capturar las imágenes de interés en un formato digital. Cabe destacar que en esta etapa de adquisición se introduce un cierto nivel de ruido en las imágenes asociado a los propios dispositivos electrónicos empleados. Además, el sistema óptico es el responsable de las posibles distorsiones geométricas y desenfoques existentes en la imagen.

- Realce. En un intento por mejorar la apariencia visual de una imagen y en ausencia de conocimiento cuantitativo de las causas de la degradación, podrán emplearse técnicas de realce. Con este término se hace referencia a un conjunto de operaciones que tienen como fin actuar sobre la calidad visual de la imagen en lo referente al brillo y contraste.

- Restauración. También se ocupa de mejorar la apariencia de una imagen pero al contrario que el realce, la restauración es objetiva, en el sentido que las técnicas usadas tienden a estar basadas en modelos matemáticos o probabilísticos de degradación de imágenes. El realce por otro lado está basado en la preferencias subjetivas que aparentan un buen resultado.

- Segmentación. Normalmente las imágenes contienen información relativa a varios objetos o entidades diferenciadas. Cuando se está interesado en procesamientos de nivel superior, casi siempre será preciso dividir u organizar la imagen en regiones de forma que cada una de ellas represente a alguno de los objetos físicos que han sido percibidos. Así, en toda etapa de análisis se hace necesario un proceso de segmentación. Nos referimos a organizar o dividir una imagen en entidades o regiones que posean significado y que guarden una estrecha relación con objetos o áreas del mundo real percibido a través de imágenes.

- Análisis. El fin último de casi cualquier aplicación en procesamiento de imágenes es la obtención de algún tipo de información a partir del análisis de los datos de imagen. La etapa de análisis normalmente vendrá precedida por una gran serie de operaciones como las mencionadas en los pasos anteriores de forma que ya no se trabaje sobre elementos de imagen como tales sino sobre representaciones simbólicas obtenidas a partir de las primeras. En otros casos, además de la detección se precisa efectuar medidas sobre los elementos a analizar. Aquí el objetivo no es simplemente mejorar la calidad visual de las imágenes sino la obtención de información cuantitativa. En esta etapa es frecuente el empleo de técnicas propias de la inteligencia artificial y reconocimiento de formas.

Aunque la distinción entre procesamiento y análisis de imágenes no es inmediata, es importante establecerla. Bajo el término de procesamiento se incluyen un conjunto de operaciones que transforman una imagen en otra. Por el contrario, en análisis de imágenes se transforma una

imagen en otra entidad diferente, es decir, se genera alguna descripción o bien se toma una decisión. No obstante, las técnicas de análisis se aplican sobre imágenes previamente procesadas.

A parte de las distintas etapas por las que puede pasar una imagen cuando es procesada se puede hablar, en términos muy genéricos, de tres tipos de operaciones a acometer sobre una imagen digital. Son las operaciones puntuales, de vecindad y geométricas.

- Operaciones puntuales o de punto, también conocidas como operaciones de transformación de los niveles de gris, se caracterizan porque cada pixel en la imagen resultante es función únicamente del nivel de gris del pixel correspondiente (el que ocupa la misma posición) de la imagen de entrada, y sólo de él.

- Operaciones de vecindad, son operaciones de procesamiento que obtienen el nivel de gris de cada pixel de la imagen transformada en función de los niveles de gris de un conjunto de pixels localizados en una cierta vecindad del pixel sometido a transformación en la imagen original. Con frecuencia a este tipo de operaciones de vecindad se les conoce también como operaciones de filtrado.

- Operaciones geométricas, involucran la modificación de los niveles de gris de cada pixel basándose en la posición del pixel en la imagen. Con estas operaciones se busca expandir o contraer una imagen, orientar una imagen según una dirección deseada, corrección geométrica de una imagen...etc.

2.2.1 Enumeración de operaciones a aplicar

En el procesamiento digital de las imágenes con las que trabajaremos en este proyecto no se realizarán todos los pasos sino sólo algunos de ellos al igual que con las operaciones descritas anteriormente.

Distinguiremos tres fases a aplicar en este proyecto:

Una primera fase de preparación de la imagen en donde aplicaremos una serie de operaciones que hagan que la siguiente fase de detección de las líneas sea bastante más fácil y computacionalmente más ligera. En esta fase aplicaremos operaciones de punto y geométricas, tratando de conseguir una imagen en donde se haya eliminado la información que resulta superflua y donde se hayan resaltado las zonas verdes o vegetales que contienen las líneas de cosecha en la imagen. Las operaciones aquí aplicadas son explicadas en la sección 2.3.

La segunda fase consiste íntegramente en la detección de las líneas presentes en las imágenes generadas por la fase anterior usando la transformada de Hough. Esta fase corresponde a la fase de segmentación en el procesamiento de imágenes digitales. En esta fase se partirá de una imagen binaria (blanco y negro) donde la parte blanca representará la zona vegetal en donde se deben detectar las líneas y se usará además una imagen de dimensiones inferiores a la original debido a las operaciones de la primera fase que consiguen que el aplicar la transformada de Hough sea un proceso más rápido y ligero. La transformada de Hough es completamente explicada en la sección 2.4.

La tercera fase consiste en mejorar los resultados obtenidos por la segunda fase (la transformada de Hough) de forma que se aplicará una sencilla técnica de apoximación del punto de fuga que forman las líneas de cosecha en la imagen con los puntos de fuga que forman las líneas detectadas entre sí. Esta última fase se explica en la sección 2.5.

2.3. Preparación de la imagen (pre-procesamiento)

En esta primera etapa transformaremos la imagen original en otra imagen más adecuada para su procesamiento por la transformada de Hough de forma que se puedan detectar el mayor número de líneas y que las líneas detectadas sean lo más cercano a la realidad posible.
Las operaciones que se van a llevar a cabo en esta etapa son:

- Binarización de la imagen
- Supresión de información superflua
  - recorte superior
  - recorte de esquinas

2.3.1 Binarización de la imagen

El objetivo del proceso de binarización es convertir la imagen de entrada RGB en una imagen en blanco y negro, donde las partes de vegetación de la imagen de entrada serán representadas como blanco y el resto como negro. Veamos en que consiste exactamente la binarización de una imagen.

Una imagen binaria es aquella en la que la escala de grises resultante del proceso de digitalización consta únicamente de dos posibles niveles.

La generación de una imagen binaria a partir de otra con una escala de grises más amplia se obtiene mediante un proceso de umbralizado. Existen diversas formas de aplicar umbralizado sobre una imagen. El caso más simple corresponde a la comparación del nivel de gris de todos y cada uno de los pixels con un único umbral U, de forma que si el pixel comparado tiene mayor valor que el umbral se le asigna un valor y si es menor otro valor.

Este tipo de operación se emplea a menudo como un primer intento por separar un objeto del fondo de la imagen, siendo la forma más elemental de segmentación aunque con unos resultados pocos satisfactorios en imágenes cuyo histograma presente un rango de grises extenso. En nuestro caso las imágenes con las que trabajamos son recogidas en color, por lo que tendrán un componente rojo, otro azul y otro verde (RGB). Esto implica que para binarizarla tenemos primero que pasarla a la escala de grises y este proceso debe de hacerse de forma que el rango de grises resultante no sea muy extenso y queden bien diferenciadas las partes vegetales.

Aunque hay varios métodos para la binarización de imágenes hemos seleccionado el método propuesto en Ribeiro[7] debido a su rendimiento. La segmentación está basada en las tres componentes RGB que describen cada punto de la imagen. La primera etapa de la binarización transforma la imagen original RGB en una imagen de niveles de gris (monocromo) obtenida mediante la aplicación de la siguiente expresión a cada pixel (i,j) de la imagen:

$$T(i,j) = r \cdot R(i,j) + g \cdot G(i,j) + b \cdot B(i,j)$$

donde r, g y b son un conjunto de coeficientes reales que deben ser seleccionados. De acuerdo a Ribeiro[7] el mejor rendimiento se alcanza con los siguientes valores r=-1, g=2 y b=-1. Si T(i,j)<=0 entonces el valor de ese pixel se deja en cero y si T(i,j)>=255 entonces se deja en 255 de forma que tendremos un rango de valores en los pixels de [0,255].

El siguiente paso es determinar el nivel umbral de gris que indique la diferencia entre los pixels que contienen vegetación y aquellos que no contienen vegetación como sombras, piedras, paja o cualquier otra cosa que no sea vegetación y entonces transformar la imagen de niveles de grises a una imagen en blanco y negro para obtener una imagen binaria. Veremos más adelante que la selección del nivel de umbral dependerá del campo de cosecha por lo que éste parámetro no debería ser fijo.

Para hacernos una idea del resultado que queremos vemos una imagen antes de ser binarizada y la misma imagen tras ser binarizada por el método anteriormente explicado.

Figura 4: Imagen original.

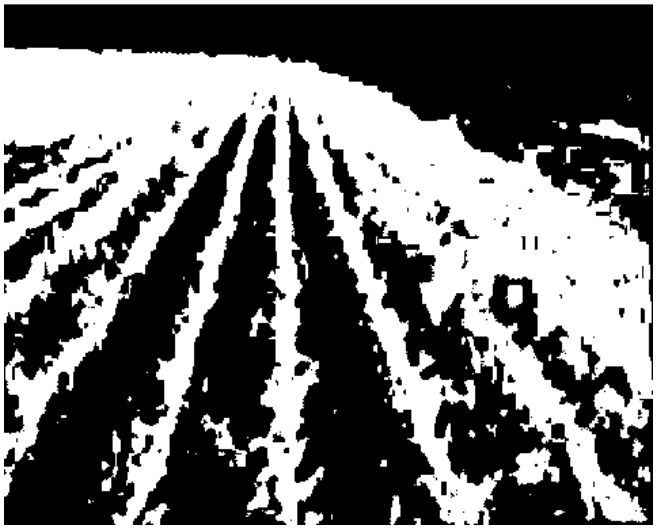

Figura 5: Imagen binarizada.

2.3.2 Supresión de información superflua

Tras binarizar la imagen se puede observar que hay varias partes de la misma que no nos interesan ya que no pueden aportar información sobre las líneas de cosecha e incluso pueden interferir de forma negativa al proceso de detección de las líneas por lo que a continuación veremos las operaciones necesarias para eliminarlas.

Las partes de la imagen que no interesan a la hora de obtener las líneas de cultivo son dos:

1. El tercio superior de la imagen ya que contiene el horizonte de la imagen y además convergen las líneas de cultivo en esa zona (debido a la proyección perspectiva) y no se pueden distinguir claramente unas de otras.
2. Las esquinas superiores de la imagen una vez eliminado el tercio superior de la imagen, ya que debido a la proyección perspectiva las líneas de cosecha situadas en esta zona están demasiado juntas y se tomarían como una sola lo cual es un error. Para eliminar las esquinas haremos uso del algoritmo de Bresenham.

En la siguiente imagen podemos apreciar las zonas de la imagen que contienen información superflua y que serán eliminadas para no tenerlas en cuenta a la hora de detectar las líneas. En rojo marcamos el tercio superior de la imagen y en naranja las esquinas superiores.

Figura 6: Zonas de la imagen suprimidas.

Ha de observase que esta eliminación de zonas de la imagen se realizará sobre la imagen de forma variable ya que dependiendo del campo de cosecha y de la inclinación de la cámara en el vehículo se podrá modificar este parámetro para ajustarse de la mejor forma posible a la realidad.

2.3.2.1 El algoritmo de Bresenham

Una imagen siempre se trata como una matriz de puntos o pixels con cierto valor asociado a cada uno de estos pixels que es lo que hace que tengan cierto color e intensidad. Por lo tanto sabemos que una imagen es representada por una matriz n x m. Esto hace que a la hora de recortar las esquinas debamos de asignarle un valor a cada punto contenido en la esquina de la imagen que indique que esa parte de la imagen no tiene ningún valor. Para ese objetivo fue anteriormente binarizada la imagen, para dejar los pixels que contienen información en blanco y los que no nos interesan dejarlos en negro. Por lo tanto los pixels que estén contenidos en las esquinas superiores de la imagen deberán ser pixels que no representen información, es decir, pixels en negro. Pero la pregunta es, ¿como seleccionar todos los pixels que conforman las esquinas de una imagen?. Para ello usaremos el algoritmo de Bresenham.

El algoritmo de Bresenham es un algoritmo que determina los puntos en un mapa de bits de n-dimensiones que deben ser trazados con el fin de formar una aproximación a una linea recta entre dos puntos dados.

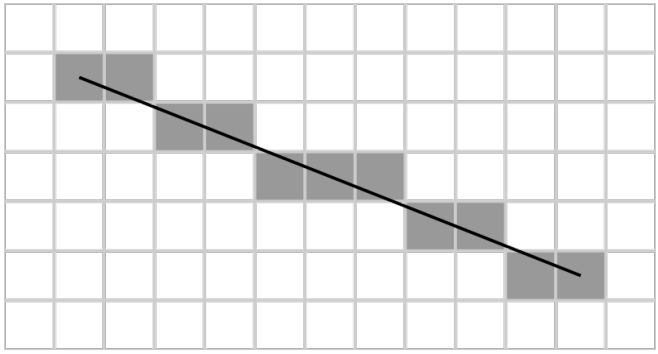

Figura 7: Resultado del algoritmo de Bresenham.

Partimos de que las coordenadas de los pixels en una imagen son coordenadas enteras y que conocemos los extremos del segmento que forma la línea siendo sus coordenadas (x_0,y_0) y (x_1,y_1). El algoritmo de Bresenham selecciona el entero 'y' correspondiente al pixel central que está más cercano del que sería calculado con fracciones y lo mismo para la coordenada 'x'. En las sucesivas columnas la coordenada 'y' puede permanecer con el mismo valor o incrementarse en cada paso a una unidad.

La ecuación general de la línea que pasa por los extremos conocidos es:

$$\frac{y - y_0}{y_1 - y_0} = \frac{x - x_0}{x_1 - x_0}.$$

Puesto que conocemos la columna, 'x', la fila 'y' del pixel se calcula redondeando esta cantidad al entero más cercano según la siguiente fórmula.

$$y = \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) + y_0.$$

La pendiente (y_1 - y_0) / (x_1 - x_0) depende sólo de las coordenadas de los extremos y puede ser previamente calculada, y el valor ideal de 'y' para los sucesivos valores enteros de 'x' se puede

calcular a partir de y_0 e ir añadiendo en varias ocasiones la pendiente.

De esta forma podremos determinar todos los pixels que forman las líneas que delimitan ambas esquinas y así recorrer los pixels incluidos en las esquinas y eliminar su información binarizándolos a negro. En la imagen siguiente se pueden ver las líneas (en verde) que delimitan las esquinas y la zona de pixels a pasar a negro (contenidos por la zona azul).

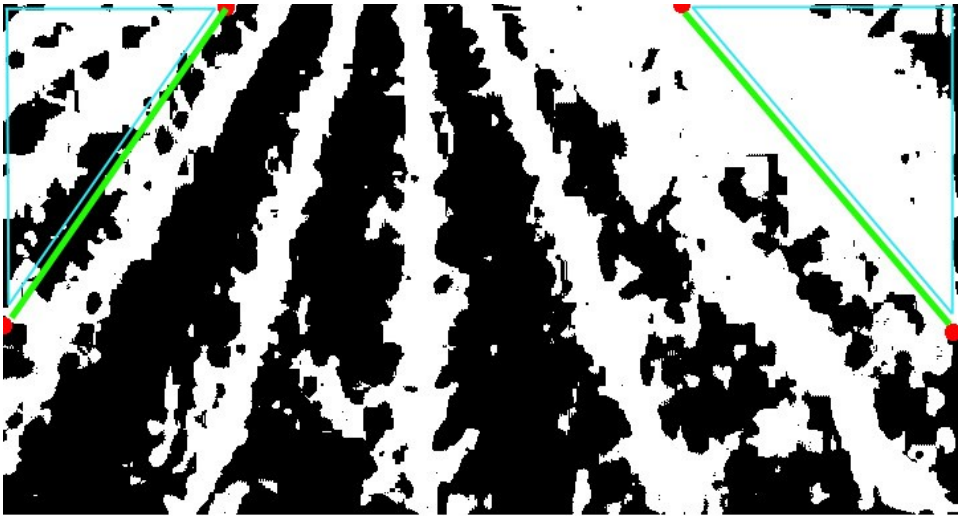

Figura 8: Esquinas de la imagen limitadas por líneas de Bresenham.

2.4. La transformada de Hough

La transformada de Hough es un método para encontrar estructuras parametrizadas(rectas, círculos, etc.) en las imágenes y que en nuestro caso concreto nos servirá para identificar las líneas que aparecen en la imagen y que son líneas de cosecha. A continuación se explica en profundidad en qué consiste la transformada de Hough.

Dados n puntos en una imagen, suponemos que queremos encontrar subconjuntos de esos puntos que se encuentran en una misma línea recta. Una posible solución sería primero encontrar todas las líneas determinadas por cada par de puntos y entonces encontrar todos los subconjuntos de puntos que están cercanos a ciertas líneas particulares.

Este método implicaría buscar $n(n-1)/2 \sim n^2$ líneas y entonces realizar $n(n(n-1))/2 \sim n^3$ comparaciones de cada punto con todas las líneas. Se puede apreciar la carga computacional que esto conlleva. Este problema puede verse de una forma diferente usando un método propuesto por Hough en 1962 y comúnmente conocido como la transformada de Hough.

Consideremos un punto (x_i,y_i) y la ecuación en forma explícita de la recta $y_i=a\cdot x_i+b$. Hay un número infinito de rectas que pasan por (x_i,y_i) . Todas ellas satisfacen la anterior ecuación al variar los valores de a y b. Sin embargo, si escribimos la anterior ecuación de esta otra forma: $b=-x_i\cdot a+y_i$ y consideramos el plano ab (también conocido como espacio paramétrico), entonces tenemos la ecuación de una simple recta para un par fijo (x_i,y_i).Además, un segundo punto (x_j,y_j) tendrá asociada también una recta en el espacio paramétrico, y esta recta intersectará con la línea asociada a (x_i,y_i) en (a',b'), donde a' es la pendiente y b' es el término independiente de la recta que contiene tanto a (x_i,y_i) como a (x_j,y_j) en el plano xy. De hecho, todos los puntos contenidos en esta recta tendrán rectas en el espacio paramétrico que intersectan en (a',b'). Estos conceptos se muestran en las siguientes figuras.

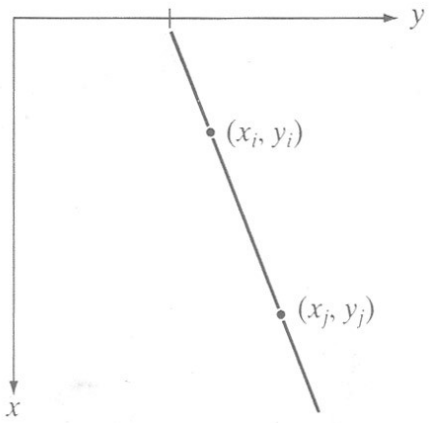

Figura 9: Plano xy.

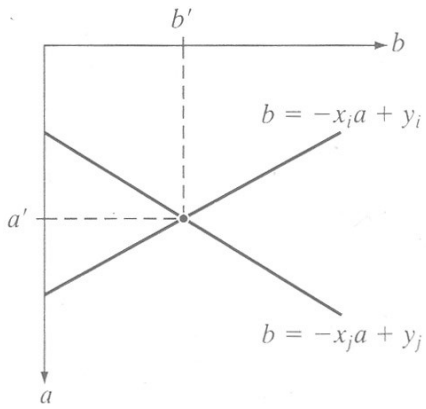

Figura 10: Espacio de parámetros.

La ventaja computacional de la transformada de Hough aparece al subdividir el espacio paramétrico en las llamadas “celdas acumuladoras”. Los intervalos $(a_{mín},a_{máx})$ y $(b_{mín},b_{máx})$ son los esperados para los valores de la pendiente (a') y del término independiente (b'). La celda de

coordenadas (i,j), con valor acumulador A(i,j), corresponde al cuadrado asociado a las coordenadas del espacio paramétrico (a_i,b_j) . Inicialmente, estas celdas están puestas a cero. Para cada punto (x_k,y_k) en el plano imagen, igualamos el parámetro "a" a cada uno de los valores de la subdivisión permitidos en el eje a, y resolvemos el valor " b " correspondiente, según la ecuación $b=-x_k \cdot a+y_k$. Los valores de "b" resultantes son aproximados en el eje b al valor permitido más cercano.

Si una elección de a_p da lugar al valor b_q, hacemos: A(p,q):=A(p,q)+1 (se incrementa de uno en uno el valor acumulador correspondiente). Así, al final del proceso un valor de M en A(i,j) corresponde a M puntos en el plano xy sobre la recta $y=a_i \cdot x+b_j$. La precisión en la linealidad de estos puntos se establece por el número de subdivisiones en el plano ab.

Si subdividimos el eje a en k partes, entonces para cada punto (x_k,y_k) obtenemos k valores de "b" que corresponden a los k posibles valores de "a". Dado que hay n puntos de imagen, esto implica $n \cdot k$ cálculos. Así, este procedimiento es lineal en "n". El producto $n \cdot k$ no se acerca al número de operaciones discutido al principio de este apartado, a no ser que k sea similar o superior a n.

Un problema que surge con el uso de la ecuación $y=a \cdot x+b$ para representar una recta es que tanto la pendiente como el término independiente tienden a infinito cuando la recta tiene una posición cercana a la vertical. Una manera de solventar esta dificultad consiste en emplear la siguiente representación de una recta: $x \cos \Theta+y \operatorname{sen} \Theta = \rho$.

El significado de los parámetros de esta ecuación se encuentra en la siguiente figura.

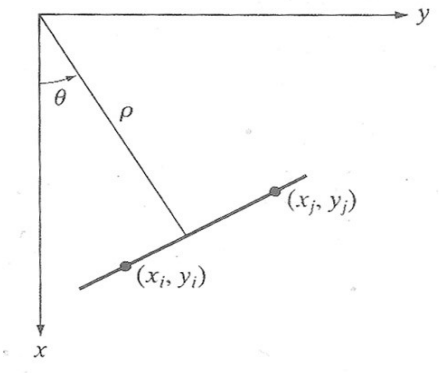

Figura 11: Parametrización de la línea en el plano xy.

El uso de esta representación en la construcción de una tabla de acumuladores es idéntico al método discutido anteriormente para la representación pendiente - término independiente. Lo que sucede ahora es que en lugar de líneas rectas, tenemos curvas sinusoidales en el plano $\rho\Theta$. Al igual que antes, M puntos de la recta $x \cdot \cos \Theta_j+y \cdot \operatorname{sen} \Theta_j= \rho_i$ darán lugar a M curvas sinusoidales que se cortan en el espacio paramétrico en (ρ_i, Θ_j) como vemos en la siguiente imagen.

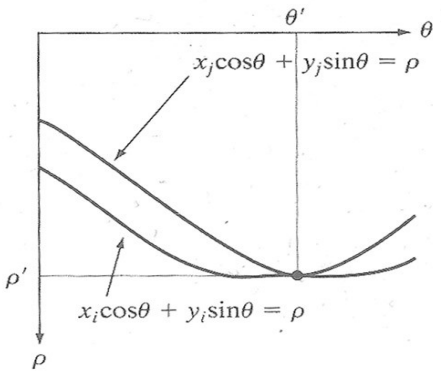

Figura 12: Curvas sinusoidales en el plano ρΘ.

Esas curvas se cortan en unos puntos que pertenecen a una celda del espacio ρΘ, y al terminar el proceso de calcular todas las curvas y sus puntos de intersección, se tendrán un conjunto de puntos de intersección en cada celda que determinarán todos los puntos pertenecientes a una misma recta en el plano xy. Podemos apreciar en la siguiente imagen una división del plano en celdas de acumulación.

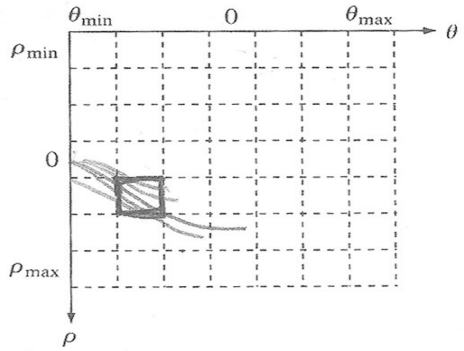

Figura 13: División en celdas del espacio ρΘ.

La complejidad de este método es linear respecto a n el número de puntos que no forman parte del fondo de la imagen en el plano xy.

En la siguiente imagen se muestra el espacio de parámetros para la imagen de la figura 5 y se puede apreciar las acumulaciones de puntos de una misma línea en las partes más blancas de la imagen. Cada recuadro verde de la imagen equivale a una gran acumulación que indica el número de puntos de la imagen que coinciden con esa línea que tiene como valor de los parámetros ρΘ específicos para esa celda acumuladora.

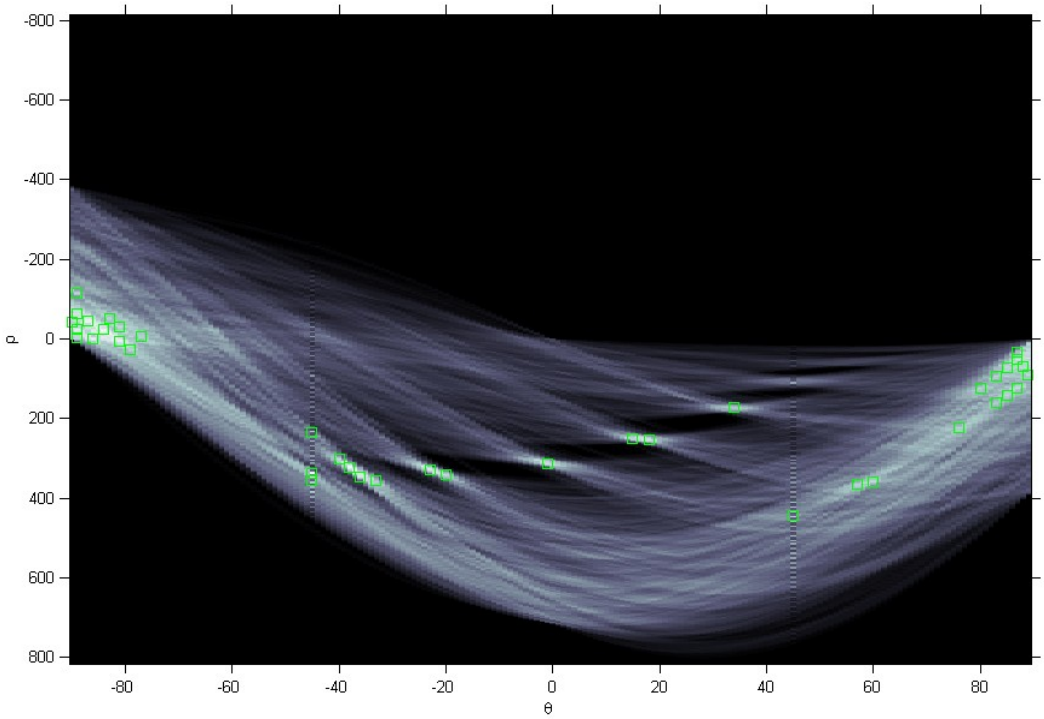

Figura 14: Espacio de parámetros de la figura 5.(División en celdas).

2.5. Proyección perspectiva y puntos de fuga

Las imágenes adquiridas por medio del dispositivo de captura de vídeo se obtienen bajo una proyección en perspectiva. Esto implica que las líneas que en realidad son paralelas, en la imagen convergen todas hacia un punto, el punto de fuga (vanishing point). En la siguiente figura se puede apreciar cómo se reflejan dos líneas paralelas en un plano de imagen en proyección perspectiva.

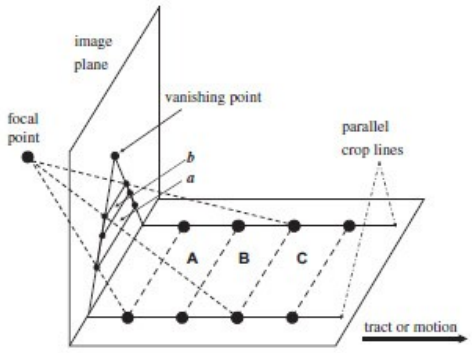

Figura 15: Proyección perspectiva.

Como se puede apreciar en la figura 15, las áreas A y B con el mismo tamaño real son proyectadas en la imagen bajo diferentes tamaños. Para cada imagen, el tamaño de las áreas de terreno proyectado es inversamente proporcional a la distancia del plano de la imagen.

Esta propiedad que se da en la proyección perspectiva nos es de gran utilidad ya que los campos de cultivo tienen líneas de cosecha que por norma general son paralelas entre ellas y en la imagen obtenida del campo se aprecian cómo esas líneas convergen a un punto en el horizonte que es el punto de fuga de las líneas de cosecha. Podemos apreciarlo en la siguiente imagen.

Figura 16: Lineas de cosecha convergentes al punto de fuga.

Usaremos esta herramienta para, una vez detectadas las líneas mediante la transformada de Hough, seleccionar las líneas que más se aproximan a las líneas del campo de cosecha usando los puntos de fuga.

El procedimiento creado para combinar la información de las líneas detectadas por Hough y el punto de fuga de la imagen y así obtener unas líneas detectadas más reales es el siguiente:

1. Para cada par de líneas detectadas calcular su punto de intersección usando la ecuación punto pendiente.

2. Dependiendo del ángulo de las rectas que hemos detectado, clasificar los puntos de corte mediante unos niveles de confianza teniendo en cuenta los ángulos de las rectas que se cortan en ese punto. Esta asignación de confianza se realiza asignando pesos a cada tipo de línea.

3. Obtener un mapa de todos los puntos de corte que nos indique la zona aproximada del punto de fuga de las líneas de cosecha.

4. Con el mapa de puntos y sus coordenadas calcularemos una aproximación al punto de fuga real de la imagen.

5. Combinamos las coordenadas del punto de fuga obtenido con las coordenadas de las líneas detectadas para obtener un segmento en cada línea de cosecha.

En el siguiente gráfico se muestra el mapa de puntos de corte de las líneas detectadas y una aproximación de cómo deberían ser las líneas detectadas.

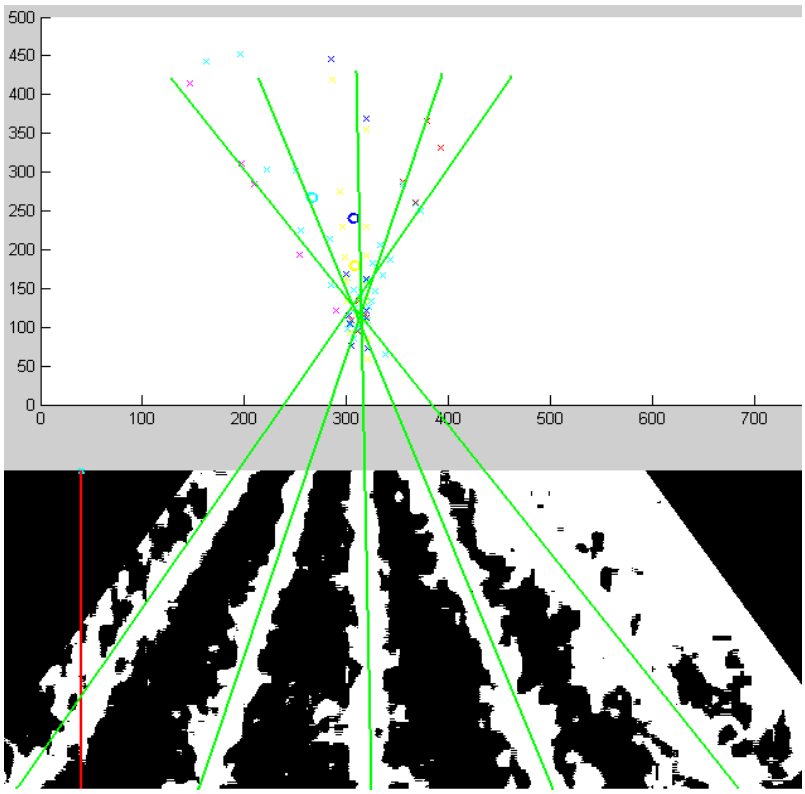

Figura 17: Mapa de puntos de corte y su correspondencia con la imagen.

3. Diseño y resolución del trabajo realizado.

En esta sección se estudiará el diseño de la solución propuesta en este trabajo. Esta solución se ha diseñado bajo el entorno de desarrollo MATLAB ya que gracias a su *Image Processing Toolbox* tenemos un método rápido y cómodo para cargar imágenes, realizar modificaciones sobre ella y aplicar algunas de las operaciones típicas en procesamiento de imágenes sin tener que programar éstas operaciones desde cero. Debemos indicar también que al estar MATLAB optimizado para trabajar con matrices obtenemos también una forma óptima para manejar las imágenes ya que éstas son representadas como matrices de pixels. Cabe decir que también se eligió este entorno porque proporciona una implementación de la transformada de Hough que se complementa con dos implementaciones más que son *houghpeaks* y *houghlines* que nos proporcionan toda la información que necesitamos para extraer la información de las líneas detectadas por la transformada de Hough.

En los distintos puntos que se van a ir desarrollando podremos ver cómo se ha ido construyendo la solución desde cero hasta alcanzar el resultado deseado y paso a paso veremos cómo se ha ido afrontando cada operación sobre las imágenes ya que nos encontraremos con una gran cantidad de parámetros variables, que pueden variar dependiendo de la imagen de entrada, y que tendremos que analizar para conseguir el resultado más preciso.

Lo primero que veremos serán las operaciones de pre-procesamiento y segmentación (sección 3.1) que se han realizado para obtener una imagen adecuada que sirva de entrada para aplicar de forma óptima la transformada de Hough (sección 3.2.1). A la hora de aplicar la transformada de Hough deberemos de tener en cuenta varios parámetros que analizaremos a fondo obteniendo unos valores bastante adecuados para alcanzar nuestro objetivo (sección 3.2.2). Una vez aplicada la transformada de Hough y optimizados los parámetros para detectar las líneas de cultivo pasaremos a aplicar un método de aproximación a las líneas reales que aparecen en la imagen que combina la información extraída por la transformada de Hough con la información que nos aporta la propiedad de la proyección en perspectiva a través del punto de fuga de todas las líneas detectadas por Hough.

3.1. Pre-procesamiento y segmentación en Matlab.

Las primeras operaciones a las que someteremos a las imágenes serán la de binarización de la imagen y de eliminación de información superflua. Para ello se ha programado sobre MATLAB varias funciones que se encargarán de realizar ese trabajo.

segmentA.m: Es una función encargada de aplicar una transformación lineal sobre los pixels de la imagen según la siguiente expresión:

$$\text{Imagen} = r \cdot A(:,:,1)+g \cdot A(:,:,2)+b \cdot A(:,:,3);$$

donde la matriz A es la imagen de entrada compuesta por las tres componentes RGB (Rojo, Verde y Azul) y las constantes r,g y b son los pesos usados para resaltar los tonos verdes (r,g,b) = (-1,2,-1). Vemos que en MATLAB podemos seleccionar la tercera dimensión de una matriz A(x,y,z) mediante el índice z siendo la dimensión del rojo la numero 1, la dimensión del verde la 2 y la dimensión del azul la 3.

Tras realizar la segmentación para generar una imagen binaria donde los pixels con tonos verdes en la imagen (las plantas de cultivo y en general la vegetación que aparece en la imagen) quedarán en blanco y el resto se quedará a negro se realiza un umbralizado para conseguir que todos los valores de la imagen se sitúen en el rango [0,255] teniendo únicamente los pixels los valores 0 (negro) o 255 (blanco).

El código MATLAB de la función segmentA.m es bastante sencillo y se muestra a continuación:

```
function im = segmentA(A,r,g,b,umb)
[f,c,x] = size(A);
temp = zeros(f,c);
A = A*255;


temp = r*A(:,:,1)+g*A(:,:,2)+b*A(:,:,3);
temp = 255-temp;
temp = temp+abs(min(min(temp)));
temp = (temp/max(max(temp)))*255;


im=temp;
if(umb==0)
  umb = mean(mean(im))-5;
  im(temp>=umb)=0;
  im(temp<umb)=255;
elseif(umb==1)
  umb = mean(mean(im))+5;
  im(temp>=umb)=0;
  im(temp<umb)=255;
else
  im(temp>=umb)=0;
  im(temp<umb)=255;
end;
```

Listado 1: Función segmentA de binarización de la imagen.

Como se puede observar en el código anterior los únicos parámetros de entrada que exige ésta función son los valores a aplicar en la transformación lineal, la imagen de entrada y el nivel de gris al cual se llevará a cabo el umbralizado.

El resultado que nos ofrece esta operación de binarización depende en gran parte del nivel final de umbralizado que se seleccione a la hora de llamar a la función segmentA y que debería variar teniendo en cuenta el tiempo de plantación de la cosecha, ya que como veremos en las siguientes imágenes variando el nivel de umbralizado entre 200, 205 y 210 las líneas de cultivo se detectarán mas o menos densas en la imagen segmentada, siendo más densas cuando el valor del umbral es mayor y menos densas cuando es menor. Podemos decir que a mayor valor del umbral se tienen en cuenta un mayor número de tonos verdes. En las siguientes imágenes vemos el efecto de este parámetro:

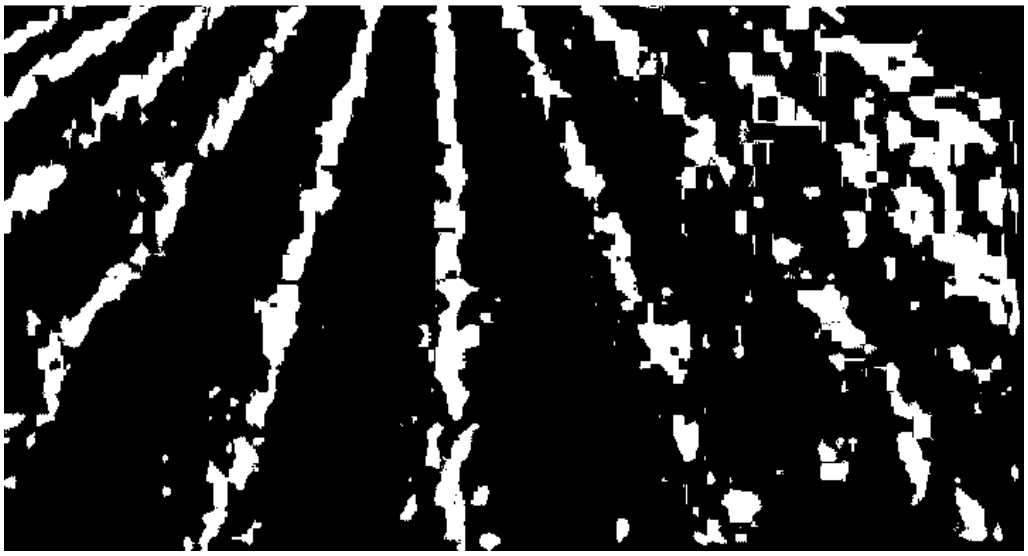

Figura 18: Segmentación con umbral 200.

Figura 19: Segmentación con umbral 205.

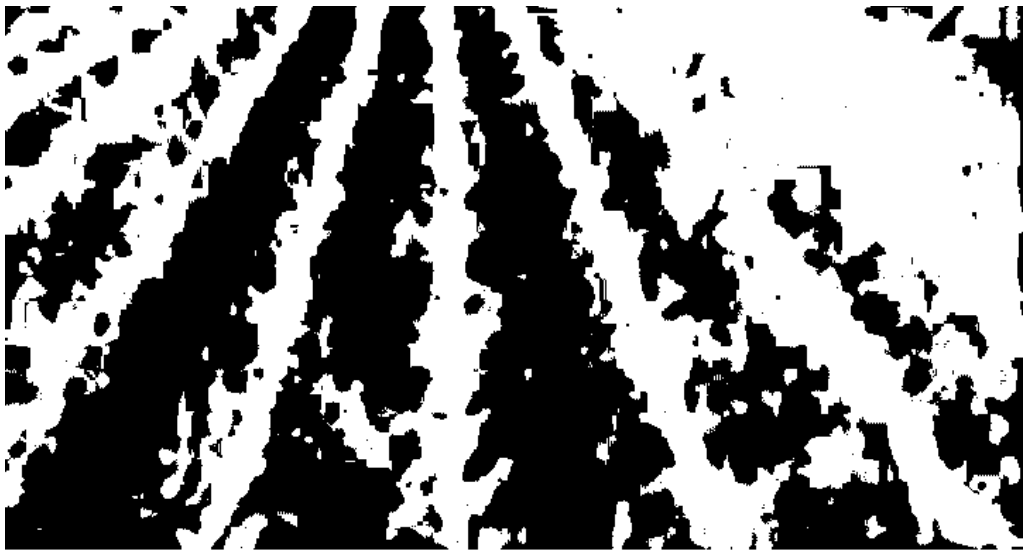

Figura 20: Segmentación con umbral 210.

A la vista de este resultado, podemos indicar que una cosecha en sus primeras fases de crecimiento necesitaría un valor de umbralizado mayor que consiga detectar la poca densidad de zona verde vegetal y una cosecha en avanzado estado de crecimiento se segmentaría mejor con un nivel de umbralizado menor ya que ésta tendría mayor densidad de vegetal.

La otra operación a realizar sobre la imagen es la de supresión de toda aquella información que no nos sirve a la hora de detectar las líneas de cosecha (tal como se vió en la sección 2.3.2.) . Una de estas operaciones es la de eliminar el tercio superior de la imagen, operación muy sencilla en la que no nos detendremos a explicar. La otra operación consiste en eliminar las esquinas debido a que por norma general influyen de forma negativa a la detección de líneas y para este cometido se realizó la función recortaEsquinas.m. Esta función se basa en otra función llamada myBresenham.m que nos devuelve el conjunto de coordenadas de todos los puntos que conforman un segmento entre dos puntos en la imagen.

La implementación del algoritmo de Bresenham se puede ver en el siguiente listado.

```
function [X,Y] = myBresenham(mycoordinates)

% BRESENHAM: Genera una línea de pixels entre dos puntos de una imagen
% [X,Y] = bresenham(mymat,mycoordinates)
%
%
% - mycoordinates son coordenadas de la forma: [x1, y1; x2, y2]
%

mycoords = mycoordinates;

x = round(mycoords(:,1));
y = round(mycoords(:,2));
steep = (abs(y(2)-y(1)) > abs(x(2)-x(1)));

if steep, [x,y] = cambia(x,y); end

if x(1)>x(2),
    [x(1),x(2)] = cambia(x(1),x(2));
    [y(1),y(2)] = cambia(y(1),y(2));
end

delx = x(2)-x(1);
dely = abs(y(2)-y(1));
error = 0;
x_n = x(1);
y_n = y(1);
if y(1) < y(2), ystep = 1; else ystep = -1; end
for n = 1:delx+1
    if steep,
        X(n) = x_n;
        Y(n) = y_n;
    else
        X(n) = y_n;
        Y(n) = x_n;
    end
    x_n = x_n + 1;
    error = error + dely;
    if bitshift(error,1) >= delx,
        y_n = y_n + ystep;
        error = error - delx;
    end
end

function [q,r] = cambia(s,t)
% function cambia
q = t; r = s;
```

Listado 2: Función myBresenham que calcula la línea de Bresenham en MATLAB.

Una vez que tenemos las coordenadas de todos los pixels que forman la línea, únicamente tendremos que recorrer todos los pixels de las esquinas para poner su valor a cero (negro). Esta operación la realizamos en la función recortaEsquinas.m como podemos ver en el siguiente listado de código.

```
function matrizImagen = recortaEsquinas(matrizImagen,rowsL,colsL,rowsR,colsR)

%Funcion para poner con valor = 0 las esquinas superiores de una matriz
%dado unas porcentajes de dimensiones de los extremos de las esquinas.
%Por defecto la dimension de filas de la esquina es de 2/3 y la de columnas
%1/5.
%
% Buscamos la línea de pixels que unen dichos puntos mediante el algoritmo
% de Bresenham para tener las coordenadas de los pixels que hacen frontera
% y poder poner a 0 a los q estan contenidos en la esquina que recorta la
% línea.
%
% La imagen que recibe es una matriz de imagen con valores entre 0 y 255.

if nargin < 2
    rowsL = 2/3;
    rowsR = 2/3;
    colsL = 1/4;
    colsR = 1/4;
end

[rows,cols] = size(matrizImagen);

%Esquina izquierda: La recta de la forma [x1, y1, x2, y2]
rectaA = [1, rows*rowsL, cols*colsL, 1];

%Esquina derecha: La recta de la forma [x1, y1, x2, y2]
rectaB = [cols-cols*colsR, 1, cols, rows*rowsR];

%Obtenemos en X e Y los pares de coordenadas de todos los puntos que forman
%la línea que delimita la esquina izquierda
[XA,YA] = myBresenham([rectaA(1), rectaA(2); rectaA(3), rectaA(4)]);

%y dwespues la linea derecha
[XB,YB] = myBresenham([rectaB(1), rectaB(2); rectaB(3), rectaB(4)]);

%Recorremos todos los puntos de la imagen que están dentro de la esquina
%definida por esas líneas y los fijamos a negro.

%Esquina izquierda
for i = 1:size(XA,2)
    matrizImagen(i,1:YA(i))=0;
end

%Esquina derecha
for i = 1:size(XB,2)
    matrizImagen(i,YB(i):cols)=0;
end
```

Listado 3: Función para recortar las esquinas de la imagen.

Como se puede apreciar en el código anterior, la función recortaEsquinas recibe 5 parámetros, uno de ellos es la imagen sobre la que se aplicará la operación y los demás parámetros marcan los puntos de inicio y fin de cada uno de los segmentos que delimitan las esquinas. Las coordenadas de estos puntos cuando no se indica su valor es de 2/3 de la imagen en los lados izquierdo y derecho de la imagen y de 1/4 de la imagen para el lado superior. Obviamente estas dimensiones que marcan la posición del segmento no deben ser fijas, es un valor que dependiendo de la anchura de las líneas de cosecha y de la separación entre las mismas puede variar por eso mismo son parámetros de la función.

El resultado que obtenemos elimina bastante información que de alguna forma u otra molestaría a la

hora de detectar las líneas. En las siguientes imágenes se puede apreciar el efecto de aplicar este recorte de las esquinas superiores sobre una imagen ya segmentada.

Figura 21: Imagen segmentada con esquinas sin recortar.

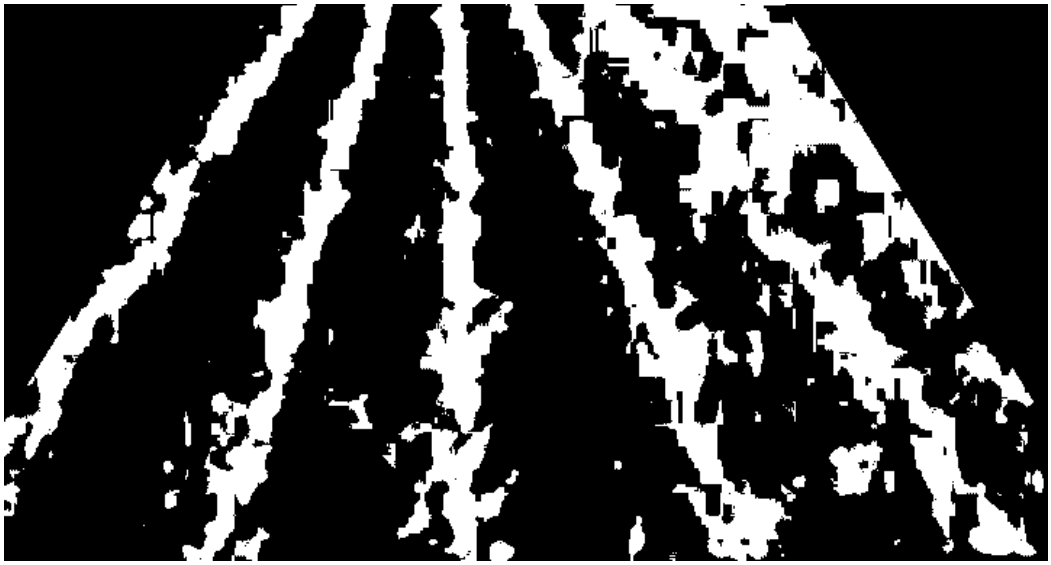

Figura 22: Imagen segmentada y recortada en las esquinas.

En la figura 21 se puede apreciar sobre la esquina superior derecha que la binarización de la imagen con umbral 205 ha hecho que la mayor parte de esa zona se tenga como información sobre zona vegetal que no tiene forma de línea de cosecha causado por el avanzado estado de crecimiento de la cosecha y la perspectiva de la cámara. Es por esa razón que debemos de eliminar esa zona conflictiva para que no interfiera a la hora de aplicar la transformada de Hough y detectar de forma óptima las líneas de cosecha.

Una vez aplicadas las operaciones de pre-procesamiento sobre la imagen el resultado es el presentado en la figura 22 que será la imagen de entrada que usaremos con la transformada de Hough.

3.2. Aplicación de la transformada de Hough en Matlab.

Gracias al toolbox de MATLAB de procesamiento de imágenes podremos hacer uso de una versión ya implementada de la transformada de Hough en la que tan sólo pasándole la imagen sobre la que queremos aplicarlo nos devolverá una tripleta de elementos sobre los que trabajaremos que son [H,theta,rho] ≡ (H,Θ,ρ). H es la matriz de la transformada de Hough y Θ(en grados) y ρ (rho) son los arrays de valores de theta y rho sobre los que se generó la matriz de la transformada de Hough (espacio de parámetros).

Analizando el contenido de la matriz H de la transformada de Hough vemos que ésta contiene en cada celda H(i,j) un valor igual o mayor a cero que contabiliza el número de puntos que están situados sobre la línea: $x \cdot \cos(\Theta_j) + y \cdot \sin(\Theta_j) = \rho_i$ de la imagen. De todas las celdas nos quedaremos con las que tengan los mayores valores lo cual significa que coinciden con las líneas que tienen más puntos y por tanto son más largas, y que serán probablemente las líneas que estamos buscando.

La imagen que tendremos de entrada se tomará al principio sin eliminar las esquinas para poder comprobar el efecto de aplicar la transformada de Hough en una imagen binarizada con la información que afecta de forma negativa a la detección de líneas provocando que se detecten una gran cantidad de líneas erróneas. A continuación veremos sobre una imagen binarizada cuáles son los resultados que se obtienen, concretamente usaremos la imagen de la figura 21.

Tras llamar a Hough mediante la línea de código *[H,theta,rho] = hough(I);* y obtener el espacio de parámetros de Hough, se llama a la función houghpeaks que nos selecciona las celdas del espacio de parámetros que mayor valor tienen. Así podemos ver en el siguiente gráfico el resultado de remarcar las celdas con mayor valor acumulado:

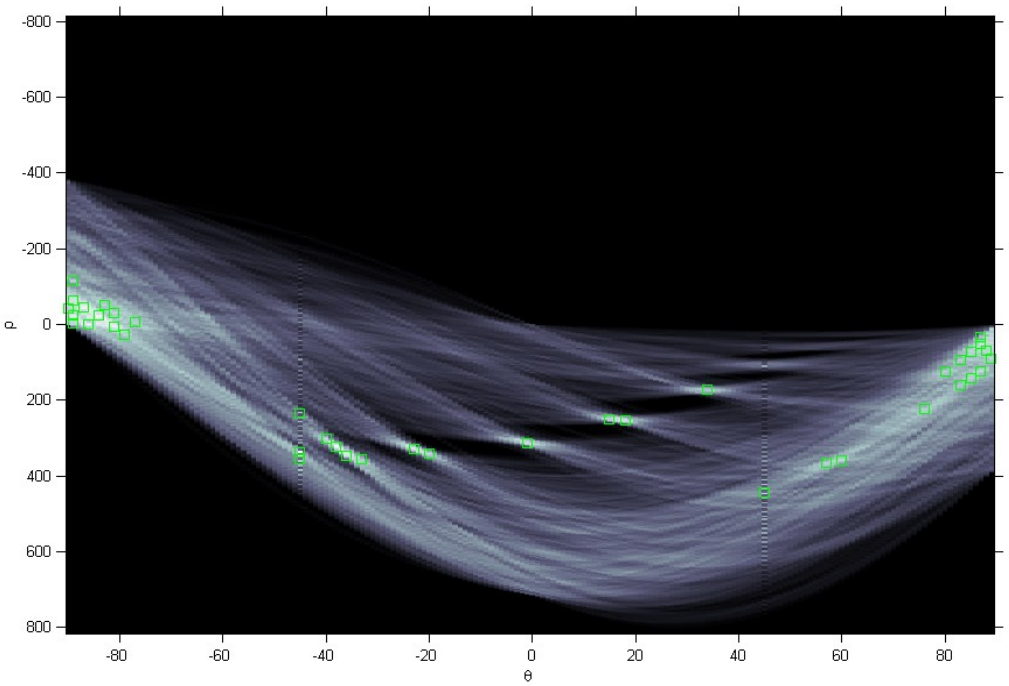

Figura 23: Celdas del espacio de parámetros ρΘ construido por Hough.

Partiendo de este espacio de parámetros detectado por Hough podemos seleccionar las líneas detectadas gracias a la función houghlines de MATLAB. El resultado tras trazar las líneas sobre la imagen es el siguiente:

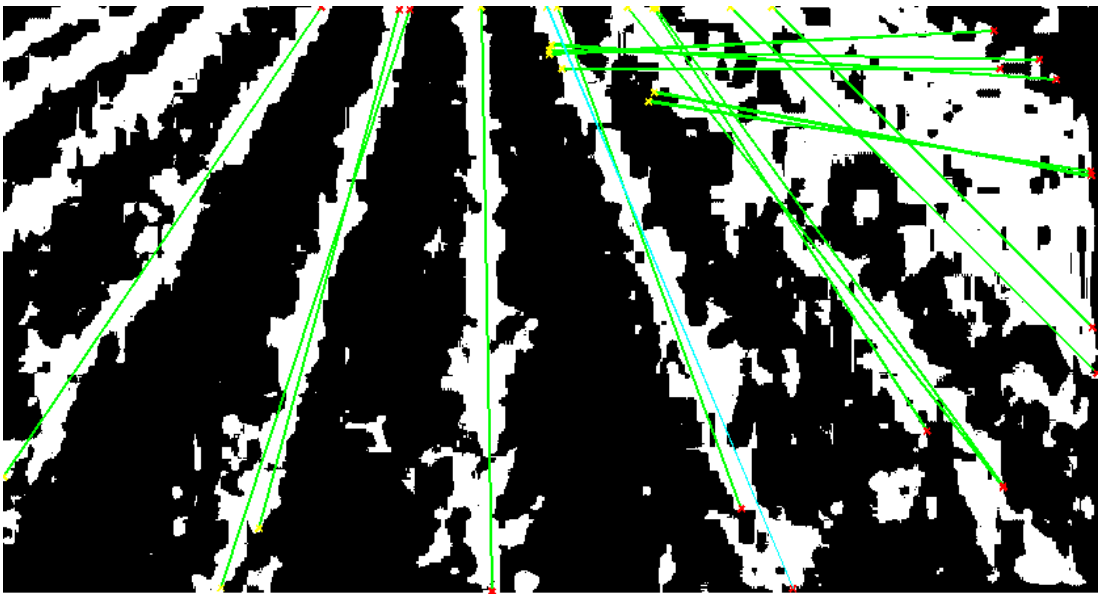

Figura 24: Líneas detectadas por Hough.

Más adelante nos centraremos en los distintos parámetros de uso de las funciones houghpeaks y houghlines que ayudarán a una mejor detección de las líneas pero ahora continuaremos viendo las propiedades de las líneas detectadas para poder comprender la relación entre éstas y el espacio de parámetros ρΘ. Concretamente estudiaremos el ángulo de las líneas detectadas para establecer un ángulo de corte que nos permita eliminar líneas detectadas de forma incorrecta.

De esta forma podremos descartar las líneas de la imagen cuya inclinación sea menor que el valor del ángulo elegido. Con esto podemos eliminar las líneas horizontales o casi horizontales de la imagen que la transformada de Hough haya detectado ya que en la imagen obtenida nos interesan las verticales o que tengan por lo general un angulo de mínimo 40º aproximadamente ya que las líneas de cultivo en la imagen tendrán esa inclinación (40º a 90º aprox).

Este valor de corte se aplica a la matriz H obtenida al calcular Hough y la forma de obtenerlo es haciendo que las celdas cuyo índice Theta (Θ) esté por debajo del ángulo de corte se pongan a cero indicando así que no hay puntos que pertenezcan a esa recta. Esta operación es sencilla de aplicar ya que simplemente hemos de poner a valor cero las posiciones de la matriz H correspondientes.

```
H(:,1:anguloCorte)=0;
H(:,180-anguloCorte:180)=0;
```

Listado 4: Código para descartar las líneas cuyo ángulo sea incorrecto.

Como se ve en la siguiente figura, se ha reducido bastante el espacio de parámetros de Hough que contienen posibles rectas quedándonos con las rectas más verticales señaladas en el recuadro rojo:

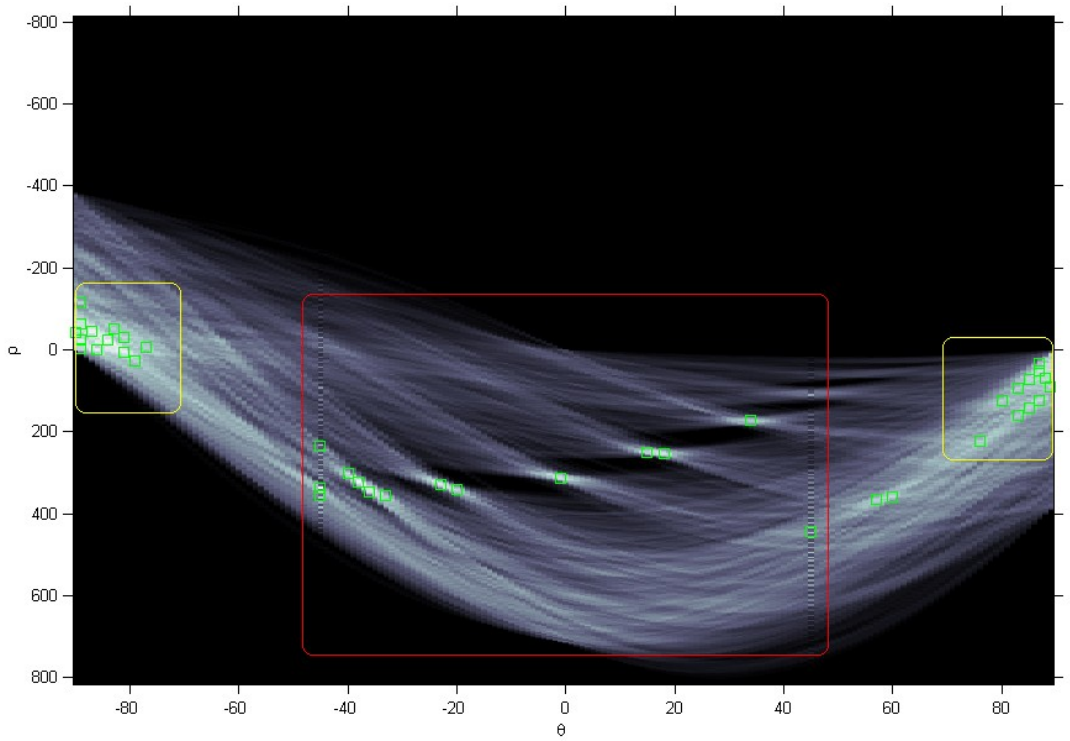

Figura 25: Zona de interés para detectar líneas (marcada en rojo).

y en la siguiente figura podemos observar el espacio de parámetros de Hough una vez que se ha eliminado la zona cuyo ángulo no nos interesa.

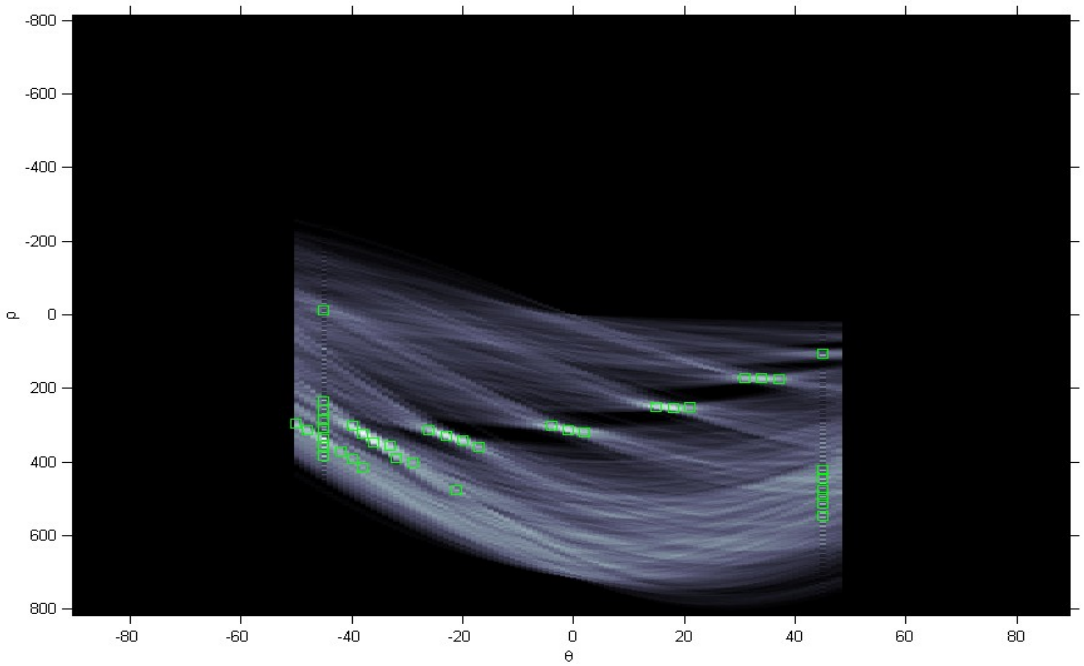

Figura 26: Espacio de parámetros de Hough recortado a 40º.

Y como resultado final en la imagen ya no se detectan líneas cuya inclinación esté por debajo del ángulo de corte y que antes sí se detectaban como en la imagen siguiente, que están indicadas dentro del recuadro rojo.

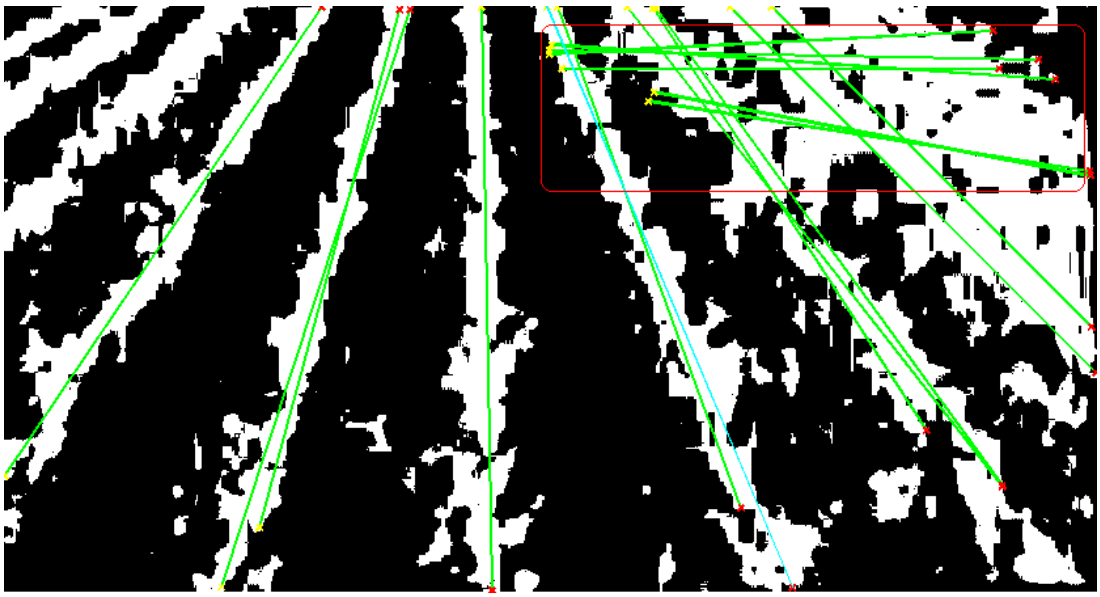

Figura 27: Líneas detectadas con ángulo incorrecto.

Y éste es el resultado de las líneas que ahora detecta después de aplicar el ángulo de corte.

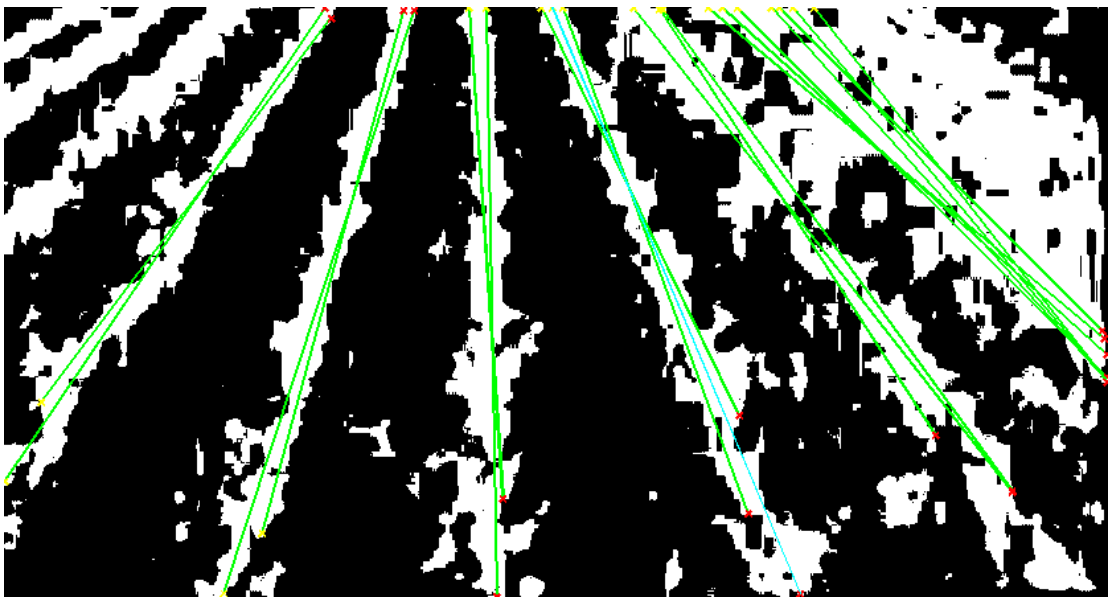

Figura 28: Líneas detectadas tras aplicar el ángulo de corte.

Como se puede observar ahora es menos probable detectar líneas que no correspondan a líneas de cosecha después de aplicar el ángulo de corte y a la vez se obtienen más líneas pertenecientes a las propias líneas de cosecha. La respuesta a por qué se detectan líneas nuevas en la

figura 27 que antes de aplicar el ángulo de corte no se detectaban se explica viendo el funcionamiento de los distintos parámetros de las funciones houghpeaks y houghlines que veremos a continuación.

3.2.1. Parámetros de la transformada de Hough.

Una vez construido el espacio de parámetros ρΘ mediante la llamada a la función Hough de MATLAB ya tenemos toda la información sobre las líneas detectadas en la imagen, lo único que hay que hacer es extraerla mediante las funciones houghpeaks y houghlines. Estas funciones serán las encargadas de interpretar toda la información del espacio de parámetros en base a los parámetros que reciben para detectar las líneas que se aproximan a las líneas de cosecha reales.

Comenzaremos con la funcion houghpeaks, encargada de identificar los picos de la transformada de Hough. Con 'picos' nos referimos a las celdas acumuladoras del espacio de parámetros ρΘ que mayor valor poseen. Esta función recibirá 3 parámetros como sigue:

Houghpeaks(H, Numpeaks, 'threshold', valueThreshold)

- *H*, es el primer parámetro y contiene la matriz de la transformada de Hough, es decir, el espacio de parámetros ρΘ.

- *Numpeaks*, indica el número de picos de mayor valor que serán seleccionados. Cuanto mayor sea este valor se detectarán un mayor número de líneas.

- *'threshold' y valueThreshold* indican que se seguirá un valor umbral que se especifica en el parámetro valueThreshold. Equivale a un umbral que indica el valor a partir del cual los valores de H se consideran altos y por lo tanto posibles líneas de interés. Por defecto se tendrán en cuenta todas las líneas cuya longitud en puntos sea como mínimo la mitad de la línea más larga detectada. A continuación se muestra el efecto de este parámetro con valor 0.5·max(H) y otra con 0.8·max(H).

En la siguiente imagen se muestra el espacio de Hough con ángulo de corte aplicado y con un Threshold con coeficiente 0.5·max(H):

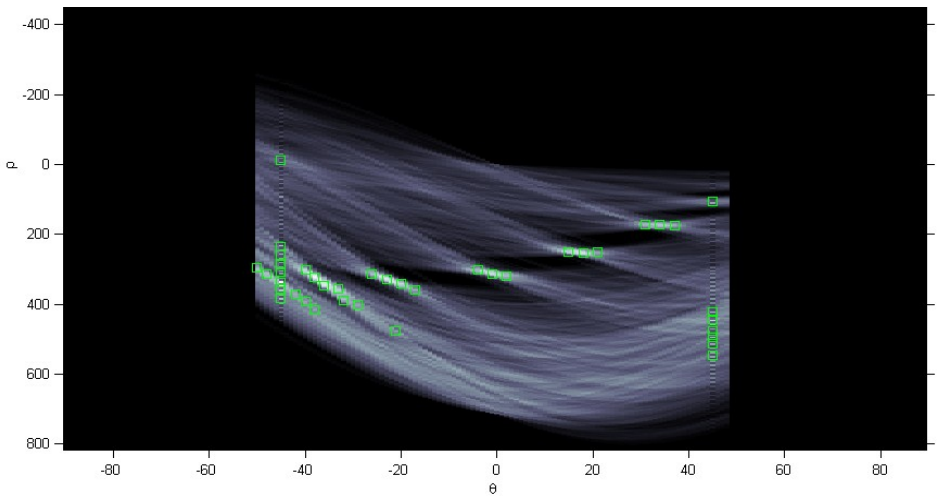

Figura 29: Picos detectados con valor de Threshold=0.5·max(H)

y como resultado se obtienen las siguientes líneas.

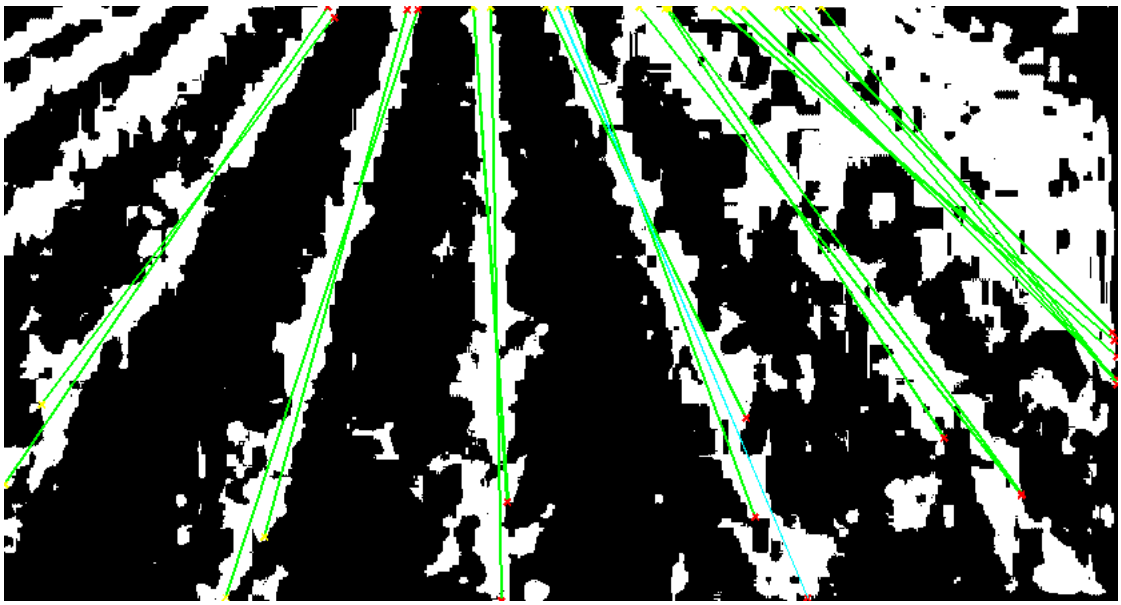

Figura 30: Líneas detectadas tras aplicar Houghpeaks con threshold 0.5·max(H)

si ahora aplicamos un Threshold de 0.8·max(H) significa que sólo nos quedaremos con las líneas que tengan como mínimo una longitud del 80% de la línea más larga detectada. En el espacio de Hough se localizarían menos celdas que cumplan con este requisito:

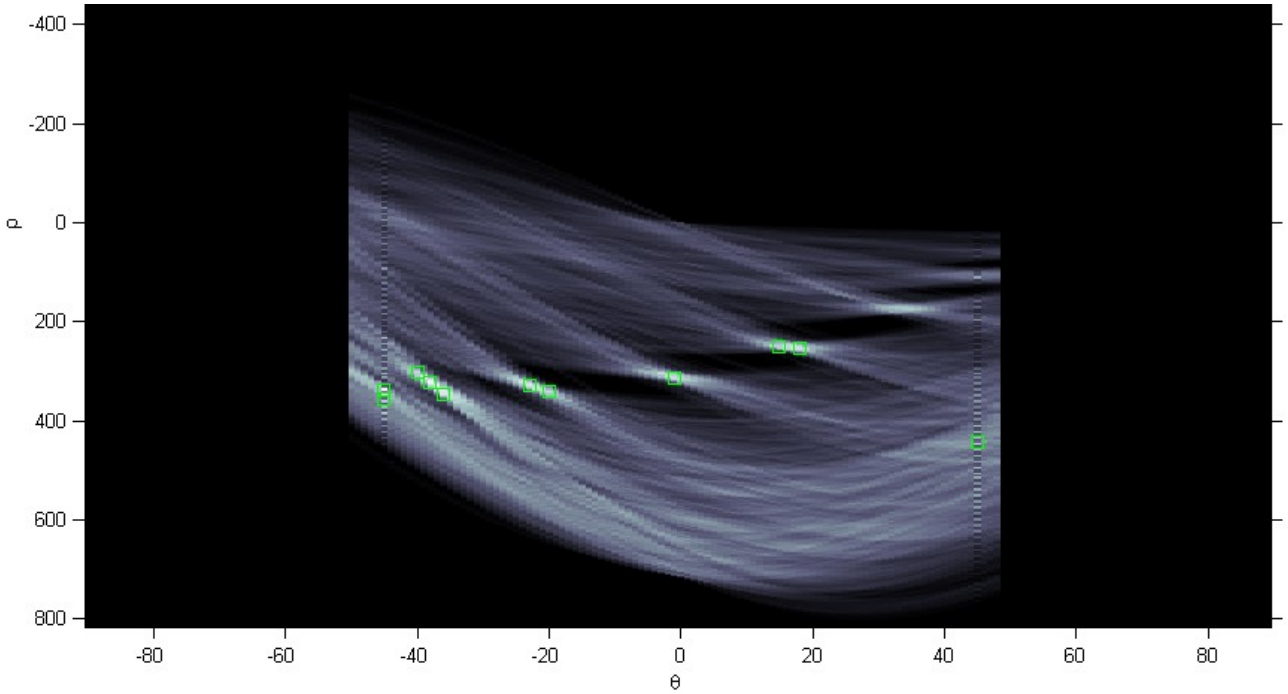

Figura 31: Picos detectados con valor de Threshold=0.8·max(H)

Y al seleccionar un número menor de celdas, también se detectarán un número menor de

líneas que en la figura 29 usando un valor de Threshold menor. En la siguiente figura se puede notar que el número de líneas detectadas con valor Threshold = 0.8·max(H) es sensiblemente inferior.

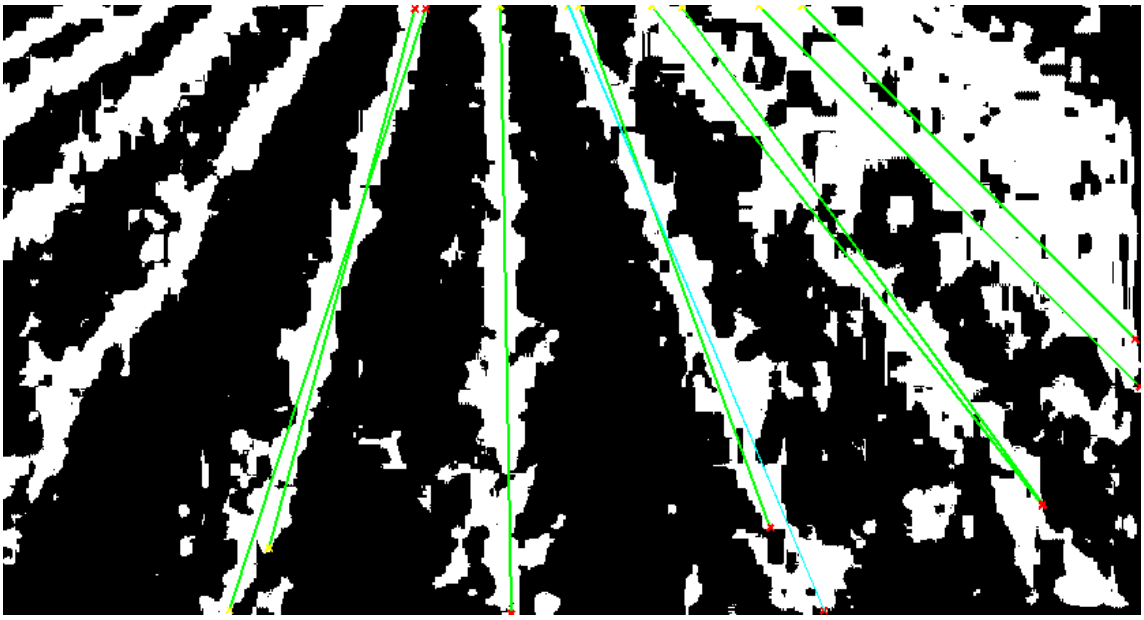

Figura 32: Líneas detectadas tras aplicar Houghpeaks con threshold 0.8·max(H)

Como hemos visto con la función Houghpeaks, podemos seleccionar la cantidad de segmentos a detectar y la longitud de los segmentos detectados en base al segmento de mayor longitud detectado. Ahora, en la función HoughLines también podremos introducir más parámetros que mejoren la detección de los segmentos.

La función HoughLines es la encargada de extraer los segmentos de las líneas basandose en la transformada de Hough. Esta función recibe seis parámetros al ser llamada:

HoughLines(IB, theta, rho, peaks, 'fillgap', valueFillgap, 'minLength', valueMinLength)

- *IB*, es la imagen sobre la que se detectarán los segmentos, que ha sido previamente binarizada y en donde se ha suprimido la información no válida.

- *Theta*, es el array de valores Θ (en grados) que devuelve la función Hough y que indica los valores de Θ sobre los que fué generada la matriz H de la transformada de Hough.

- *Rho*, es el array de valores ρ que devuelve la función Hough y que indica los valores de ρ sobre los que fué generada la matriz H de la transformada de Hough.

- *Peaks*, es una matriz que genera la función hougpeaks y que contiene las coordenadas en fila y columna de las celdas de la transformada de Hough usadas para la búsqueda de los segmentos.

- *'fillgap'* y *valueFillgap*. El primero indica que el siguiente parámetro especificará un valor de Fillgap. ValueFillgap es un valor positivo que especifica la distancia entre dos segmentos de líneas. Cuando la distancia entre los dos segmentos es menor que el valor especificado, se considera esos segmentos como una misma línea. Su valor por defecto está puesto para que dos segmentos separados por una vigésima parte de los 2/3 de la altura de la imagen (rows)

se considere el mismo segmento. Es un valor de distancia entre segmentos pequeño porque se quiere evitar unir segmentos cuya separación sea muy grande y que correspondan a distintas líneas de cultivo. En resumen, con este parámetro se consiguen rellenar huecos en negro que se encuentren en una misma línea de cultivo y que se hayan generado tras la segmentación. En la siguiente imagen se comprende mejor este parámetro visto gráficamente.

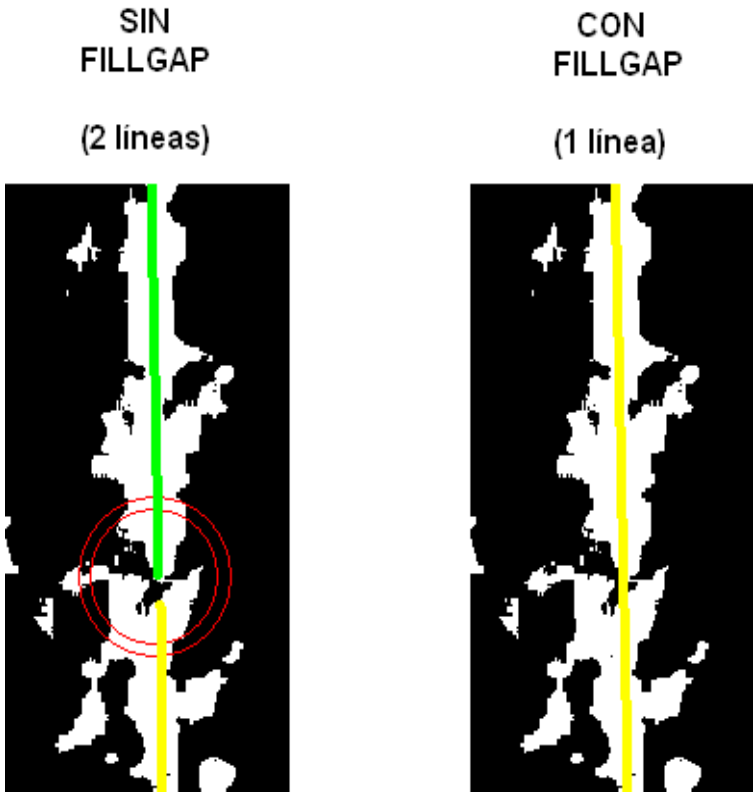

Figura 33: Resultado de usar el parámetro Fillgap

Un valor alto de FillGap (p.e. altura * 2/3 * 1/4) daría lugar a la detección de líneas no deseadas que tienen un ángulo correcto pero que se generan entre líneas de cultivo como se muestra en la imagen:

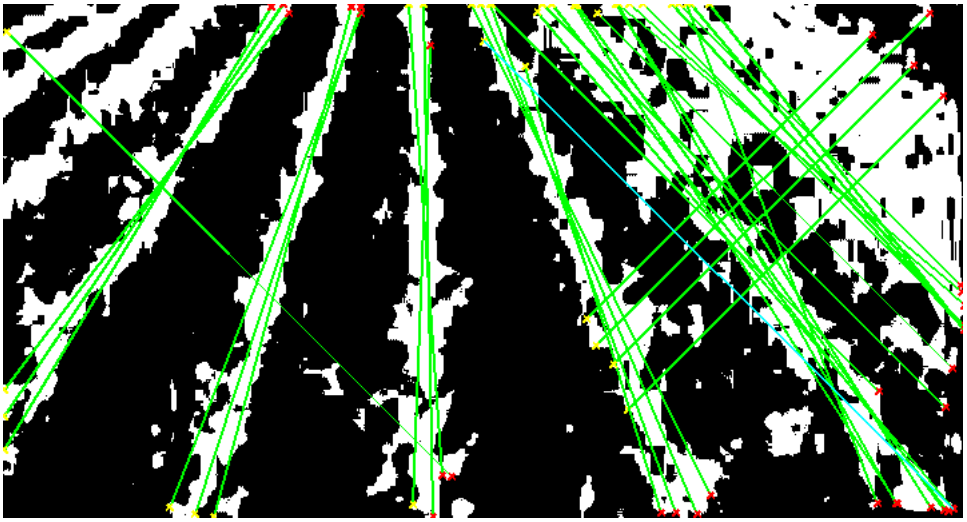

Figura 34: Líneas detectadas con un alto valor de FillGap.

Por lo tanto deberemos ajustar bien este parámetro para realizar una correcta detección.

- *'minLength'* y *valueMinLength*. El primero indica que el siguente parámetro a especificar será el valor de valueMinLength. Valor positivo que especifica si las lineas unidas deberían ser guardadas o descartadas. Las líneas menores que el valor especificado serán descartadas. En MATLAB se asocia a la función houghlines(...). Si una vez unidas las líneas en una sola ésta no tiene una longitud mínima entonces se descarta.

La buena elección de estos parámetros mejora la detección de las líneas que buscamos, y la combinación de ellos permite obtener unas importantes mejoras. Hay que tener en cuenta que existen parámetros que están relacionados como el valor de umbral y el FillGap ya que si se usa un valor de umbralizado bajo entonces es posible que tengamos que rellenar huecos mas grandes en las líneas de cultivo. Estos parámetros también deberían ajustarse a las condiciones sobre las que se extraen las imágenes(anchura de las lineas de cultivo, anchura entre líneas de cultivo, densidad de las plantas, iluminación...etc).

3.3. Selección de líneas óptimas usando el punto de fuga.

El uso de la transformada de Hough nos hace posible detectar gran cantidad de segmentos sobre la imagen que se aproximan bastante a las líneas de cosecha de la imagen real, pero de esta forma lo que hacemos es detectar uno o varios segmentos por cada línea de cosecha. Como cada uno de los segmentos detectados dentro de una misma línea posee distinta pendiente lo que pretendemos es reunir todos esos segmentos en uno sólo y de forma que se aproxime lo más posible a la línea de cosecha real de la imagen.

Este método se centra en el punto de fuga que forman líneas rectas paralelas cuando son vistas desde una proyección en perspectiva que en nuestro caso corresponde con las imágenes del campo de cosecha. Así, el primer objetivo es localizar las coordenadas del punto de fuga que forman las líneas de cosecha.

Para identificar el punto de fuga de las líneas nos encontramos con el siguiente problema: Tendremos muchos “puntos de fuga” ya que las líneas identificadas por Hough tienen distintas pendientes que hacen que cada par de líneas se corten en un punto distinto. De ese conjunto de puntos de corte hay que identificar cuáles son los más veraces, es decir, que se acercan más al punto de fuga real. Para determinar esa 'veracidad' asignaremos un peso a cada punto para luego poder calcular un único punto de fuga.

Calcular el conjunto de puntos de corte es muy sencillo teniendo un punto y la pendiente de cada recta gracias a la ecuación punto pendiente de la recta, $y - y1 = m\ (x-x1)$, ya que se crea un sistema de dos ecuaciones y dos incógnitas para dos rectas distintas donde las incógnitas son las coordenadas x e y donde éstas se cortan.

Pero para mejorar éste cálculo y hacerlo más veraz sólo tendremos en cuenta las líneas que pasen por el borde inferior de la imagen y por el borde superior de la imagen, de este modo

reducimos el número de líneas a tener en cuenta y además nos estamos quedando con las líneas que mejor han sido detectadas, que suelen ser las de la parte central de la imagen. Un ejemplo se ve en la siguiente imagen donde aparecen unas líneas moradas que marcan la última línea a cada lado que pasa por ambos bordes de la imagen:

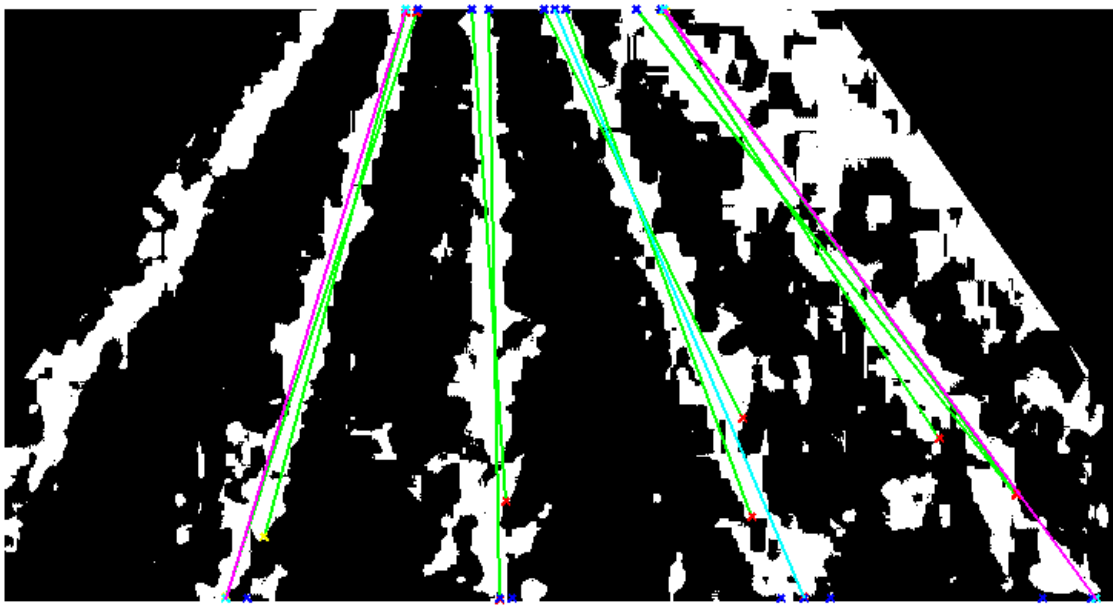

Figura 35: Líneas detectadas que pasan por los bordes superior e inferior.

Como se puede observar en la imagen anterior, hay líneas detectadas sobre la misma línea de cultivo que tienen su punto de corte con otra línea dentro de la misma imagen. Estos puntos de corte tampoco se tendrán en cuenta a la hora de calcular el punto de fuga de forma que sólo se computen aquellos puntos de corte cuyas coordenadas están por la parte superior de la imagen.

Una vez detectados los puntos de corte de las líneas de la imagen anterior podemos observarlos en un gráfico:

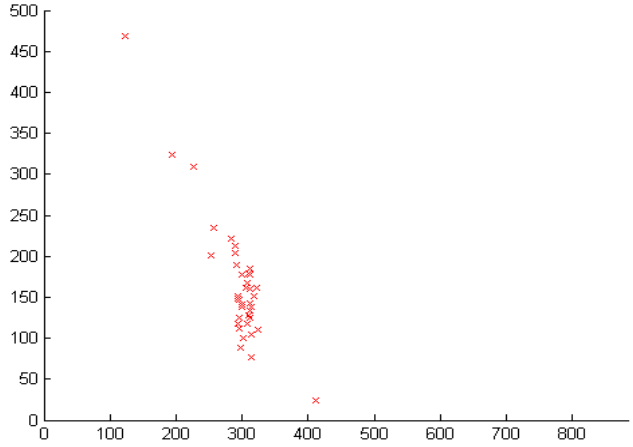

Figura 36: Mapa de puntos de corte de las líneas detectadas.

y como se puede apreciar, estos puntos tienden a situarse en una zona en concreto, que será la más

propensa a contener el punto de fuga aproximado. Una forma de comenzar a dar cierto nivel de veracidad a cada punto es asignar mayor peso si éste ha sido construido a partir de alguna recta con una pendiente muy vertical, ya que las líneas centrales serán más correctas porque es más fácil detectarlas en la imagen. En la siguiente imagen se aprecia en azul los puntos que se han generado cuando alguna de las rectas que se corta tiene un ángulo de 90º±5º:

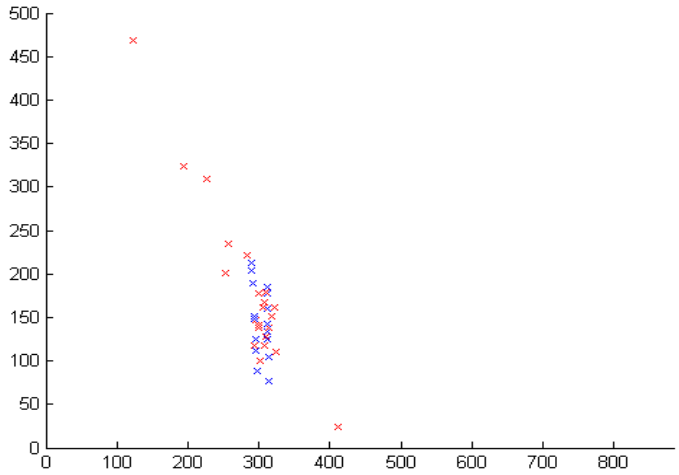

Figura 37: Puntos de corte de líneas con ángulo 90º aprox. (En azul).

A partir de esta imagen se puede ver que este conjunto de puntos azules está dentro de la zona donde la mayoría de puntos se acumulan. Es posible dentro de estos dos niveles de confianza meter otros niveles, que tengan en cuenta otros ángulos de las rectas.

Si analizamos más a fondo esta misma imagen de puntos teniendo en cuenta el ángulo Theta de las rectas que se cruzan podremos obtener más detalles para mejorar el reparto de pesos.
En el siguiente gráfico se muestra por color los siguientes puntos de corte:

Color	**Theta de las rectas que se cortan**	
Azul Marino	$\Theta \pm 5$	$\Theta \pm [5, 25]$
Amarillo	$\Theta \pm 5$	$\Theta \pm [25, \infty]$
Rojo	$\Theta \pm [5, 25]$	$\Theta \pm [5, 25]$
Cyan	$\Theta \pm [5, 25]$	$\Theta \pm [25, \infty]$
Magenta	$\Theta \pm [25, \infty]$	$\Theta \pm [25, \infty]$

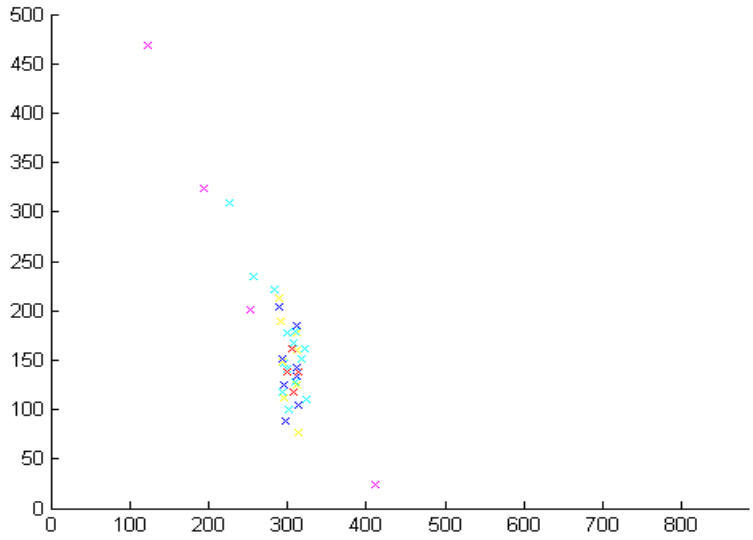

Figura 38: Puntos de corte según el ángulo de sus rectas.

En la figura anterior se aprecia que las intersecciones entre rectas cuyo Theta sea mayor de 25 representadas en color magenta están muy dispersas por lo que tendrán el menor peso. Después encontramos las de color cyan, (Theta de una recta es mayor de 25 y la otra está entre 5 y 25). Seguidamente están las de color azul marino y amarillo apreciándose las amarillas más dispersas. Y como más concentradas las rojas.

Hay que tener en cuenta que este resultado es específico para esta imagen, pero sí que se puede afirmar lo que anteriormente se citó: cuanto mayor es la pendiente de ambas rectas (cuando el Theta tiende más a cero) más se aproximan las intersecciones de las rectas al punto de fuga real. Esto se debe a que son líneas totalmente verticales y centradas que son las que pasan por el punto de fuga o quedan muy cerca.

A continuación veremos cómo averiguar las coordenadas del punto de fuga a partir de los puntos de corte detectados anteriormente. Lo primero que haremos será una asignación de pesos al punto de corte entre dos rectas dependiendo del parámetro Theta de la siguiente forma:

Color	**Theta de las rectas que se cortan**		**Peso**
Azul Marino	$\Theta \pm 5$	$\Theta \pm [5, 25]$	50
Amarillo	$\Theta \pm 5$	$\Theta \pm [25, \infty]$	30
Rojo	$\Theta \pm [5, 25]$	$\Theta \pm [5, 25]$	10
Cyan	$\Theta \pm [5, 25]$	$\Theta \pm [25, \infty]$	8
Magenta	$\Theta \pm [25, \infty]$	$\Theta \pm [25, \infty]$	2

Lo primero es calcular para puntos de un mismo tipo la media aritmética de sus coordenadas obteniendo un solo punto y una vez calculado el punto medio aritmético de cada tipo se halla el punto cuyas coordenadas sea la media ponderada de éstos.

En el siguiente gráfico se muestra indicado en círculos el punto medio aritmético de cada tipo (en el color correspondiente) y el circulo negro la media ponderada (Nota: La imagen ha sido

ampliada sobre la zona de más concentración de puntos para apreciar mejor los puntos de media aritmética y ponderada).

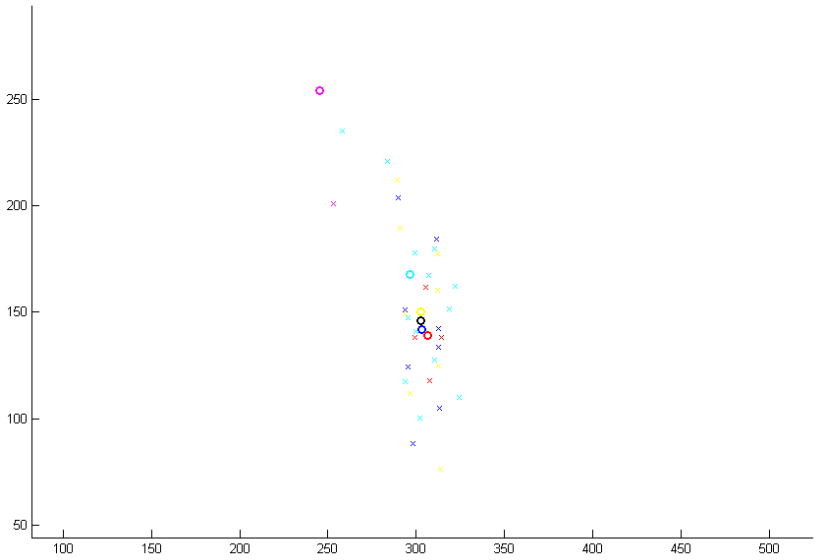

Figura 39: Puntos calculados mediante media aritmética.

Con las imágenes que estamos trabajando se ha usado la siguiente distribución de pesos que se ha seleccionado por método prueba y error hasta obtener una buena aproximación al punto de fuga real.

Color	**Theta de las rectas que se cortan**		**Peso**
Azul Marino	$\Theta \pm 5$	$\Theta \pm [5, 25]$	40
Amarillo	$\Theta \pm 5$	$\Theta \pm [25, \infty]$	60
Rojo	$\Theta \pm [5, 25]$	$\Theta \pm [5, 25]$	10
Cyan	$\Theta \pm [5, 25]$	$\Theta \pm [25, \infty]$	8
Magenta	$\Theta \pm [25, \infty]$	$\Theta \pm [25, \infty]$	2

Pero no todos los puntos de corte hallados nos servirán a la hora de calcular el punto de fuga aproximado y esto se debe a que la zona de puntos fiables siempre se encuentra en el mismo sitio. Es por ello que elegiremos los puntos de corte que pertenezcan a una zona en concreto del mapa de puntos.

Esta mejora la podemos ver en los siguientes gráficos. Este es el grafo de puntos de corte de las rectas para la imagen que de la figura 22:

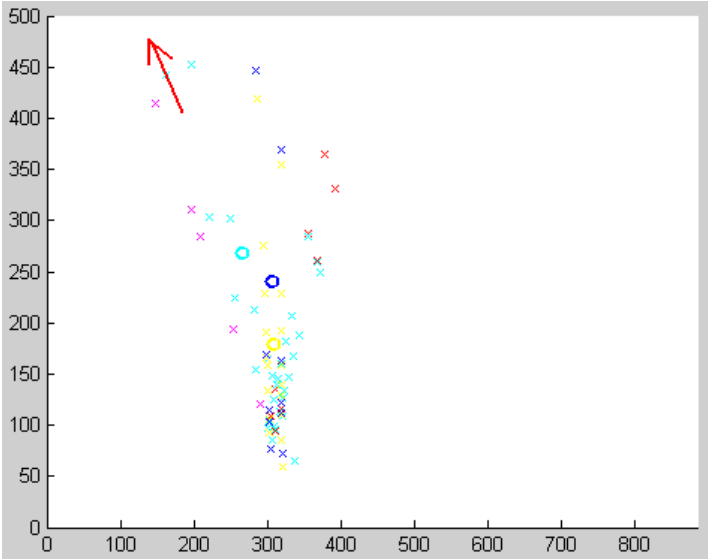

Figura 40: Mapa de puntos de corte para la figura 22.

Como se puede observar no aparecen los círculos rojo ni negro (correspondientes a la media aritmética de los puntos rojos y a la media ponderada que indica el punto de fuga aproximado) ya que éstos se encuentran en la dirección que indica la flecha roja debido a que existen otros puntos de corte rojos que se encuentran dispersos por encima del gráfico mostrado y que hacen que el cálculo del punto de fuga aproximado se altere siendo éste incorrecto. El efecto por tanto es que la línea central de cosecha no se detecta correctamente al estar desplazado el punto de fuga. En la siguiente imagen la línea roja vertical indica la posición del punto de fuga detectado:

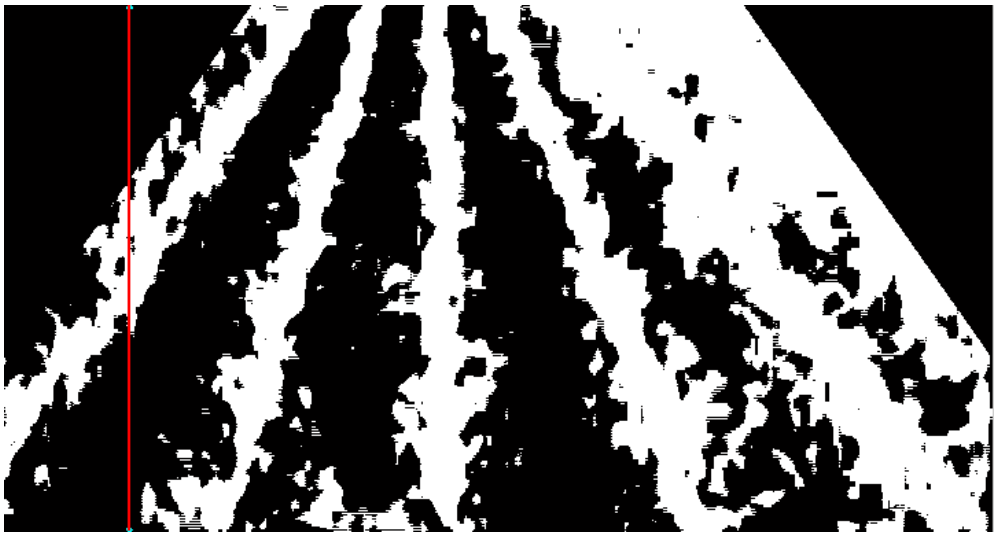
Figura 41: Línea que indica la posición del punto de fuga detectado.

Para solucionar este inconveniente hay que eliminar aquellos puntos de corte que de alguna manera no son correctos, ya que es posible que se detecten líneas que se corten en un punto cuya posición sea incorrecta. Estudiando los puntos de corte de las rectas detectadas, cabe señalar que la mayoría de éstos se acumulan en torno al centro de la imagen y en una altura por encima de la imagen igual a la altura total de la imagen. Esto se puede ver en el siguiente gráfico:

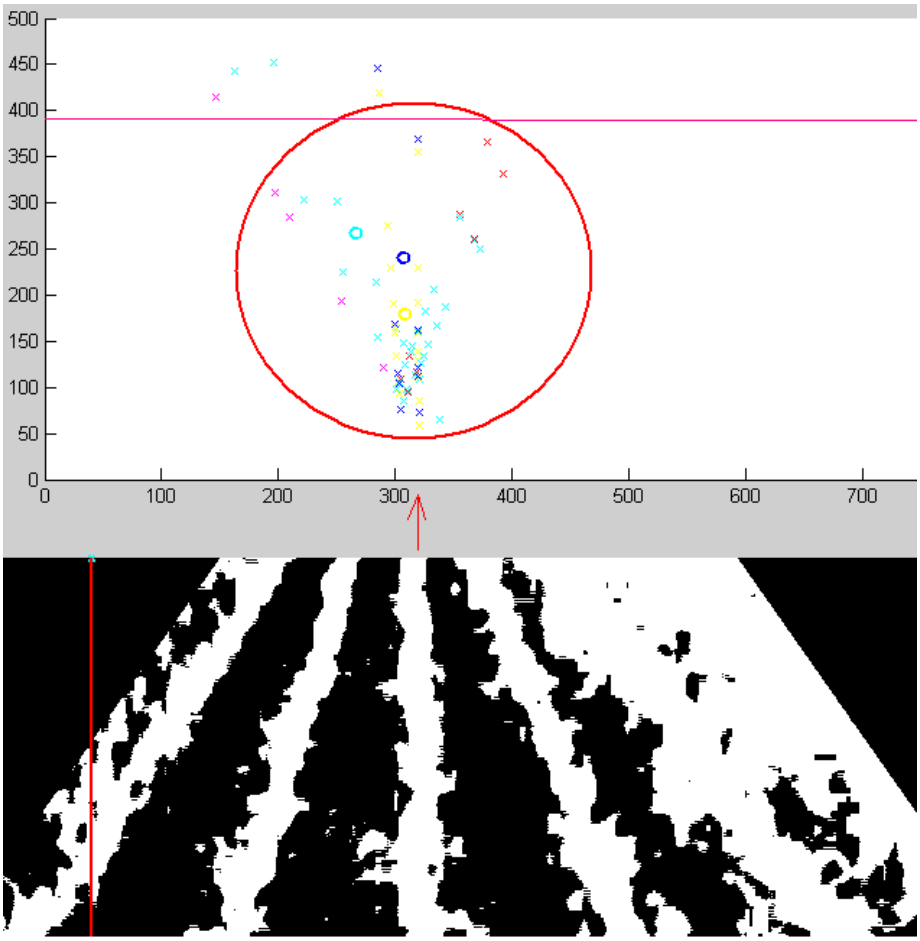

Figura 42: Zona de concentración de puntos de corte.

En la figura anterior se puede apreciar que la mayor densidad de puntos de corte se localizan en el círculo rojo, más o menos a la mitad del eje X (mitad de la imagen) y por debajo de la línea rosa equivalente a la altura correspondiente a la imagen (384 pixels). Éstos puntos son los que tienen una mayor fiabilidad de ser aproximaciones al punto de fuga de la imagen debido a que se encuentran en la zona donde se cortan realmente las líneas de cultivo que se aprecian en la imagen. Un gráfico que muestra esta coincidencia entre líneas de cultivo y puntos de corte es el siguiente:

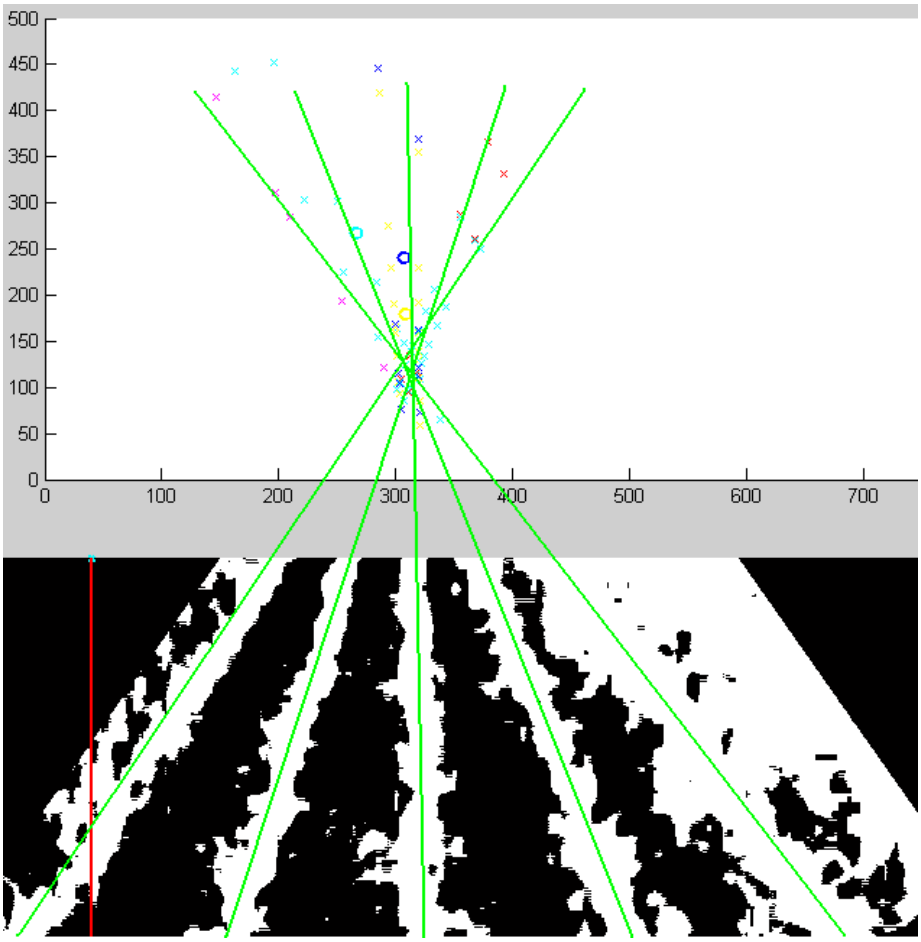

Figura 43: Mapa de puntos de corte y líneas de cosecha.

El resultado obtenido es mucho mejor y se puede ver en el video siguiente para cada frame la línea roja que indica la posición del punto de fuga: 'EstimacionPuntoFugaPesos40-60-10-8-2DistanciaImagen.avi'. También se puede observar en la siguiente figura (figura 43) la línea roja.

Para comparar resultados se ha realizado también la misma prueba pero teniendo en cuenta sólo la mitad de la altura de la imagen pero de esta forma se pierden puntos de corte que influyen de manera positiva a la hora de calcular el punto de fuga aproximado por lo que los resultados son aceptables pero menos precisos. Se puede apreciar en el video: 'EstimacionPuntoFugaPesos40-60-10-8-2DistanciaMitadImagen.avi'. (implementado en calculaPuntoFuga.m).

Aun con todo esto, es posible añadirle otra mejora para que se obtenga un punto de fuga aproximado muy cercano al original. Esto se consigue analizando las líneas verticales de pixels de la imagen próximas a la del punto de fuga, es decir, buscaremos en un intervalo de columnas cuyo centro es la columna de pixels donde está el punto de fuga. El objetivo es contabilizar el número de pixels de cada columna que están en blanco y quedarnos con la columna que mayor número posea. Primero hay que definir un porcentaje de columnas a ser exploradas (que será la longitud de cada mitad del intervalo), el cual está situado en el 2% de las columnas totales de la imagen y después se exploran las columnas para quedarnos con la de mayor numero de pixels en blanco. Esta operación de ajuste del punto de fuga consigue muy buenos resultados tal y como se puede ver en el video 'EstimacionPuntoFugaPesos40-60-10-8-2DistanciaImagenAjustado.avi'.

Podemos ver las operaciones anteriormente explicadas en la función que lleva a cabo esas operaciones, la función ajustaPuntogFuga.m:

```
function [fugaXajustado] = ajustaPuntoFuga(fugaX,imagen,ajusteLineas)


%HASTA AQUI SE OBTIENE UN PUNTO DE FUGA BASTANTE FIABLE PERO QUE NO
%ESTA AJUSTADO TENIENDO EN CUENTA LOS PIXELS DE LA IMAGEN.
%--AJUSTE DEL PUNTO DE FUGA EN RELACION A LOS PIXELS DE LA IMAGEN--
% Simplmente se miran los pixels de la imagen que están en la misma
% columna que el punto de fuga y se contabiliza el número de blancos.
% Esto se hace con varias columnas próximas a ambos lados del punto de
% fuga para ajustar posibles desviaciones.
if nargin < 3
    ajusteLineas = 2;
end

%Pasamos la imagen a binario (0's y 1's) es como umbralizarla.
I = im2bw(imagen);

%Calculo el numero de columnas a explorar a cada lado de la columna del
%punto de fuga.
[filas,cols] = size(I);
porcentaje = ajusteLineas;%Este valor esta puesto en %.P.e. si es un 2 pues se refiere a 2% del
total
intervalo = round(porcentaje * cols / 100);

%Cogemos el numero de pixels blancos de la columna del punto de fuga para
%poder comparar
columna = I(:,fugaX);
N = find(columna>0);
[totalFugaX,num]= size(N);
fugaXajustado = fugaX;

%Recorremos el intervalo de columnas proximas a la del punto de fuga.
for i = fugaX-intervalo:fugaX+intervalo
    %Cogemos la columna correspondiente
    columna = I(:,i);
    %Recogemos los pixels que sean blancos
    N = find(columna>0);
    %Vemos cuantos son
    [total,num]= size(N);
    %Si el numero de pixels blancos es mayor que el anterior almacenado se
    %registra como mejor columna de pixels.
    if total > totalFugaX
        fugaXajustado = i;
    end
end
```

Listado 5: Función de ajuste del punto de fuga ajustaPuntoFuga.

En la siguiente imagen se puede observar el resultado de este ajuste para un frame cualquiera del video anterior. Si nos fijamos la línea roja es la del punto de fuga aproximado calculado a partir de los puntos de corte de las líneas detectadas por Hough y la línea verde es la línea del punto de fuga obtenida después de la operación de ajuste.(Implementado en ajustaPuntoFuga.m).

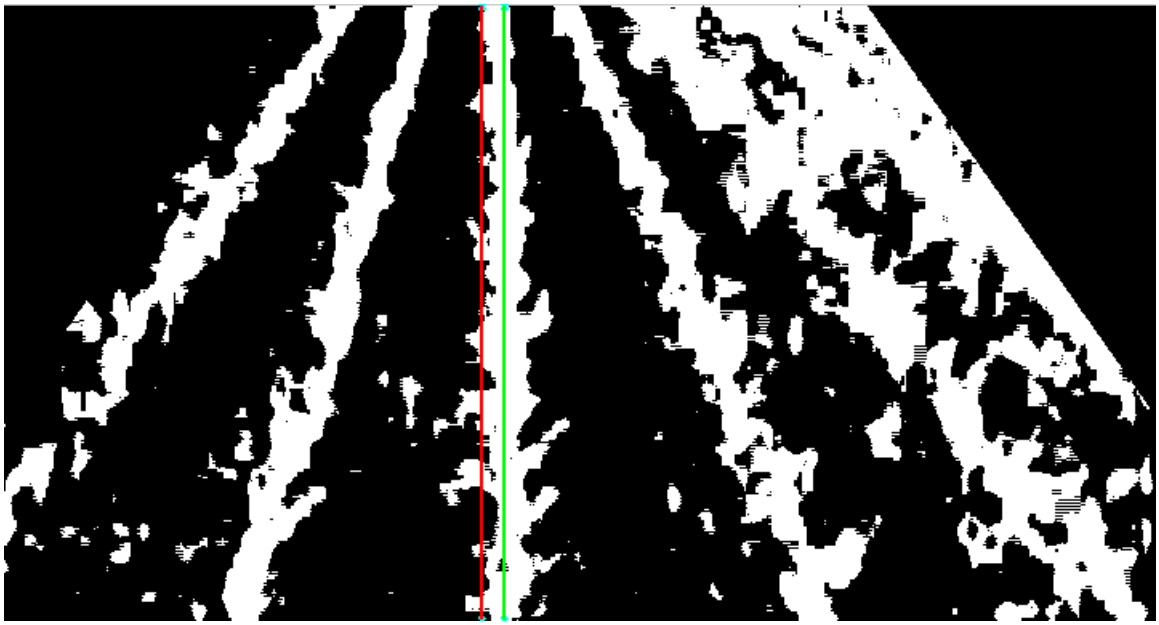

Figura 44: Líneas de punto de fuga ajustadas.

Una vez ajustada la coordenada 'X' del punto de fuga (representada como la línea verde en la imagen anterior) ya podemos generar las líneas de cosecha restantes. Éstas las generamos a partir de las coordenadas del punto de fuga y los puntos de corte de las líneas detectadas por Hough con el borde inferior de la imagen.

Usando los puntos de corte de las líneas detectadas por Hough con el borde inferior de la imagen(última fila de la imagen) se crean unos intervalos que corresponden a puntos de una misma línea de cosecha que están en el borde inferior, como vemos en la imagen en morado:

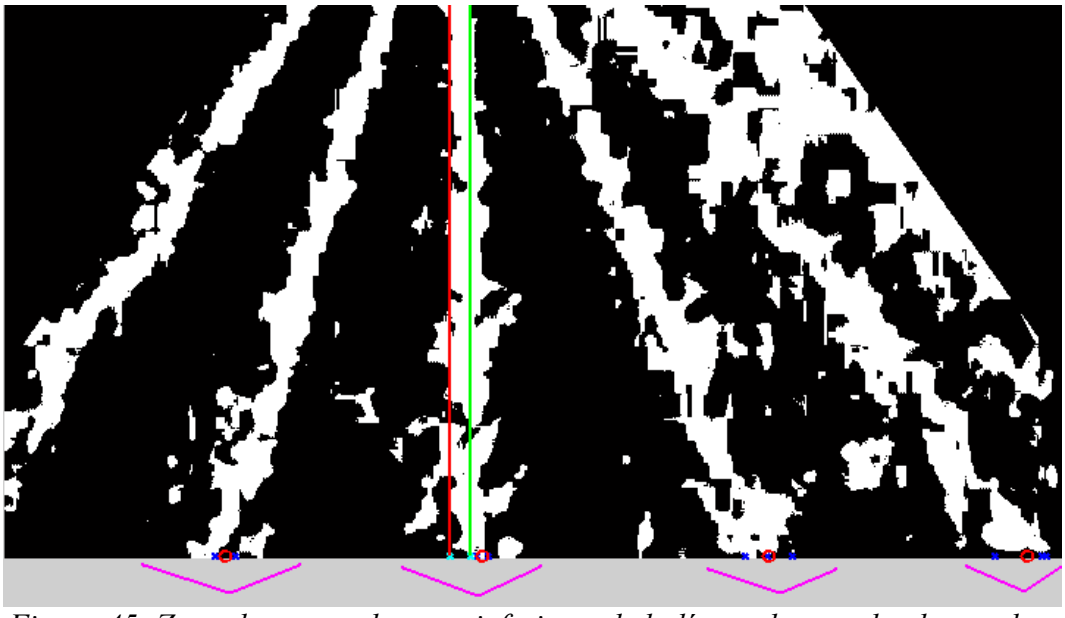

Figura 45: Zona de puntos de corte inferiores de la líneas de cosecha detectadas.

y a partir de esos puntos de corte marcamos para cada intervalo la posición media mediante un círculo rojo que indica el punto por donde pasará la línea de cosecha aproximada. Las operaciones realizadas para calcular las coordenadas de estos puntos se pueden ver en la función localizaLineas.m. Usando estos puntos inferiores de las líneas de cosecha aproximadas marcados en circulo rojo trazamos una línea a partir de éstos y que también pase por las coordenadas del punto de fuga calculado y con ello obtenemos otro punto que es el que corta con la primera fila de la imagen (borde superior) que podemos ver en la imagen siguiente:

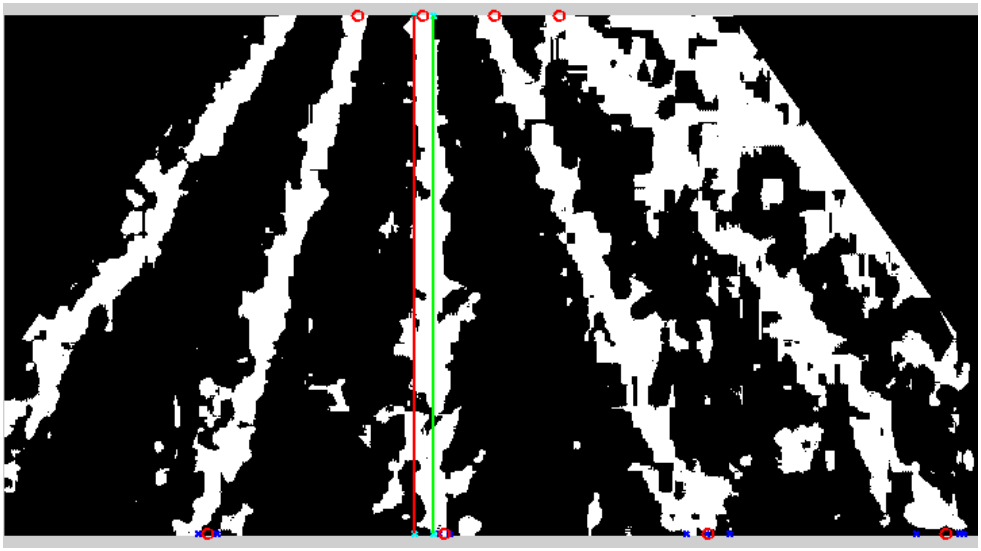

Figura 46: Puntos de corte superiores e inferiores de las líneas.

y con cada par de puntos (superior e inferior) ya tendremos las coordenadas de las líneas de cosecha que hemos detectado. Se pueden ver en azul sobre la siguiente figura.

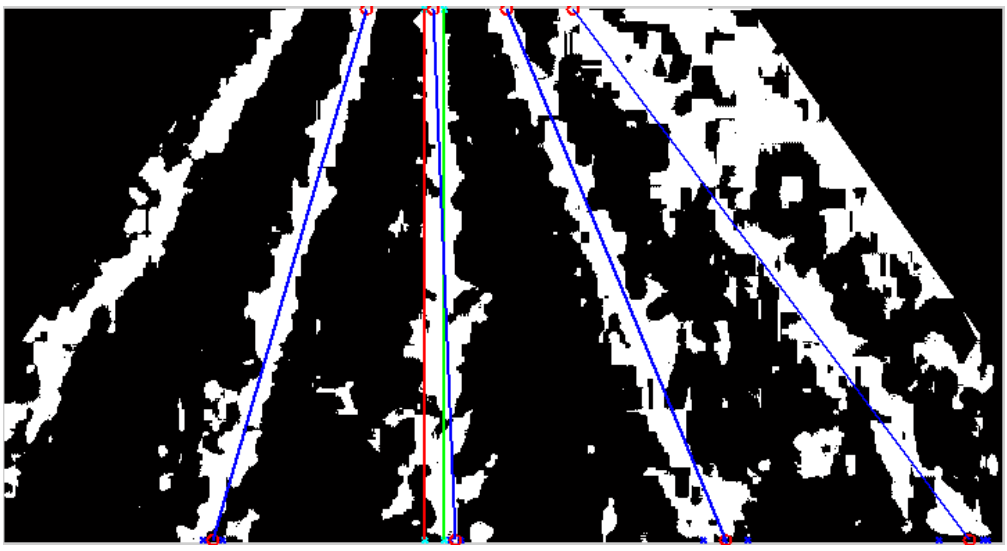

Figura 47: Líneas de cosecha detectadas (en azul).

Se puede observar que las líneas generadas coinciden con las líneas de cosecha de la imagen con gran precisión. Se pueden ver también los resultados obtenidos en el video generado en el archivo 'LineasEstimadasConPtoFugaSinAjustar.avi' y 'LineasEstimadasConPtoFugaAjustado.avi'.

Esta implementación que ejecuta Hough en su inicio y luego intenta ajustar las líneas mediante el punto de fuga y la imagen actual produce el siguiente resultado visto sobre la imagen original.

Figura 48: Resultado final de detección de líneas.

Este resultado que se puede apreciar sobre un frame del vídeo sobre el que se esta trabajando muestra un resultado bastante bueno, pero no siempre será así debido a los distintos factores que influyen la correcta detección de las líneas (iluminación, vibraciones,...etc).

El resultado obtenido para cada frame es bastante aceptable en lo que refiere a detección de líneas, pero en el momento en el que miramos el coste computacional para cada uno de los frames obtenemos una complejidad demasiada alta al estar trabajando sobre vídeo a 25fps.

Esto implica un pre-procesamiento de la imagen y el posterior cálculo de la transformada de Hough 25 veces cada segundo. Anotando el tiempo de procesamiento de estas operaciones durante 25 frames en un segundo obtuvimos un tiempo medio de procesamiento de 1.33532917 segundos para cada frame cuando para ser un tiempo aceptable debería ser del orden de 1/25 segundos (0.04s) ya que este tiempo es el que tiene cada frame de la imagen a 25 frames por segundo.

Debido a este inconveniente se puede pensar en aplicar otras técnicas para la detección de líneas que tengan un menor tiempo de ejecución. En el siguiente apartado nombraremos algunas de esas técnicas alternativas.

3.4. Otros resultados experimentales

Como en cualquier proyecto es importante también hacer notar las dificultades que presenta en su aplicación y como muestra de ello analizaremos un frame en donde no se llega a localizar de forma correcta las líneas de cosecha debido a ciertos factores que expondremos a continuación.

En la siguiente imagen se pueden apreciar las líneas que se han detectado superpuestas sobre la imagen original.

Figura 49: Líneas detectadas incorrectamente.

Como se puede apreciar algunas de las líneas presentan una inclinación errónea por lo que analizaremos el porqué de ese error.

Para empezar a comprender estos errores que no sólo se dan en esta imagen sino en varios frames del vídeo comenzaremos analizando la imagen binarizada para apreciar su composición.

Figura 50: Imagen binarizada.

Si nos fijamos bien en la figura 50 de la imagen binarizada se aprecia una gran zona en la parte superior derecha que está en blanco. Todos esos pixels en blanco se consideran zona vegetal y en realidad sí que es zona vegetal como se aprecia en la figura 49. Esto es debido al avanzado estado de crecimiento de la cosecha y de las propias características de ésta las que hacen que al segmentar la imagen se obtenga como resultado la figura 50. Pero este resultado sería posible arreglarlo si antes de recortar las esquinas de la imagen se hiciera un pequeño análisis de la cantidad de pixels en blanco de cierta zona para ajustar el recorte de la esquina de forma óptima, trabajo que se tiene en cuenta como trabajos futuros de este proyecto.

Al analizar la imagen con tanta información sobre densidad vegetal que dificulta la detección de las verdaderas líneas, el resultado de aplicar Hough es el siguiente:

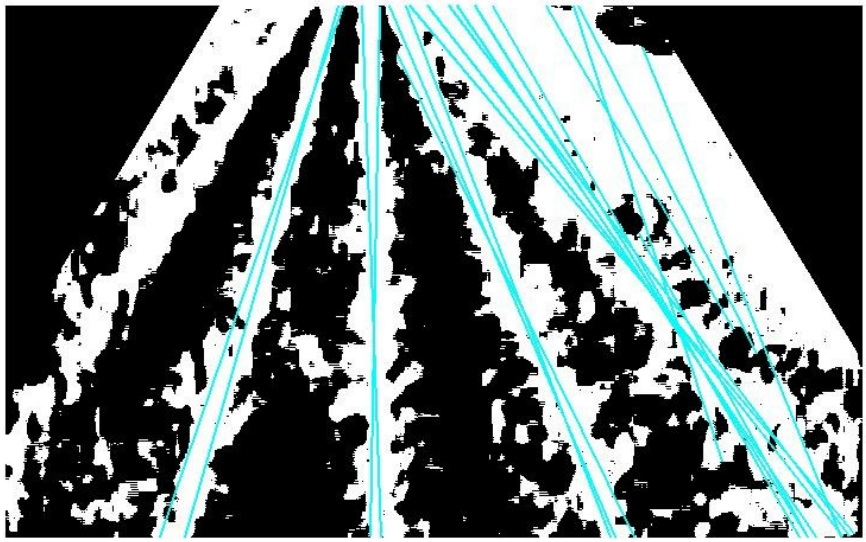

Figura 51: Líneas detectadas por Hough.

Como resultado se obtienen 4 líneas (líneas situadas más a la derecha) que son detectadas pero que realmente no existen así que deberían de eliminarse pero al tener una inclinación correcta se siguen teniendo en cuenta. Tal vez se puede pensar en cambiar los parámetros de Hough de forma que se eliminen estas líneas pero esto influiría sobre el resto de líneas que sí han sido detectadas de forma correcta y que son mayoría. Ni con los parámetros MinLength ni Fillgap es posible obtener resultados iguales y que además eliminen esas líneas erróneas.

Para trabajar con las imágenes binarizadas que no han podido eliminar estas zonas que interfieren en la correcta detección de las líneas es posible intentar modificar la parte de cálculo del punto de fuga, pero habría que hacer un estudio más profundo de los pesos asignados a cada tipo de recta y su inclinación o como propuesta alternativa intentar clasificar todas y cada una de las rectas obtenidas no sólo por su inclinación como se hace en este proyecto sino también por su situación en la imagen en un intento por diferenciar líneas erróneas de las correctas.

La detección de estas líneas erróneas al aplicar Hough son el verdadero problema a intentar resolver aunque también hay que indicar que las imágenes de entrada también tienen mucho que decir ya que en algunos casos no se tendrá información suficiente para detectar ninguna línea y no devolver por tanto ninguna línea detectada, sobre todo en campos totalmente cubiertos por zona vegetal.

4. Alternativas de diseño

Como se ha visto a lo largo de este proyecto se ha considerado únicamente la transformada de Hough a la hora de hacer la detección de segmentos. El principal problema de la transformada de Hough es su elevado coste computacional, sin embargo han aparecido multitud de variantes que pretenden solucionar esto.

Una alternativa es considerar sólo un subconjunto de los datos de entrada seleccionados de forma aleatoria. Las variantes que adoptan esta solución son conocidas como transformadas de Hough Probabilísticas (PHT) y ha sido demostrado que pueden reducir considerablemente el coste temporal de ejecución obteniendo resultados muy próximos a los de la transformada de Hough estándar.

El coste computacional es un aspecto crítico para elegir la implementación a utilizar ya que en este proyecto se pretende aplicar la transformada de Hough para realizar un control en tiempo real de una plataforma móvil. Se puede encontrar en OpenCV (una biblioteca libre de visión artificial originalmente desarrollada por Intel) tres versiones diferentes de la transformada Hough, donde está la implementación de la Transformada de Hough Progresiva Probabilística (PPHT) propuesta por Matas, Galambos y Kittler[13].
La idea de este algoritmo es considerar píxeles aleatorios uno a uno. En cada ocasión el acumulador es actualizado y se comprueba si el pico más alto sobrepasa un umbral. En caso de que esto ocurra, los puntos que pertenecen al pasillo especificado por el pico son eliminados. Si el número de puntos excede un valor predefinido de longitud mínimo de línea, entonces es considerado una línea, en otro caso en considerado ruido. Luego se repite el proceso desde el principio hasta que no queden más píxeles en la imagen. El algoritmo mejora el resultado en cada paso, por lo que puede ser interrumpido en cualquier momento.

Este método posee una desventaja ya que, al contrario que la transformada de Hough estándar, no es capaz de procesar correctamente algunas características de la imagen, como por ejemplo líneas cruzadas. Sin embargo esto no supone un problema para la aplicación concreta en este caso ya las líneas a detectar en la imagen no se cruzan al ser líneas de cosecha que siempre son paralelas.

5. Conclusiones y vías futuras

En este proyecto se han conseguido los objetivos propuestos en un principio, de aplicar una técnica conocida sobre un caso real y solucionar los inconvenientes que surgen de este caso en concreto, obteniendo una herramienta que funciona perfectamente para los requisitos de este proyecto, sin embargo conviene realizar una implementación más eficiente en C si se quiere usar bajo Hardware de bajo coste.

En el trabajo con imágenes y su procesamiento en tiempo real es de gran importancia usar métodos que permitan obtener una respuesta rápida para lograr el objetivo propuesto, es decir, poder analizar las imágenes de forma que obtengamos la información necesaria para tomar una decisión en un tiempo asequible. Hay veces que este tiempo es crítico y otras no, y en nuestro caso el tiempo sí es crítico ya que el objetivo es poder analizar la imagen y extraer las líneas de cosecha antes de que el vehículo tractor pase por encima del lugar que se está analizando.

Debido a la importancia del tiempo de respuesta se piensa siempre en aligerar los cálculos que se han usado. La transformada de Hough es un cálculo bastante pesado y que requiere multitud de operaciones por lo que a priori no parece una opción adecuada a manejar en cada uno de los frames del vídeo. Teniendo en cuenta que nos llegan imágenes a 25 frames por segundo y que se debe aplicar Hough a cada frame del vídeo, estamos ante una cantidad de cálculo que no es conveniente para nuestros objetivos.
En la idea de aligerar este cálculo y como trabajo futuro se piensa en no aplicar Hough en todos y cada uno de los frames del vídeo de entrada sino que se podría hacer uso de un historial de las líneas que han sido anteriormente detectadas para tratar de intuir cómo serán las siguientes líneas de cosecha en el siguiente frame, ya que por la naturaleza de éstas líneas tienden a ser casi iguales que el frame anterior. Para aprovecharnos de esta 'monotonía' en los campos de cosecha se piensa en hacer uso de ese historial de líneas intercalado con la aplicación de la transformada de Hough en el sentido de que cada cierto tiempo o número de frames se puede aplicar la transformada de Hough para intentar corregir desviaciones que se hayan producido por el uso de historial de líneas.

A lo largo de este documento se han ido aplicando distintas operaciones que se han realizado de forma específica para el vídeo sobre el que se ha trabajado en este proyecto. Es por ello que se han usado en varias funciones parámetros específicos que no siempre serán válidos cuando se utilice otro vídeo como input. Teniendo esto en mente a continuación se indican una serie de trabajos futuros que pueden hacer este proyecto aplicable a más casos de campos de cosecha que no sea el aquí analizado. Las operaciones que automatizarían la selección de éstos parámetros podrían ser las siguientes:

- Selección automática del nivel de umbral. Esta parte se lleva a cabo en el segmentado de la imagen cuando se binariza la imagen (ver figuras 18, 19 y 20). El valor del nivel de umbral a aplicar deberá depender exclusivamente del estado de la plantación donde se analizará la densidad vegetal de cada línea de cosecha. Este nivel de umbral deberá tenerse en cuenta también a la hora de aplicar el parámetro de FillGap (ver sección 3.2.1) para poder rellenar líneas en caso de tener poca densidad vegetal que presentaría bastantes huecos (ver figura 18).
- Ajuste automático de los puntos extremos de la línea de Bresenham óptimos. Usado especialmente para recortar las esquinas y eliminar la información no válida que hay en éstas. Se puede pensar en algún método para dar con las coordenadas que formarían la línea

que consigue eliminar de forma óptima las esquinas sin alterar las líneas de cosecha bien detectadas.

- Estudio de la distribución de pesos para puntos de corte. Usados en la aproximación del punto de fuga, ya que la distribución usada en este proyecto (40,60,10,8,2) ha sido conseguida mediante el método de prueba y error y para conseguir una distribución óptima se podría hacer un estudio que intentara averiguarlos.

6. Bibliografía

[1] R.C. Gonzalez, R.E. Woods, S.L. Eddins, Digital Image Processing using Matlab, Prentice Hall, New York, 2004.

[2] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Addison-Wesley, New York, 1993.

[3] Alberto Ruiz, Apuntes de Sistemas de Percepción y Visión por Computador, Departamento de Informática y Sistemas, Facultad de Informática, Universidad de Murcia, 2009.

[4] Alberto Tellaeche, Xavier P. Burgos-Artizzu, Gonzalo Pajares, Angela Ribeiro, A vision-based method for weeds identification trough the Bayesian decision theory, Pattern Recognition 41 (2008) 521-530.

[5] Xavier P. Burgos-Artizzu, Angela Ribeiro, Alberto Tellaeche, Gonzalo Pajares, Cesar Fernández-Quintanilla, Improving weed pressureassessment using digital images from an experience-based reasoning aproach, Computers and Electronics in Agriculture 65 (2009) 176-185.

[6] Alberto Tellaeche, Xavier P. Burgos-Artizzu, Gonzalo Pajares, Angela Ribeiro, Cesar Fernández-Quintanilla, A new vision-based approach to differential spraying in precision agriculture, Computers and Electronics in Agriculture 60 (2008) 144-155.

[7] A. Ribeiro, C. Fernandez-Quintanilla, J.Barroso, M.C. García-Alegre, Development of an image analysis system for estimation of weed, in: Proceedings of the 5th European Conference on Precision Agriculture (5ECPA), 2005, pp. 169-174.

[8] Algoritmo de Bresenham, http://en.wikipedia.org/wiki/Bresenham's_line_algorithm

[9] Documentación Matlab Online. Función Hough. Agosto 2009. http://www.mathworks.com/access/helpdesk/help/toolbox/images/hough.html

[10] Documentación Matlab Online. Función HoughLines. Agosto 2009. http://www.mathworks.com/access/helpdesk/help/toolbox/images/houghlines.html

[11] Documentación Matlab Online. Función HoughPeaks. Agosto 2009. http://www.mathworks.com/access/helpdesk/help/toolbox/images/houghpeaks.html

[12] G. Pajares, J.M. De la Cruz, Visión por computador: Imágenes digitales y aplicaciones, RA-MA, 2001, Madrid.

[13] J. Matas, C. Galambos and J. Kittler, Progressive Probabilistic Hough Transform, University of Surrey, United Kingdom.