UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de un sistema de reconocimiento y conteo del parásito *Varroa destructor* en colmenas para probar efectividad de tratamientos

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Juan Luis Hernández García

Tutor: Juan Miguel García Gómez

Curso 2019-2020

Resum

Este projecte va ser proposat per un apicultor que coneix esta malaltia de primera mà. Es pretén com a principal objectiu desenrotllar un sistema que permeta realitzar un conteo automatitzat del paràsit. Actualment açò es realitza de forma manual, per la qual cosa amb este projecte es vol aconseguir un sistema que principalment accelere este procés.

Per a complir este objectiu s'ha utilitzat OpenCV, una llibreria de processament d'imatges de codi obert i molt completa, que permet el desenrotllament de sistemes de reconeixement d'objectes i formes en imatges, amb la qual s'ha implementat gran part d'este projecte. A més s'ha desenrotllat una pàgina web, per a així fer més senzill i accessible als usuaris l'ús d'esta ferramenta.

El mètode utilitzat per al desenrotllament d'este projecte, comença per un processat de la imatge i posterior extracció de característiques d'esta, seguit d'una anàlisi de les dades extrets, per a aconseguir aïllar les característiques que definixen a una varroa en una imatge i finalment l'aplicació d'estes per a la visualització dels resultats.

Com a primera aproximació, este projecte s'acosta prou als objectius que es plantegen, però no s'ha aconseguit obtindre uns resultats òptims que poguera aplicar-se a entorns reals.

Paraules clau: Apicultura, Varroa, Visió per Computador, Servici Biotecnològic

Resumen

Este proyecto fue propuesto por un apicultor que conoce esta enfermedad de primera mano. Se pretende como principal objetivo desarrollar un sistema que permita realizar un conteo automatizado del parásito. Actualmente esto se realiza de forma manual, por lo que con este proyecto se quiere conseguir un sistema que principalmente acelere este proceso.

Para cumplir este objetivo se ha utilizado OpenCV, una librería de procesamiento de imágenes de código abierto y muy completa, que permite el desarrollo de sistemas de reconocimiento de objetos y formas en imágenes, con la cual se ha implementado gran parte de este proyecto. Además se ha desarrollado una página web, para así hacer más sencillo y accesible a los usuarios el uso de ésta herramienta.

El método utilizado para el desarrollo de este proyecto, empieza por un procesado de la imagen y posterior extracción de características de esta, seguido de un análisis de los datos extraídos, para conseguir aislar las características que definen a una varroa en una imagen y por último la aplicación de éstas para la visualización de los resultados.

Como primera aproximación, este proyecto se acerca bastante a los objetivos que se plantean, pero no se ha conseguido obtener unos resultados óptimos que pudiera aplicarse a entornos reales.

Palabras clave: Apicultura, Varroa, Visión por Computador, Servicio Biotecnológico

Abstract

This project was suggested by a beekeeper that knows those illness at first-hand. The main goal develop a system that is able to carry out an automatic count of the total amount of parasitic in the area. Nowadays, it is makes manually, so this project will create a faster system, reducing the timing process.

To achieve this goal we have used OpenCv, a complete open-source library of image processing. That's allows to develop a system recognition of shape and object. Next to it, we have create a website with friendly interface making easier in order to facilitate its use.

The method begins with an image processing and later extract features from this image. After that, we analice this extracted data, to achieve the isolation of the features that defines a varroa in a image and finally, we apply this characteristics to display the results.

This project is very close to the set goals almost obtaining the optimum results that allows to apply to real environments.

Key words: Apiculture, Varroa, Computer Vision, Biotechnological Service

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

El ácaro *Varroa destructor* afecta de forma generalizada a todas las colmenas del mundo, y su enfermedad, la varroosis, es la que más daños causa en la apicultura. Esta enfermedad afecta tanto a las crías de las abejas como a las adultas. En España es la única enfermedad apícola que necesita de manera obligada un tratamiento para mantener las tasas de parásitos por debajo de un umbral [1].

Simplificando, para el control de este parásito se aplica un determinado tratamiento que despega a las varroas del cuerpo de la abeja, cayendo éstas al fondo de la colmena, donde previamente se ha colocado una hoja cuadriculada. Posteriormente, cuando el tratamiento ha hecho efecto, se saca esta hoja y se contabilizan el número de varroas caídas, para así conocer el estado de salud de esa colmena. Este proceso de contabilización actualmente lo realiza el apicultor de manera manual, y debido al extenso tiempo que emplea se pensó en crear una solución tecnológica, un sistema de reconocimiento de imágenes que sea capaz de identificar cúal es una varroa y cuantas hay.

Los sistemas de reconocimiento en imagen están en continuo avance, debido al desarrollo de nuevas técnicas de *Deep Learning* o aprendizaje profundo, siendo éste de las áreas de la inteligencia artificial donde más invierten las empresas de Estados Unidos, según un estudio de O'Really [2].

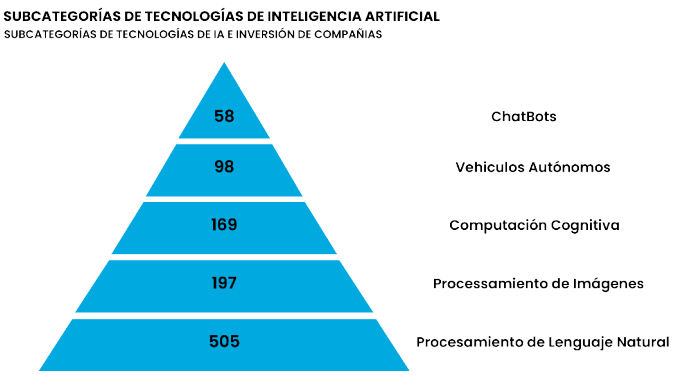

Figura 1.1: The New Artificial Intelligence Market. O'Reilly. Aman Naimat

1.1 Motivación

Debido a que la varroosis es una enfermedad que afecta a gran parte de las colmenas del mundo, y sus consecuencias pueden ser muy graves para la población de abejas, así como para las propios apicultores, profesionales y aficionados, como a niveles económicos y del ecosistema. Puesto que el sector de la apicultura genera unos 62 millones de euros al año en España, ademas de ser la abeja una de las principales polinizadoras de los cultivos. También es importante destacar que es un sector en crecimiento constante, creciendo en España un 20,87 % y en Europa un 7,63 % el número de apicultores censados en el periodo de 2017-2020 [3].

Por lo que puesto en contexto todo esto lo que me llevó a la realización de dicho proyecto es poder contribuir con una herramienta de ayuda a este sector que permitiese frenar éste problema y además contribuir con la preservación de las abejas. Además, gracias a la asignatura de Bioinformática que me ayudó a entender como aplicar la informática a entornos de biología, y los grandes avances que se pueden conseguir con ello. Otro motivo fué también el desarrollar un sistema de reconocimiento de formas y procesamiento de datos que me ha ayudado a adentrarme al mundo de la inteligencia artificial.

1.2 Objetivos

En este apartado se van a detallar los objetivos que se van a desarrollar en este TFG. Éste tiene como principal objetivo desarrollar un sistema de conteo de varroas mediante imágenes que pueda ser accedido desde cualquier sitio y dispositivo.

Para desarrollar esto, se ha dividido el objetivo principal en los siguientes objetivos:

- Desarrollo de un sistema de reconocimiento de objetos a través de imágenes, mediante el cual se clasifica los objetos reconocidos en dos clases, varroas y puntos irrelevantes
- Integración del sistema con un servidor web para poder examinar imágenes de forma sencilla.
- Desarrollo de la web en la que subir las imágenes.
- Pruebas y evaluación con diversos algoritmos e imágenes para verificar la precisión del sistema.

1.3 Estructura de la memoria

La memoria de este trabajo está dividida en ocho capítulos. El **primer capítulo**, el actual, expone la motivación y los objetivos de este proyecto.

En el **segundo capítulo** se realizará un análisis de los sistemas que realizan funciones similares, y cumplen parte de los objetivos de éste proyecto.

El **tercer capítulo** se centra en analizar el problema, para así proceder a describir el diseño de la solución.

Como se ha comentado, el **cuarto capítulo** establece los patrones de diseño de la solución al problema propuesto por este proyecto.

Posteriormente, en el **quinto capítulo** se desarrolla la solución que se ha diseñado en el capítulo anterior.

El **sexto capítulo** explica las pruebas realizadas y los resultados que se han obtenido con distintas configuraciones del sistema.

Después, en el **séptimo capítulo** se exponen las conclusiones a las que se ha llegado después de la realización del presente trabajo.

Por último, en el **octavo capítulo** se comentan las siguientes evoluciones que se plantean tras haber realizado este trabajo.

CAPÍTULO 2
Estado del arte

En este capítulo se realizará un estudio sobre los sistemas existente para la identificación de objetos en el ámbito de la biología. Además se estudiará una propuesta para el problema a resolver.

2.1 Aplicaciones de los sistemas de reconocimiento de objetos

El mercado de los sistemas de reconocimiento de objetos está actualmente en auge, se está aplicando a numerosos ámbitos muy dispares. Una de las razones de este aumento es debido al uso de ellos en la conducción autónoma o el reconocimiento facial entre otros.

En el campo de la agricultura, nos encontramos sobre todo sistemas de identificación de árboles y plantas, aunque también se han encontrado algunas para identificar enfermedades que puedan tener estos. Concretamente en el campo de la identificación de la "Varroa destructor" no se ha encontrado ningún sistema que lo haga de forma automática.

2.1.1. Plantix

Plantix[1] es una aplicación móvil para el diagnostico de enfermedades, plagas o deficiencias en cultivos. Se definen como "El doctor de sus cultivos". Este proyecto fue creado con el objetivo de aumentar el rendimiento de las campos, ya que las enfermedades de los cultivos causan perdidas de alrededor del 42 % del rendimiento, llegando en algunos casos a destruir el cultivo al completo.[4]

Permite enviar fotos a través de la aplicación y obtener la causa de la enfermedad, utilizando un simple smartphone. Plantix es capaz de detectar mas de 240 afecciones de las plantas plantas gracias a la gran cantidad de fotos subidas, proporcionando así, cada vez mejores resultados.

[1] https://plantix.net/es/

Figura 2.1: Pantalla inicio de Plantix

Figura 2.2: Resultado de analizar imagen

Además proporciona una información detallada de la enfermedad detectada, proporcionando detalles de métodos de control biológicos y químicos para frenarla.

2.1.2. PictureThis

PictureThis[2] es una aplicación móvil de identificación de plantas a través de imágenes.

Permite realizar una fotografía con el smartphone y subirla, entonces esta aplicación obtiene instantáneamente y con precisión la planta de la que se trata. Utilizando un sistema inteligencia artificial. Por lo que consigue una precisión de un 95 %, pudiendo además identificar el 99 % de las especies mas comunes, puesto que tiene una base de datos con mas de 10.000 especies.

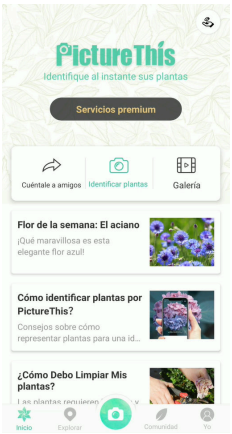

Figura 2.3: Pantalla inicio de PictureThis

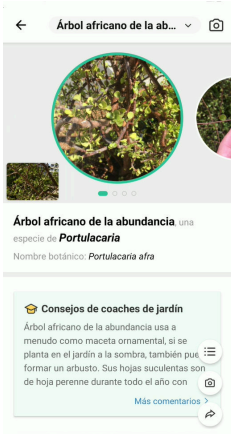

Figura 2.4: Resultado de analizar imagen

[2] https://www.picturethisai.com/

Una vez analizada la fotografía muestra varias posibilidades ordenadas por mayor semejanza, además expone un análisis de la planta en el que indica una pequeña descripción, datos de interés, características y varias fotografías de esa planta.

2.1.3. Picture Insect

Picture Insect es una aplicación móvil para identificar insectos a través de imágenes.

Con cualquier smartphone se puede realizar una fotografía a un insecto y mediante esta aplicación saber de qué especie se trata de forma rápida y sencilla. Puede conseguir hasta un 95 % de precisión y permite identificar mas de 1000 especies de insectos.

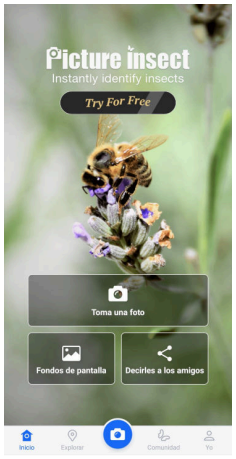

Figura 2.5: Pantalla inicio de Picture Insect

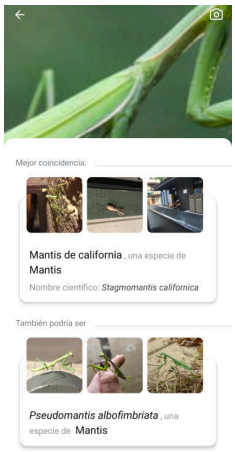

Figura 2.6: Resultado de analizar imagen

Después de analizar la imagen, muestras las posibilidades que encuentra, así como una descripción de la especie.

2.2 Propuesta

El propósito de este proyecto es proporcionar a los apicultores una herramienta que les permita conocer la salud de sus colmenas mediante el conteo de los parásitos caídos de la colmena, utilizando imágenes, de una forma rápida y sencilla. Como se ha observado en el punto 2.1 lo que se ha encontrado son aplicaciones para conocer enfermedades que puedan tener plantas y arboles, a través de imágenes, así como identificar de que especie se tratan, pero no se ha encontrado nada en el mercado que realice específicamente esta tarea. Por ello éste trabajo parte de cero, ya que actualmente esto se realiza de forma manual, es decir una persona cuenta uno a uno cada uno de los parásitos.

CAPÍTULO 3
Análisis del problema

En este capítulo se va a analizar el problema a resolver y qué solución es la que más se adecua a ello. El sistema de detección en el que se va a trabajar quiere resolver el problema actual que supone que el conteo se haga de forma manual, sobre todo enfocado a reducir el tiempo de éste, además de proponer una solución que sea sencilla de utilizar. Por ello, se han establecido una serie de requisitos que se deben cumplir:

- **Rapidez:** el tiempo que tarde el usuario en realizar la fotografía, subirla a la web y obtener los resultados debe ser reducido.
- **Versatilidad:** el proceso que haga el usuario debe ser accesible para cualquier persona, es decir no debe requerir conocimientos sobre análisis de imágenes, ni instrumental específico.
- **Fiabilidad:** los resultados obtenidos deben tener un bajo error, puesto que los resultados determinan la salud de las colmenas.
- **Económico:** el sistema debe realizar el análisis de la imagen realizada con cualquier smartphone, por que no se requiere ningún instrumental específico. Además, puesto que ahora se realiza a mano, se tardan horas, lo que se traduce en un gasto de personal.

3.1 Identificación y análisis de posibles soluciones

Después de analizar e identificar el problema a resolver, se pueden plantear las posibles soluciones que permita realizar el conteo de las varroas en las imágenes.

Una posible solución podría estar basada en la librería de código abierto, OpenCV, que proporciona un conjunto de herramientas para realizar el procesado de imágenes y la obtención de objetos de ésta, y posteriormente utilizar algoritmos de clasificación o *clustering*, para obtener una separación de los datos de los objetos.

Otra posible solución sería utilizando la misma librería que se comentó en la primera posible solución, pero en este caso utilizar redes neuronales para realizar el procesamiento de los datos, y su clasificación, o utilizar el proyecto darknet [1]

[1] https://pjreddie.com/darknet/

el cual ya implementa un sistema de identificación de objetos. Pero para esto sería necesario tener una gran cantidad de datos para poder entrenar de manera satisfactoria la red neuronal y realizar posteriormente pruebas.

3.2 Solución propuesta

La solución propuesta consiste en la utilización de una biblioteca de código abierto y multiparadigma que facilite el procesado de las imágenes. Tras una revisión de las bibliotecas que podríamos utilizar, OpenCV fue la elegida debido a su gran número de herramientas útiles para este proyecto, su facilidad de uso y su gran documentación y comunidad. Con esto podremos procesar las imágenes, y obtener los objetos de ésta para así extraer las características de cada uno de los objetos.

Para la posterior clasificación se ha escogido utilizar algoritmos de *clustering* los cuales permitirán hacer una agrupación de los datos obtenidos anteriormente. Para ésto se ha optado por utilizar la librería Sklearn de Scikit, ya que es un proyecto que está en continuo desarrollo y además cuenta con una documentación detallada y con ejemplos para el uso de los algoritmos que incluye.

Por último, para la interacción con el usuario se va a realizar una pagina web, la cual se comunicará por HTTP, con el servidor de procesamiento que recibirá la imagen subida en la web, para procesarla, señalar los puntos y devolver esta imagen señalada y el número de puntos encontrados, para posteriormente permitir visualizar y descargarla desde la web.

CAPÍTULO 4
Diseño de la solución

En éste capítulo se va a exponer la arquitectura del sistema y los distintos componentes del sistema para tener una idea del diseño de la interacción entre ellos, y se focalizará en la arquitectura *software* a nivel de proyecto.

4.1 Arquitectura del sistema

Después de establecer los requisitos del sistema que se desarrollará, se realiza el diseño de la solución al problema que plantea este proyecto. Previo a la explicación mas detallada de cada uno de los componentes, se va a hacer una breve introducción sobre cómo se ha dividido el sistema y como interactúan entre sí.

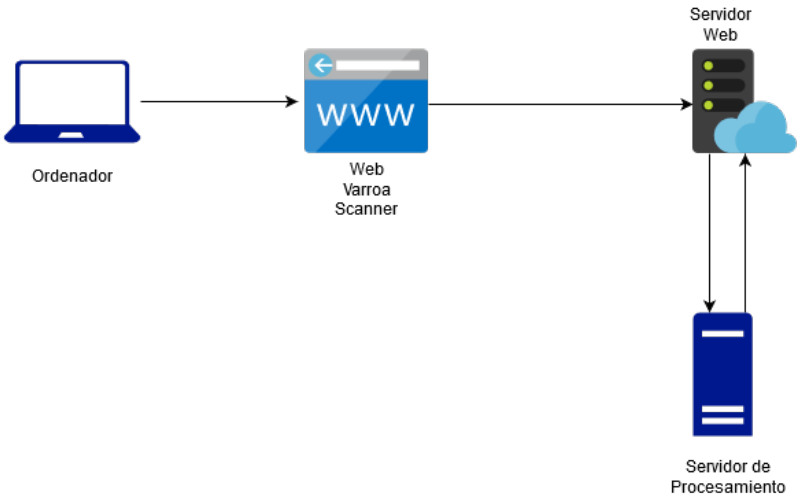

Figura 4.1: Diagrama del sistema de procesamiento

Según se indica en la imagen 4.1, sistema está compuesto por los siguiente componentes:

- **Ordenador:** se trata del ordenador del usuario, en este caso del apicultor, desde el cual accederá a la página web para subir la imágen y visualizar los resultados.
- **Web Varroa Scanner:** es la página web a la que accederán los usuarios para poder utilizar el sistema de escaneo y conteo.
- **Servidor web:** es el servidor donde estará alojada la web comentada en el punto anterior.
- **Servidor de procesamiento:** es el servidor que recibe la imagen y se encarga de hacer el procesado y conteo, para después devolverla al servidor web y mostrarla.

Una vez comentados estos puntos, se va a proceder a una explicación más detalla de cómo se ha diseñado esta arquitectura. Esta explicación se va a dividir en dos bloques, uno que abarca los puntos relacionados con la Web, y el segundo se centrará en el diseño del sistema de procesamiento de la imagen.

4.2 Diseño de la Web

Según el esquema de la figura 4.1, se va a proceder a explicar la web "Varroa Scanner" y el servidor web. El ordenador es cualquier dispositivo con acceso a Internet para que pueda acceder a la web.

La página web cuenta con una primera página de presentación donde se expone el problema de las varroas y como este sistema pretende ayudar. Desde ella podrán visualizar ésta memoria, por si quieren conocer más afondo sobre el proyecto, un contacto de quien ha realizado éste proyecto, y un enlace a la siguiente página donde se encuentra el escáner en sí.

Figura 4.2: Página de inicio de la Web

Y una segunda página en la que se ubica el escáner, al cual se arrastra la imágen a analizar, y además se puede seleccionar las propiedades que se quieran tener en cuenta a la hora del análisis, para conseguir distintos porcentajes de precisión y distintos resultados:

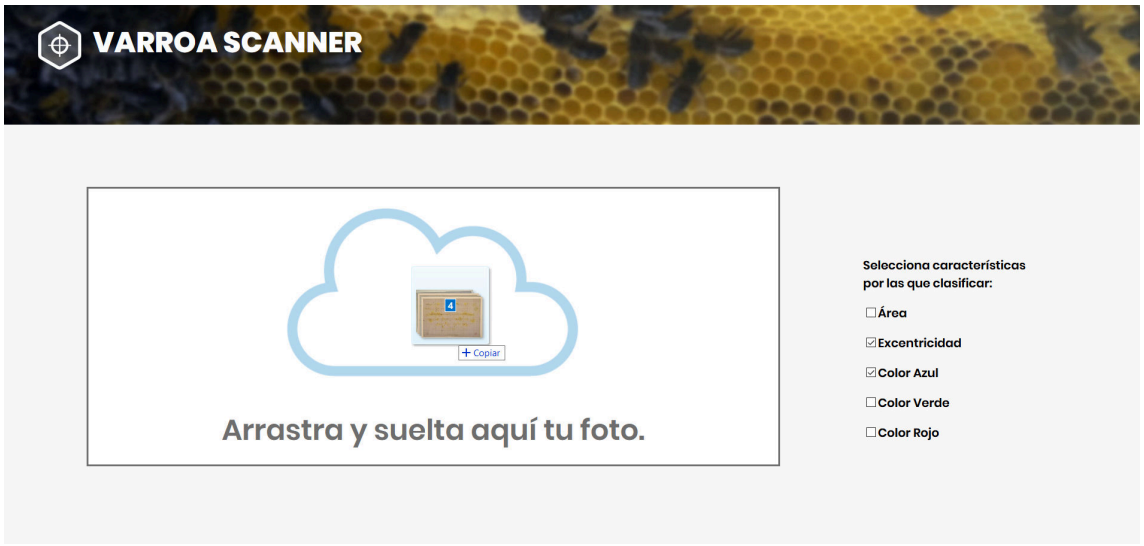

Figura 4.3: Página del escáner de la Web

4.3 Diseño del servidor de procesamiento

En este apartado se explicará el servidor de procesamiento que aparece en la figura 4.1, el cual se encarga de realizar el procesado de la imagen, clasificación de los puntos encontrados en ella y por último devolver al servidor web los resultados que se han obtenido.

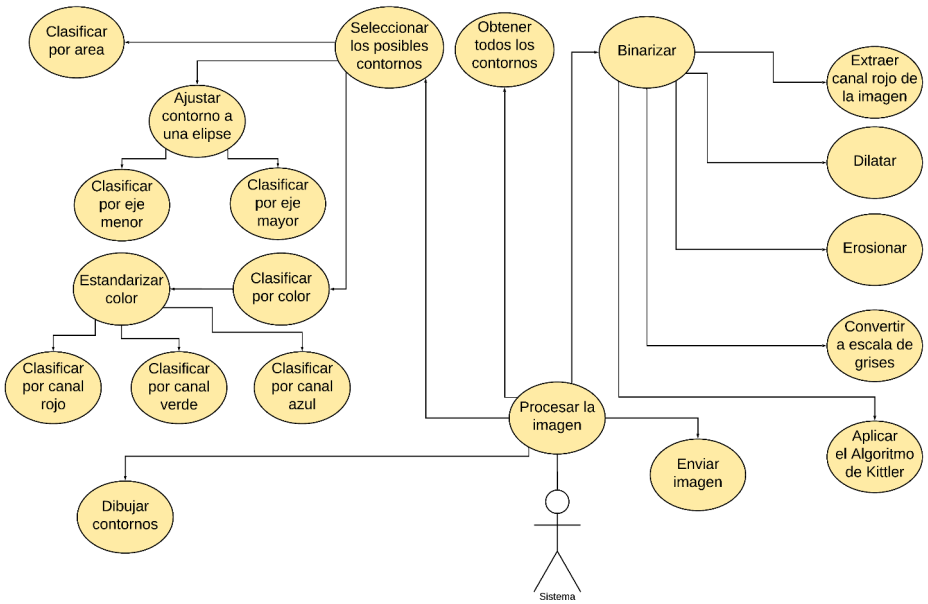

Figura 4.4: Diagrama del servidor de procesamiento

4.4 Tecnología Utilizada

En este apartado se hará una revision de los lenguajes de programación utilizados, las librerías que se han usado tanto para el tratamiento de la imagen como para la segmentación de los datos, y desarrollo del servidor web, y por último las herramientas que se han usado para desarrollar todo lo anterior mencionado.

4.4.1. Servidor de procesamiento

En esta primera sección se va a analizar la parte del servidor que se encarga del procesamiento y análisis de la imagen de entrada.

Python

Python es un lenguaje de programación interpretado, orientado a objetos y multiplataforma. El código fuente es abierto, por lo que cualquiera puede modificarlo. Fue creado con la intención de que el código fuera legible y fácil de entender, para ello utiliza sangrado en la declaración del código. Ofrece tipos de datos dinámicos, clases e interfaces predefinidas y multitud de bibliotecas para extenderlo.

Figura 4.5: Python

OpenCV

Figura 4.6: OpenCV

OpenCV es una librería de código abierto para la visión por ordenador y aprendizaje automático. Posee mas de 2500 algoritmos optimizados que incluyen un conjunto muy completo tanto de algoritmos clásicos como de última generación, puesto que la biblioteca tiene soporte constante.

Scikit learn

Scikit learn es un módulo de python que integra un amplio número de algoritmos de aprendizaje automático actuales tanto para problemas supervisados como no supervisados. Se centra en la facilidad de uso, rendimiento, documentación y mantener una API[1] consistente.

Figura 4.7: Scikit Learn

[1] API: Una API es un conjunto de definiciones y protocolos para desarrollar e integrar el software de las aplicaciones

KMeans Uno de los algoritmos que integra este módulo, y que ha sido útil para el desarrollo de este proyecto es KMeans, que consiste en tratar de separar las muestras en n grupos de igual varianza. Requiere que se especifique el número de conjuntos en los que se quiere dividir las muestras, a partir de ahí, de forma aleatoria, selecciona los centros desde los que partirá cada uno de los grupos, posteriormente asigna cada dato al centro mas cercano y se va actualizando el centro a la media aritmética de las posiciones de los datos que se han asignado a ese grupo. Este algoritmos tiene gran capacidad de escalado en cuanto al número de muestras.

DBScan Otro algoritmo que también incluye esta biblioteca y que se ha utilizado alternativamente es DBScan, detecta automáticamente el número de agrupaciones. El componente central de DBScan es el núcleo de muestras, que son muestras que están en áreas de alta densidad. Por lo que la agrupación sería un conjunto de núcleos de muestras que estén cerca y las muestras cercanos a estos. La forma de medir la cercanía es la epsilon, la cual hay que especificarla, junto con el número mínimo de muestras que indica a partir de cuantos datos se considera un núcleo de muestras.

Affinity Propagation Y por último tambien se ha utilizado el algoritmo Affinity Propagation, que crea conjuntos de muestras mandando mensajes entre pares, estos representan si una muestra puede ser un ejemplo de otra, y devuelve un mensaje actualizando los valores de otros pares. No requiere que se especifique el número de conjuntos. Por ello se requiere que se especifique cuantas muestras de ejemplo debe usar, y un factor reductor de error, que indica la variabilidad de los valores de las muestras para evitar la disparidad entre muestras que pertenecen al mismo conjunto.

4.4.2. Servidor Web

En esta sección se va a explicar las tecnologías que se han utilizado para la creación de la pagina web, la cual servirá de portal para la subida de las fotos que serán enviadas al servidor anteriormente mencionado.

HTML

HTML[2] es un lenguaje que se utiliza en el desarrollo de páginas webs para describir, principalmente, la estructura de estas. Consiste en un conjunto de elementos que indican al navegador como mostrar el contenido. Los elementos se representan mediante etiquetas, estas no se muestran en la página, pero se usan para construir el contenido. Cuando se carga la web el navegador interpreta el código y dibuja los elementos, esto conlleva que las webs hechas con HTML puro son estáticas, es decir solo cambian si se modifica el programador la modifica. Actualmente se encuentra en las version 5, que es la que utilizará en

[2]HyperText Markup Language: Lenguaje de Marcas de Hipertexto

el proyecto, puesto que aporta cambios respecto a las versiones anteriores que permiten que la web sea *responsive* o "sensible a los cambios de tamaño".

CSS

CSS[3] es un lenguaje que se utiliza para indicar como se deben mostrar en la pantalla los elementos del HTML. Permite controlar el estilo de múltiples paginas web en una sola hoja de estilo. Actualmente se utiliza la version 3, que ademas de toda la sintaxis y funcionalidades de versiones anteriores, incluye nuevas funcionalidades para poder crear una pagina *responsive* que permite que se muestre bien en todo tipo de dispositivos.

JavaScript

JavaScript es un lenguaje de programación interpretado, orientado a objetos y con un tipado dinámico, es decir el tipo está asociado al valor no a la variable. Es conocido por ser utilizado como lenguaje de scripting en paginas webs, aunque tambien es usado en muchos otros entornos fuera de los relacionados con el navegador, como Node.js [4]. Aplicado a la web permite crear contenido nuevo y dinámico, mejorando asi la interfaz de usuario y la interacción con este.

JQuery

jQuery[5] es una librería de JavaScript que permite, a la vez, enriquecer y simplificar el uso de JavaScript en entornos Web. Facilita la manipulación del CSS y el HTML, asi como el manejo de eventos.

4.4.3. Interconexión entre sistemas

En esta sección se analizará la herramienta que se ha utilizado para interconectar la pagina web con la parte del servidor de procesamiento, y viceversa

Flask

Flask[6] es un *framework*, o "entorno de trabajo", es decir, un conjunto de herramientas y funcionalidades, el cual está escrito en python y permite crear aplicaciones webs de forma sencilla y compacta, con la habilidad de poder escalarse

[3] Cascading Style Sheets: Hoja de estilo en cascada
[4] Node.Js: mas información: https://nodejs.org/es/
[5] jQuery: https://jquery.com/
[6] Flask: mas información: https://flask.palletsprojects.com/

hasta crear aplicaciones complejas. Está basado en Werkzeug[7] y Jinja[8], y se ha convertido en uno de los frameworks de aplicaciones web en python mas populares.

4.4.4. Herramientas

En esta sección se van a describir las herramientas que se han utilizado durante el desarrollo de este proyecto, tanto en la parte del servidor web como en el de procesamiento, asi como el sistema de almacenamiento y control de versiones que se ha usado para dar respaldo al proyecto al completo.

Microsoft Visual Studio Code

Visual Studio Code[9] es un editor de código liviano pero potente el cual se puede ejecutar en cualquier dispositivo de escritorio. Está desarrollado en JavaScript, TypeScript y Node.js, además incorpora un gran número de extensiones para dar soporte a otros lenguajes y sistemas de ejecución. Por lo que permite al programador desarrollar código de una manera sencilla y rápida gracias, a por ejemplo, resaltado de sintaxis, finalización inteligente de código, refactorización...

Jupyter Notebook

Jupyter Notebook, anteriormente llamado IPython Notebook, es una aplicación web de código abierto que permite crear y compartir documentos que contengan código vivo, ecuaciones, visualizaciones y texto narrativo. Soporta más de cuarenta lenguajes, entre ellos Python. Esta herramienta ayuda a crear salidas interactivas. Un documento de jupyter notebook es un documento JSON, que contiene una lista completa de las sesiones de usuario y de las celdas de entrada y salida que pueden contener código, texto, gráficos... Se puede convertir a varios formatos de salida, entre ellos HTML, PDF, Python... Al usuario se le muestra a través del nbviewer[10] que va convirtiendo el documento a HTML en tiempo real, es decir mientras el usuario lo está editando, gracias a nbconvert[11], para mostrárselo en un navegador web a través de una URL. Jupyter Notebook se puede conectar a varios núcleos, aunque por defecto, y para el ámbito de este proyecto, utiliza

[7] Werkzeug: mas información: https://palletsprojects.com/p/werkzeug
[8] Jinja: mas información: https://palletsprojects.com/p/jinja
[9] Visual Studio Code: mas información: https://code.visualstudio.com/
[10] nbviewer: mas información: https://nbviewer.jupyter.org
[11] nbconvert: mas información: https://nbconvert.readthedocs.io/

IPython.

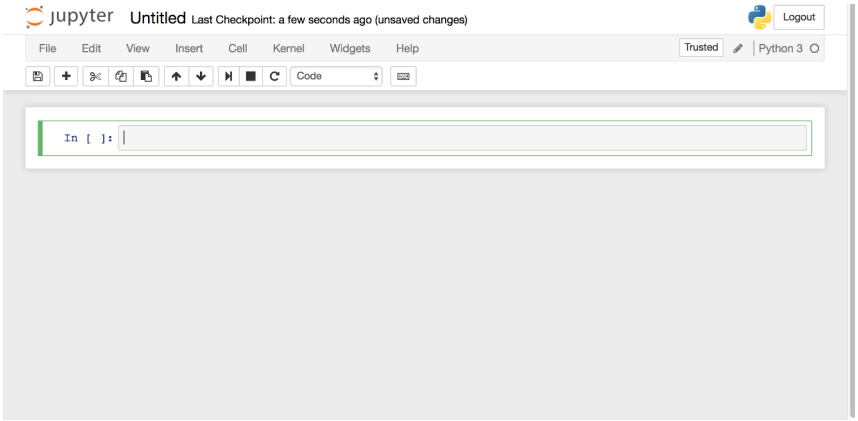

Figura 4.8: Notebook de Jupyter en blanco

GitHub

Github es una plataforma web de desarrollo colaborativo para almacenar proyectos utilizando el sistema de control de versiones Git[12] y gestionar proyectos. Además es uno de los repositorios online mas grandes de trabajo colaborativo del mundo, que recientemente fue adquirido por Microsoft, lo que nos indica que será mantenido y actualizado durante muchos años.

[12]Git: software de control de versiones. Cuyo propósito es registrar los cambios en archivos y coordinar el trabajo que los desarrolladores hacen sobre estos archivos. Mas información: https://git-scm.com/

CAPÍTULO 5
Desarrollo de la solución propuesta

5.1 Materiales Utilizados

Para la realización de este proyecto se han necesitado imágenes de las planchas, las cuales nos la ha proporcionado el tutor externo, "Miguel Angel Garrido" con la ayuda de "Apiads", la Agrupación de defensa sanitaria apícola.

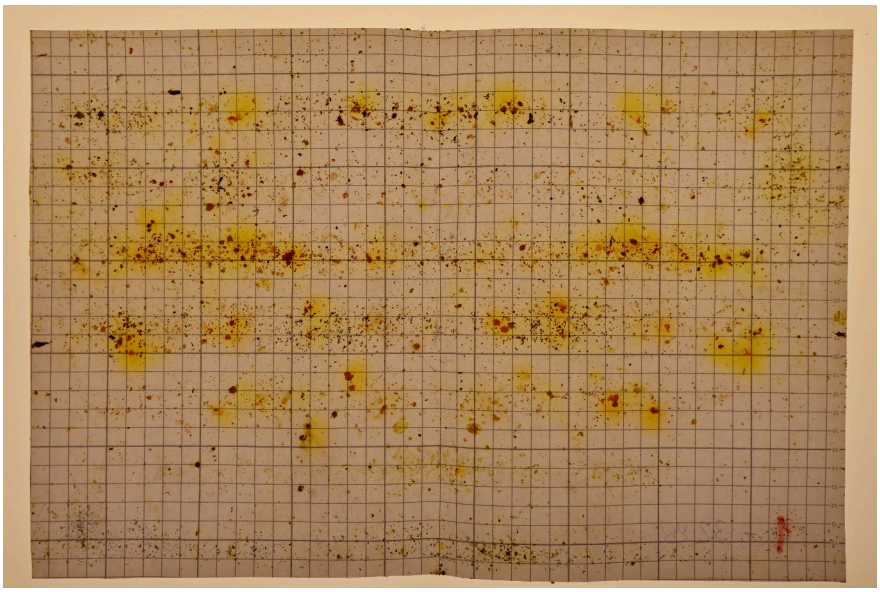

Figura 5.1: Plancha uno

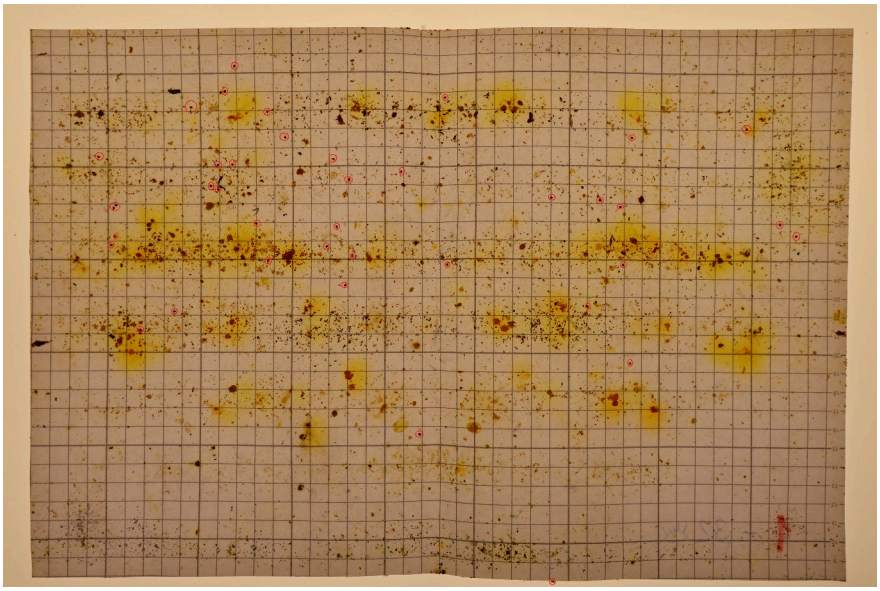

Figura 5.2: Plancha uno con las varroas señaladas

El resto que se han utilizado están en el anexo A.

Para el desarrollo del sistema se necesitaban tanto las imágenes de las planchas, en la cuales se vieran con calidad y luminosidad las varroas y también la misma imagen pero con las varroas señaladas para probar la eficacia del sistema cuando se desarrollaban los diferentes experimentos.

5.2 Implementación del servidor de procesamiento

En este apartado se van a comentar los pasos que se siguen para el procesado de la imagen:

5.2.1. Binarización de la imagen

El primer paso es binarizar la imagen es decir, reducir la información de la imagen a negro o blanco es decir, verdadero o falso, donde el 0 es representado por el color negro y el 1 por el color blanco. Para ello hay que buscar un valor de corte, o umbral, es decir hasta qué nivel de color se va a considerar como 0 y hasta cual se tomará como 1. Esto se consigue con diferentes algoritmos, para nuestro caso hemos escogido el algoritmo del mínimo error de Kittler e Illingworth.

Para conseguir ésto, antes de aplicar el algoritmo y para que este genere mejores resultado, se han realizado pasos previos.

Primero nos quedamos solamente con el canal rojo de la imagen, puesto que las varroas tienen un alto nivel de rojo y por ello será mas probable que se tome como 1 al binarizarla.

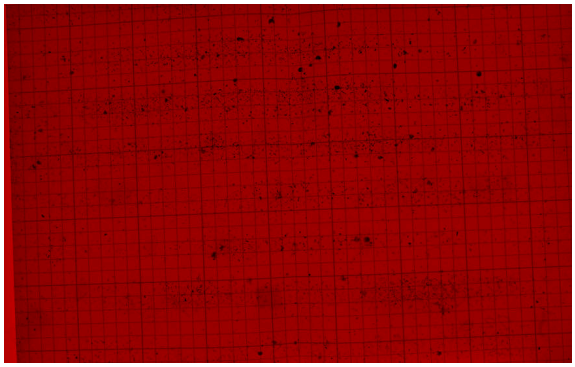

Figura 5.3: Solo con canal rojo

Posteriormente se dilata y erosiona la imagen:

- **Dilatar:** Ajustar para cada pixel el nivel de color al nivel mas bajo de los 24 píxeles de alrededor, ya que se ha utilizado una matriz de 5x5.
- **Erosionar:** Es lo contrario a dilatar, es decir, ajustar el nivel de color de cada píxel al nivel más alto de los 24 píxeles circundantes.

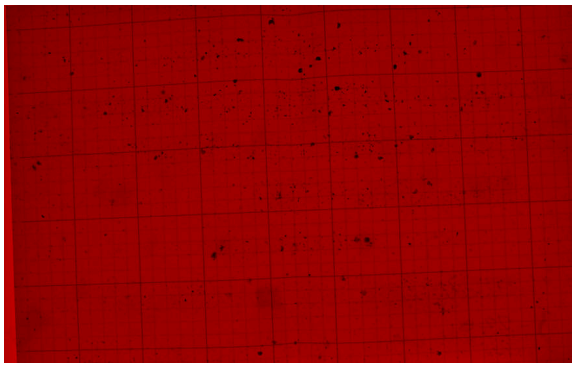

Figura 5.4: Dilatada y erosionada

Y por ultimo se aplica el algoritmo del mínimo error de Kittler e Illingworth[7], que consiste en conseguir una estimación probabilística de la función de densidad a partir del histograma de la imagen. esta función de densidad es la union de dos distribuciones, una el *foreground*, o primer plano, y el *background*, o fondo, que se obtienen de:

$$p_f(t) = \frac{1}{\sigma_f\sqrt{2\pi}} e^{-\frac{(t-\mu_f)^2}{2\sigma_f^2}} \qquad p_b(t) = \frac{1}{\sigma_b\sqrt{2\pi}} e^{-\frac{(t-\mu_b)^2}{2\sigma_b^2}}$$

Función de densidad del foreground Función de densidad del background

La función densidad de probabilidad del histograma es:

$$p(t) = P_f(t)p_f(t) + P_b(t)p_b(t)$$

Y el error mínimo se alcanzará cuando seleccionemos un umbral (t) para el cual:

$$P_f(t)p_f(t) = P_b(t)p_b(t)$$

Y aplicando logaritmo en ambas partes, resolviendo y sustituyendo μ_f, μ_b, σ_f^2 y σ_b^2 obtenidos de:

$$\mu_f(t) = \sum_{g-1}^{t} g \frac{p(g)}{P_f(T)}$$

$$\mu_b(t) = \sum_{g-t+1}^{255} g \frac{p(g)}{P_b(T)}$$

$$\sigma_f^2(t) = \sum_{g-1}^{t} (g - \mu_f)^2 p(g)$$

$$\sigma_b^2(t) = \sum_{g-t+1}^{255} (g - \mu_b)^2 p(g)$$

Se puede definir una función criterio que permita encontrar el valor mínimo:

$$J(t) = 1 + 2(P_f(t) \cdot log\sigma_f(t) + P_b \cdot log\sigma_b(t)) - 2(P_f(t) \cdot logP_f(t) + P_b \cdot logP_b(t))$$
$$T^* = Min\{J(t)\}$$

Al realizar la binarización se obtiene la siguiente imagen 5.5:

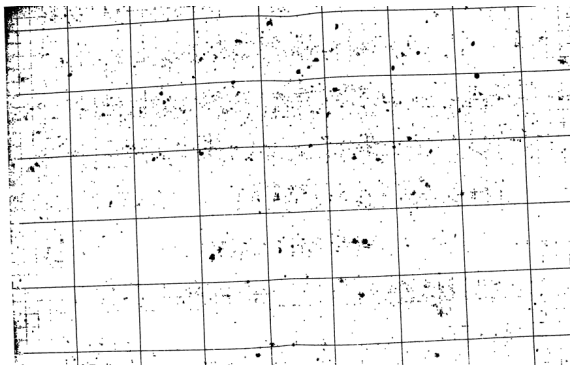

Figura 5.5: Binarizada

5.2.2. Eliminar líneas

Debido a que las imágenes de las planchas tienen una rejilla, hay que eliminarla para que cuando se obtengan los contornos esta no aparezca. Para ello se ha utilizado la transformación de Hough para detectar y eliminar las líneas. La transformada de Hough es utilizada para detectar figuras que puedan ser expresadas matemáticamente. El proceso a seguir es el siguiente:

- **Primer paso:** Obtener los puntos de la frontera de los objetos de la imagen. Esto se ha obtenido con la función *Canny*, que consiste en obtener los bordes mediante la primera derivada en la dirección horizontal y vertical de la imagen y a partir de ésto, se obtiene el gradiente de borde y la dirección. [8]

- **Segundo paso:** Se realizan posibles agrupaciones de puntos que pertenezcan a los bordes de las figuras estimadas mediante un sistema de votación, donde nosotros establecemos el mínimo número de puntos que debe tener esa figura para que sea considerada una una línea.

Como lo que queremos obtener son rectas tendremos que utilizar la ecuación:

$$x\cos\theta + y\sin\theta = \rho$$

La cual define una recta en coordenadas polares. Y para obtener la transformada de hough se coge el rango $\theta \in [0, 2\pi]$ y utilizando la ecuación anterior y los puntos (x,y) de la imagen se obtiene el espacio de Hough definido por (ρ, θ).

Entonces si dos puntos se interceptan, el punto de intersección en el espacio de Hough será una línea que pasa por esos dos puntos.

El resultado que se obtiene tras aplicar primero el algoritmo de Kittler e Illingworth, y posteriormente al resultado de este la transformada de Hough es 5.6:

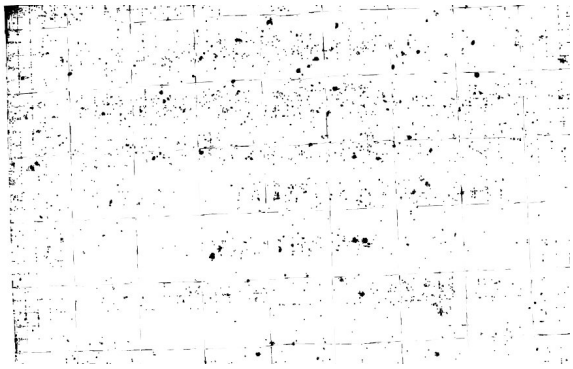

Figura 5.6: Transformada de Hough

5.2.3. Obtención de contornos

Una vez procesada la imagen y obtenida de forma binaria, se procede a buscar todos los objetos que hayan en esta, esto se hace buscando sus contornos, es decir una curva que una los puntos continuos que tengan el mismo color o intensidad. Con esto podemos analizar estos objetos y extraer sus características para así poder clasificarlos.

OpenCV, incluye una función a la cual se le pasa una imagen, preferiblemente binaria, y te busca y devuelve todos los contornos que encuentre en ella. A esta función también se le indica el modo de recuperación de los contornos, en este caso se ha utilizado "RETR TREE" que obtiene todos los contornos y los reconstruye con la jerarquía del contorno aninado; y el método de aproximación, que nosotros hemos elegido "CHAIN APPROX SIMPLE " el cual comprime los segmentos horizontales, verticales, y diagonales y deja solo los puntos que definen la curva del contorno.

Por lo que si aplicamos esta función sobre la imagen obtenida en el apartado anterior, y los dibujamos sobre ella obtenemos la siguiente imagen:

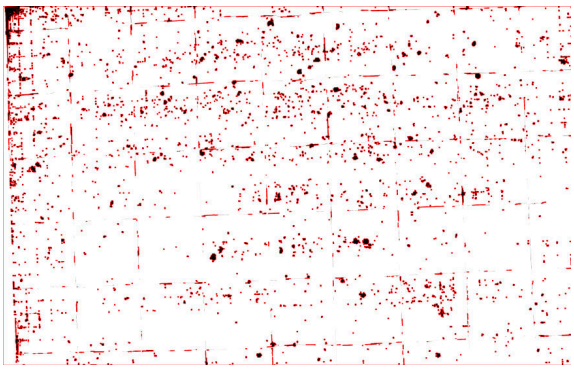

Figura 5.7: Contornos sobre la imagen binaria

Como vemos en rojo está marcado el contorno de cada objeto que se ha detectado. Pero al hacerlo sobre la imagen en binario se pierde información, como

el color, que para este proyecto es una de las características importantes, por ello una vez obtenidos los contornos a partir de la imagen binaria, estos se pueden trasladar a la imagen original en color, quedando, si los dibujamos, de la siguiente manera 5.8:

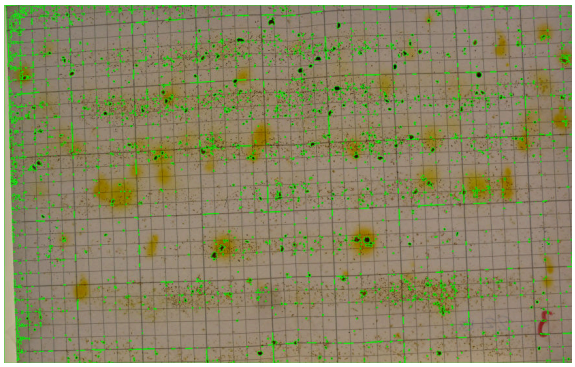

Figura 5.8: Contornos sobre la imagen original

A partir de esto ya se puede proceder a extraer todas las características de los objetos detectados, para su estudio y posterior clasificación.

5.2.4. Estudio de características

En este capítulo se va a comentar las características que han extraido de los objetos, así como una breve explicación de como se han obtenido.

- **Area:** es el área contenida dentro del contorno, es decir el área del objeto. OpenCv proporciona una función que dado un contorno devuelve como resultado el área de éste en píxeles.

- **Semieje Menor:** es el tamaño en píxeles del semieje menor de la elipse que se puede ajustar al contorno. OpenCv proporciona una función que permite realizar ésto, dado un contorno devolver una elipse en forma de array donde el primer elemento es una tupla de las coordenadas del centro de la elipse, el segundo elemento es una tupla con la longitud del semieje menor y el semieje mayor, y el tercer elemento es el ángulo de rotación.

- **Semieje Mayor:** es el tamaño en píxeles del semieje mayor de la elipse que se ajusta al contorno. Se obtiene igual que en el semieje menor, pero accediendo al segundo elemento de la tupla donde están las longitudes de los semiejes, es decir, el segundo elemento del array.

- **Excentricidad:** es la desviación de una sección cónica, la elipse, con respecto a una circunferencia. Este parámetro es útil en nuestro análisis ya que las varroas tienen una forma elipticia, y todas tienen una forma y un tamaño muy similar, en cambio otros elementos que caen, como puede ser el polen es circular. Esta característica se obtiene a partir de los dos semiejes: $e = \frac{\sqrt{(a^2-b^2)}}{a}$ donde a es el semieje mayor y b el menor

- **Nivel de color:** es el nivel de color de cada canal que tiene el objeto, para simplificar la extracción se ha tomado el color del centro, ya que las varroas tienen un color uniforme. Esto se ha obtenido primero obteniendo el centro del contorno gracias a la elipse, como se ha comentado anteriormente, y posteriormente sobre la imagen se consulta éste pixel y se obtiene un array con el nivel de color de los tres canales(azul, verde y rojo).
- **Desviación del nivel de color**: es la variación de color que existe dentro de un contorno, es decir, si el color del objeto es uniforme o no. Como se ha comentado, las varroas tienen un color uniforme, por lo que esta es una variable interesante de estudiar. Se obtiene calculando la desviación estándar de todos los niveles de color de los píxeles de un contorno para cada canal.

5.2.5. Aplicación de reglas de clasificación

Una vez extraidas todas las características de los objetos detectados en la imagen hay que etiquetarlos, ya que primero hay que identificar las características que describen a los objetos que son varroas. Para ello se ha requerido de la ayuda del tutor externo para que señalara en la imagen las varroas, quedando la imagen así:

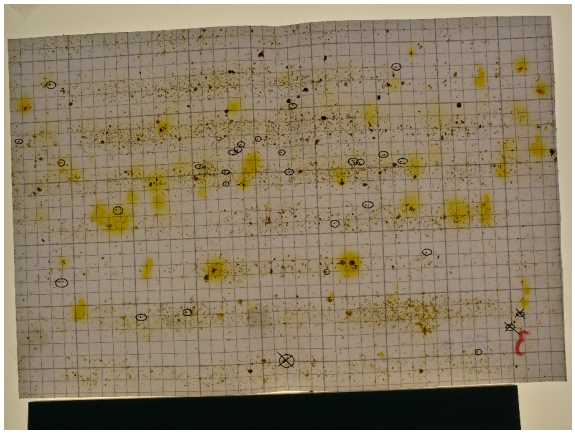

Figura 5.9: Varroas señaladas

Con la imagen ya señalada, se ha procedido a obtener manualmente, con la ayuda de un editor de imágenes en este caso se ha utilizado Photoshop, las coordenadas de estos puntos, para posteriormente buscarlo entre todos los puntos detectados y etiquetarlo como ”varroa”, mientras que el resto de puntos se marcaban como ”noVarroa”.

Una vez etiquetados todos los puntos, se ha procedido a representarlos tanto en gráficos XY como en XYZ, para ver la separación de los datos dependiendo de las características que se utilizan. Esto se detalla de forma más extensa y comentando varias pruebas que se han realizado en el capítulo de Evaluación 6.

5.2.6. Visualización de los resultados

Una vez que se ha clasificado, utilizando condicionales o mediante la predicción que realizan los algoritmos que se han utilizado, se obtienen las coordenadas de los puntos que deben ser etiquetados como "varroas", y nos quedamos solo con esos contornos. Posteriormente con la función "drawContours" de OpenCv, a la cual se le pasa como argumentos la imagen donde se dibujará y los contornos, además de otros argumentos como el color de la linea o el grosor, dibuja estos puntos y devuelve la cantidad.

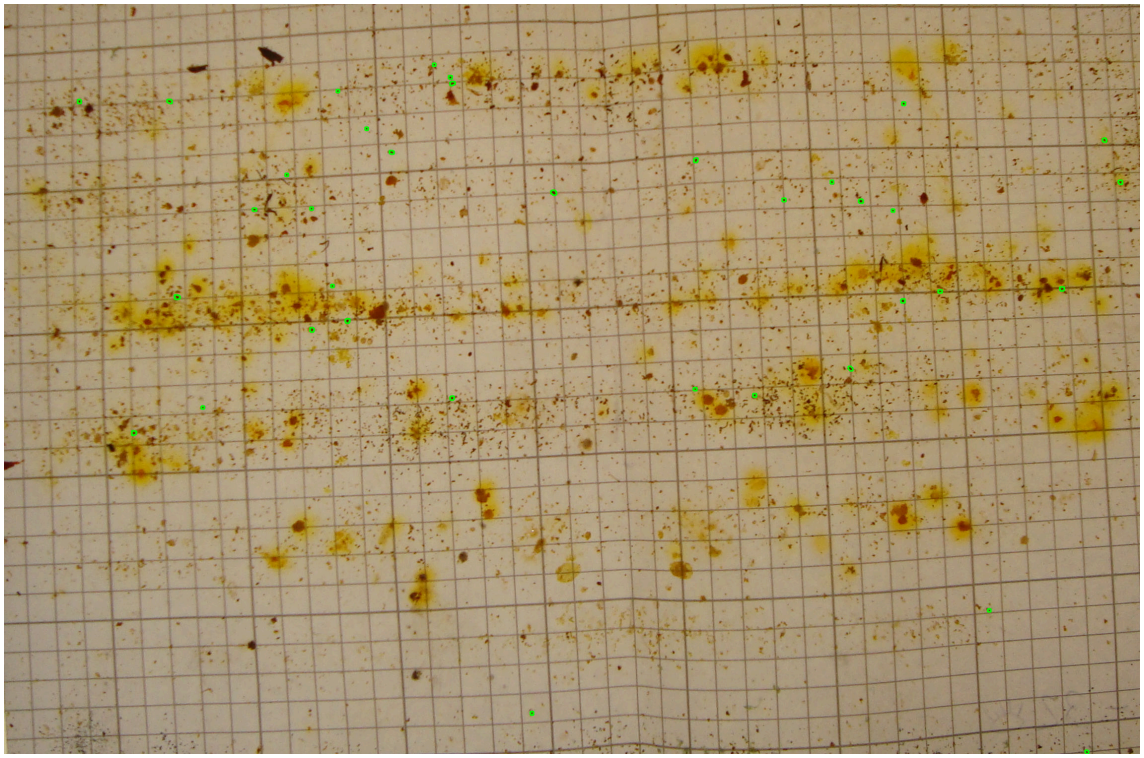

Figura 5.10: Varroas señaladas con condicional

5.3 Implementación de la web

En este apartado se va a comentar como está implementada la web desde la cual se accederá al escaner:

5.3.1. Mockups

Primero se realizó una fase de diseño, ya que se quería que la web fuera sencilla, fácil de usar y accesible desde cualquier dispositivo.

En la pagina principal se explica de forma brevé la intención del escáner y el problema de esta enfermedad:

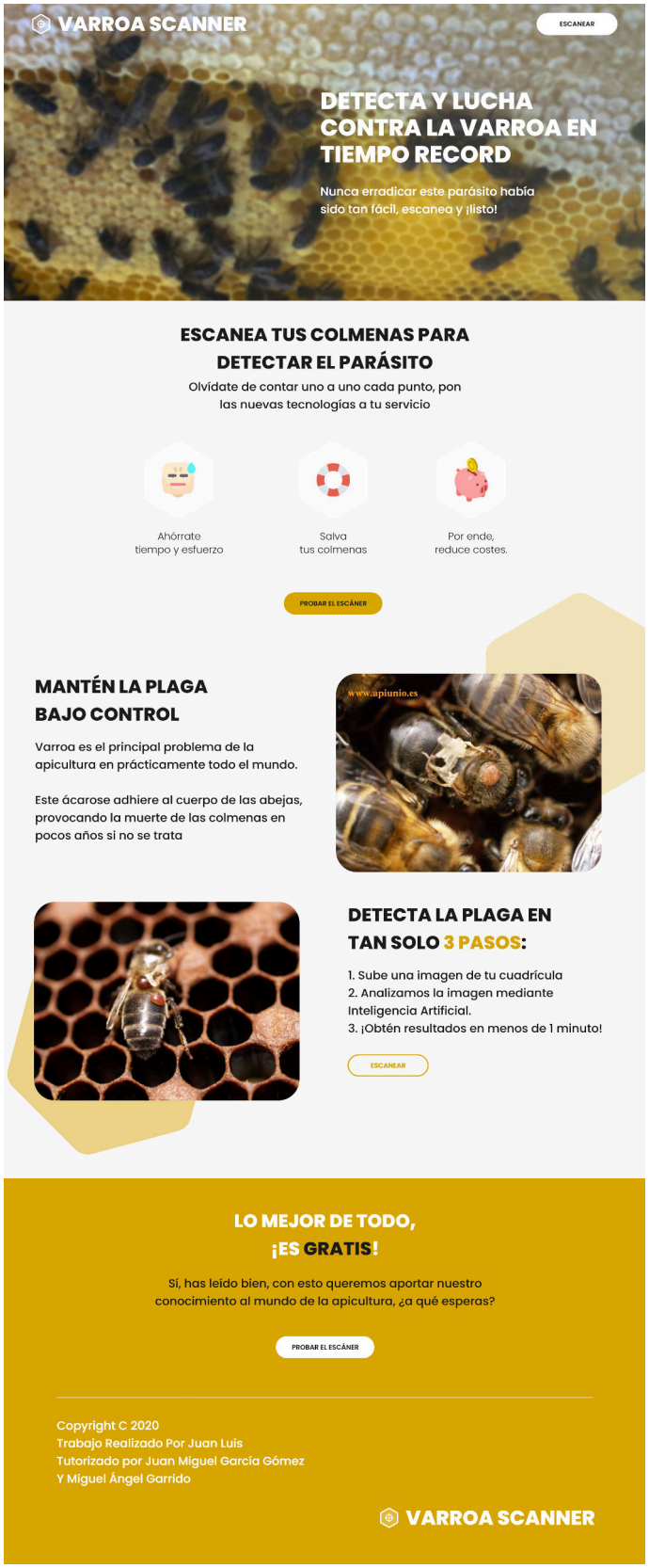

Figura 5.11: Pagina principal de la web

Posteriormente se realizó el diseño de la página del escaner, la cual tendría un cuadro *drag and drop* o arrastra y suelta, donde se subirían todas las imágenes que se deseen:

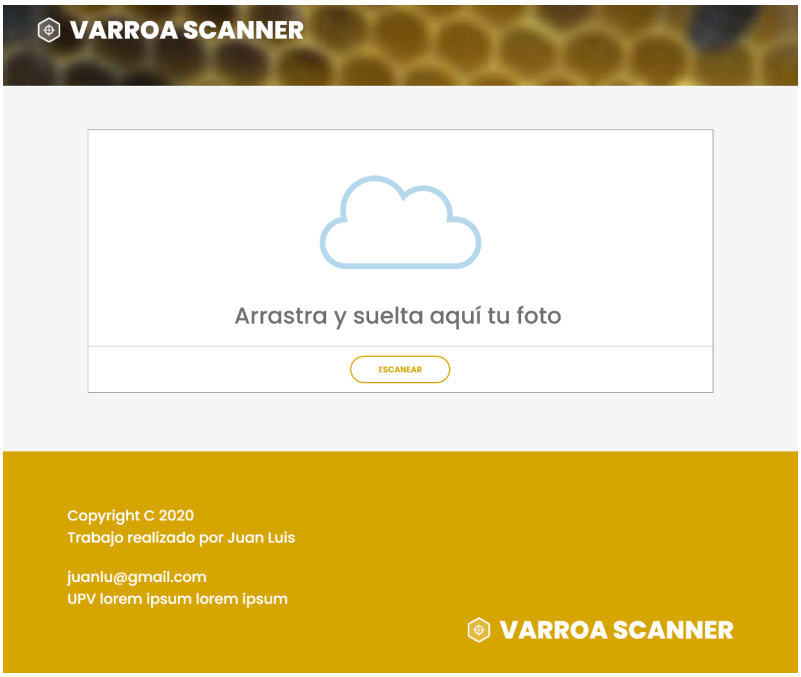

Figura 5.12: Pagina del escaner de la web

Y como se mostrarian los resultados obtenidos por el servidor de procesamiento:

Figura 5.13: Pagina del escaner con imagen procesada de la web

A partir de estos diseños ya se podía comenzar a maquetarlos y programar la web.

5.3.2. Implementación

La web se ha realizado en HTML para la creación de los componentes principales, a los que posteriormente se les ha ido añadiendo estilos mediante una hoja de estilos CSS, con lo que además de conseguir aplicar los diseños del mockup, también hemos podido hacer que la web sea *responsive*, o sensible a los cambios de tamaño.

Para la conexión con el servidor, es decir el envío de las imágenes a procesar y posteriormente recibir los resultados, se ha utilizado JavaScript en el lado de la web y Flask en la parte del servidor, que será el encargado de recibir las peticiones HTTP. JavaScript obtienen las fotos codificadas en base64 y las envía por el puerto 5000 mediante una petición "POST", gracias a la librería "Ajax" de JavaScript la conexión al servidor y el envío se realiza de forma sencilla.

La recepción de los resultados es un objeto JSON, compuesto por dos elementos, la imagen, que al igual que en el envío está codificada en base 64, y otro que es número de varroas que se ha encontrado.

CAPÍTULO 6
Evaluación

En este capitulo se realizará una revisión de los resultados que se han obtenido, comparando distintas técnicas e imágenes. Se han escogido algunos de los experimentos que se han realizado, los más representativos, ya que exponer todos haría que este apartado fuera demasiado extenso.

6.1 KMeans

Lo primero que probó tras la extracción de los datos de los objetos encontrados en la imagen es utilizar un algoritmo de clustering[1], KMeans, con dos clases, para separar lo que son varroas y lo que no lo son. Se hicieron distintas pruebas cambiando los datos de entrada, que se detallan a continuación:

6.1.1. Area y Canal Azul

En este caso se han utilizado el área de los objetos que se han encontrado en la imagen, y el nivel de color azul de ese objeto. Para esta prueba se ha utilizado la imagen de la plancha uno con todos los objetos que se han encontrado señalados(Figura 6.1)

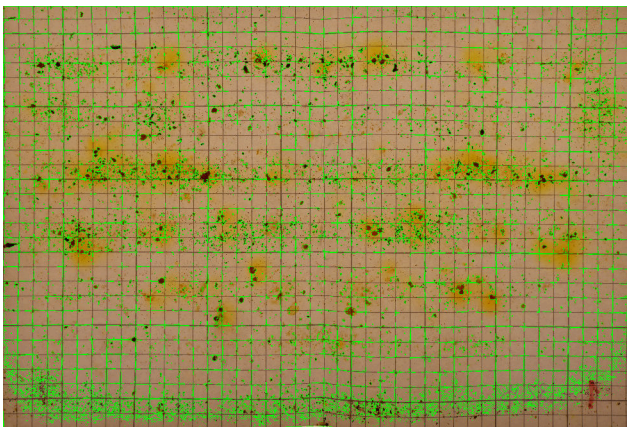

Figura 6.1: Contornos señalados

[1]Clustering: Agrupación automática de datos

Si sacamos una representación de los datos obtenidos en un gráfica obtenemos los que se muestra en la Figura 6.2. Donde los puntos naranjas son varroas y los triángulos azules otros objetos detectados.

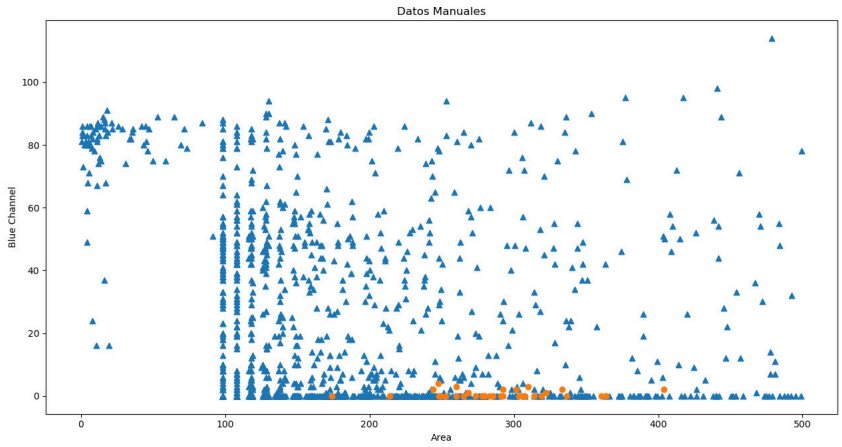

Figura 6.2: Área - Color Azul de la Imagen 1

Tras obtener los datos, se utiliza el algoritmo KMeans al que se le indica que divida el conjunto de datos en dos clases, para así obtener una clase que represente a las varroas y otra para todo lo demás. Una vez aplicado el algoritmo a los datos se obtiene las nuevas etiquetas de los que KMeans estima que debería tener cada punto. Donde la clase 1 son lo que estima que son varroas y la clase 0 todo lo demás, y con los triángulos azules se representa lo que son varroas reales.

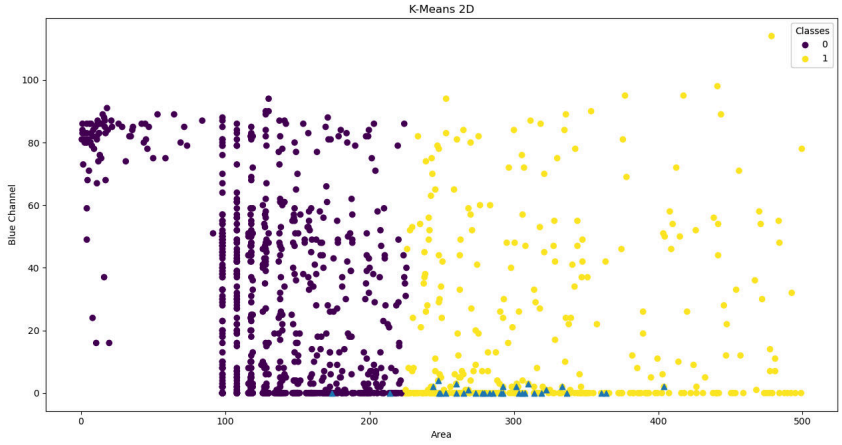

Figura 6.3: KMeans Área - Color Azul de la Imagen 1

Una vez extraidos estos datos podemos calcular la precisión y el recall [2] que tiene este método de clasificación.

[2]Recall: Cantidad total de elementos relevantes.

Clase	Precisión	Recall	Número de puntos estimados	Número de puntos reales
0	100 %	72 %	876	1216
1	8 %	94 %	372	32

Tabla 6.1: Resultados de Kmeans con 2 clases

Como se puede observar en la tabla 6.1 para la clase 1, las varroas, se obtiene un alto *Recall*, lo que nos indica que prácticamente la totalidad de las varroas son bien clasificadas en la clase correspondiente, pero por el contrario la precisión que obtenemos es muy baja, por lo que está metiendo muchos puntos que son irrelevantes en la clase de varroas. Esto se puede observar muy bien la siguiente figura 6.4

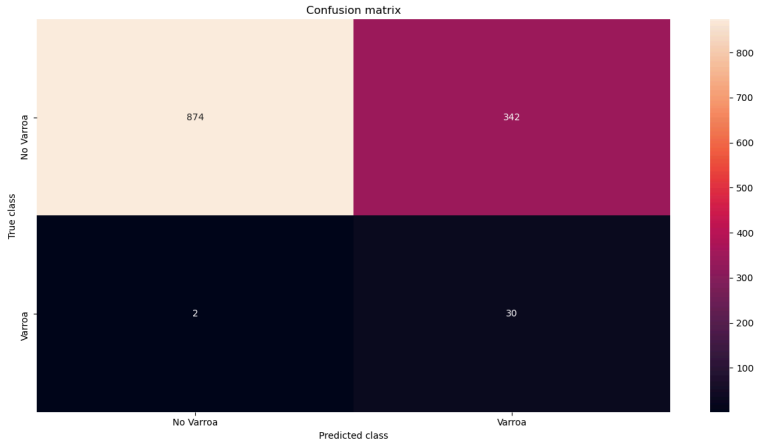

Figura 6.4: Matriz de Confusión Área - Color Azul de la Imagen 1 con el algoritmo KMeans

Donde el eje X representa los valores estimados y el eje Y los valores reales.

Para finalizar, comentar que se han escogido estas propiedades puesto que tras analizar los gráficos iniciales la propiedad área es de las más descriptivas, y combinada con cualquier de las demás propiedades se obtiene siempre, con ésta configuración, los mismos resultados.

6.1.2. Filtrado de datos y posterior clasificación por Área y Semi-eje menor

En esta prueba se ha realizado primero un filtrado de datos, es decir, se han eliminado aquellos datos que, tras observar los gráficos, se ha determinado que no son varroas. Esta prueba se ha realizado también con la imagen de la plancha número uno (Figura 6.11). El filtrado de datos que se ha realizado es usando el área, el color azul y el color rojo. Este filtrado se ha hecho de forma manual, es decir, observando el gráfico de estos datos representados y se han extraído los valores de corte. En la siguiente figura 6.5 se puede observar que región se ha

escogido para las propiedades de área y nivel de color del canal azul.

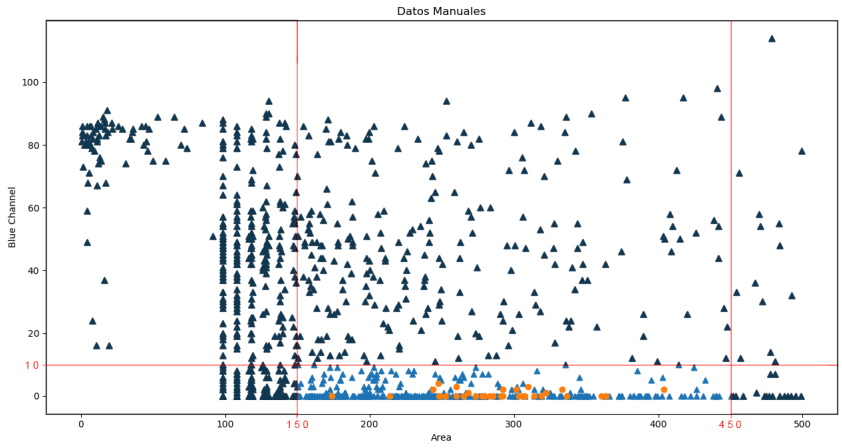

Figura 6.5: Región escogida de Área - Color Azul de la Imagen 1

Y en la figura 6.6 la región para el nivel de color del canal verde, y el área, que es la misma región que la anterior.

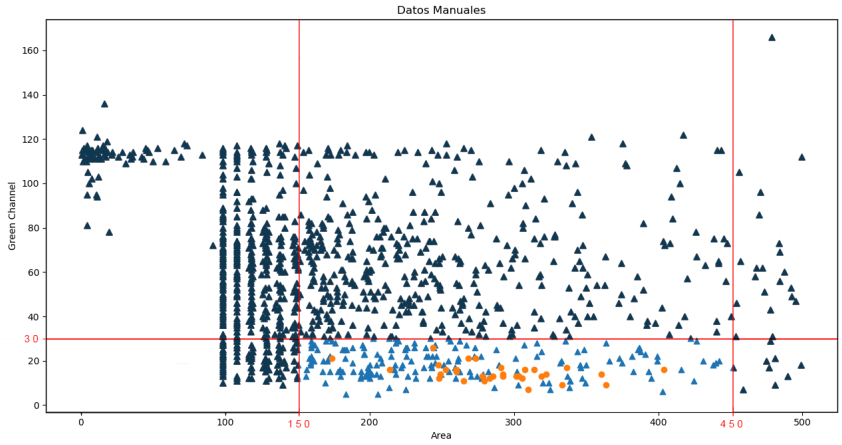

Figura 6.6: Región escogida de Área - Color Verde de la Imagen 1

Tras realizar este filtrado podemos analizar mas de cerca dónde se localizan los puntos pertenecientes a las varroas. Después de analizar varios gráficos con las distintas propiedades se decidió utilizar la combinación de área y valor del semi-eje menor de la elipse que enmarca los objetos detectados en la imagen, ya que las varroas tienen una forma elíptica y su tamaño suele ser fijo. En la figura 6.7 se puede observar como hay una región donde se acumulan practicamente la totalidad de las varroas, salvo algún punto más disperso que puede ser considerado error, puesto que al procesar la imagen puede que el contorno no se haya ajustado perfectamente al objeto.

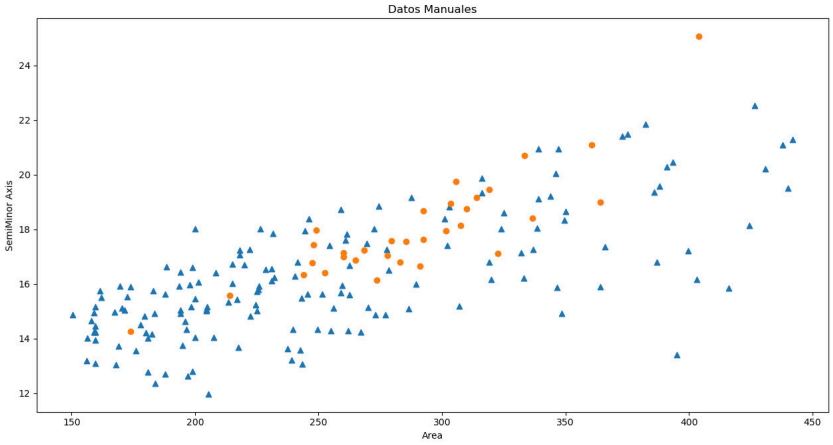

Figura 6.7: Área - Semieje Menor de la Imagen 1

Con dos clases

Después de aplicar el algoritmo kmeans con dos clases, nos queda la separación que se observa en la figura 6.8

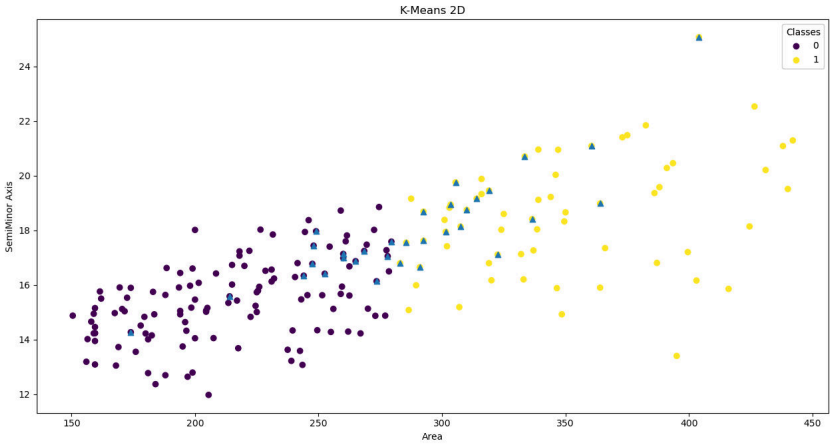

Figura 6.8: KMeans Área - Semieje Menor de la Imagen 1

Se puede observar que la división que realiza deja a muchos positivos en la clase cero, es decir en la clase de lo que no son varroas, ya que los triangulos azules representan los puntos que realmente son positivos. Si cuantificamos esto obtenemos la siguiente tabla 6.2:

Clase	Precisión	Recall	Número de puntos estimados	Número de puntos reales
0	89 %	71 %	125	157
1	28 %	56 %	64	32

Tabla 6.2: Resultados de Kmeans con 2 clases

De esta prueba se puede extraer que con esta solución se ha mejorado la precisión, aunque por el contrario el *recall* ha empeorado. Si observamos la matriz de confusión ilustrada en la figura 6.9 vemos que las varroas estimadas y que realmente son varroas ha disminuido considerablemente.

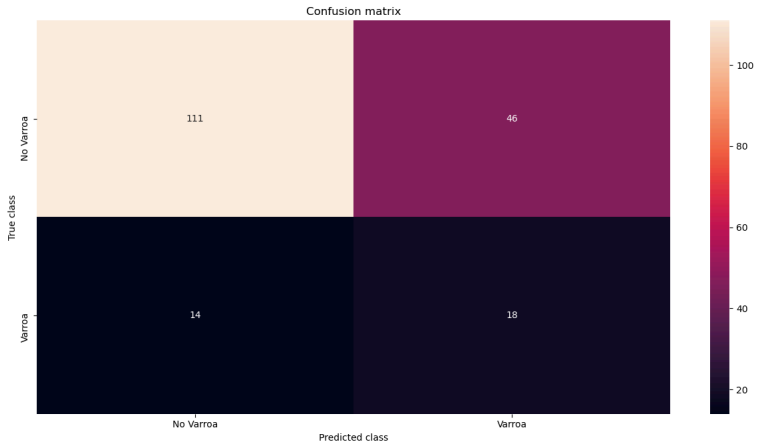

Figura 6.9: Matriz de Confusión de KMeans Área - Semieje Menor de la Imagen 1

Con tres clases

En esta nueva prueba, se han utilizado los mismos datos y mismo algoritmo que en subapartado anterior 6.1.2 pero en este caso al algoritmo, kmeans, se le ha indicado que agrupe los datos en 3 subconjuntos, por lo que se considerará una de las tres clases varroas, y las dos restantes como irrelevantes, es decir no varroas. Con ésto observamos la siguiente distribución:

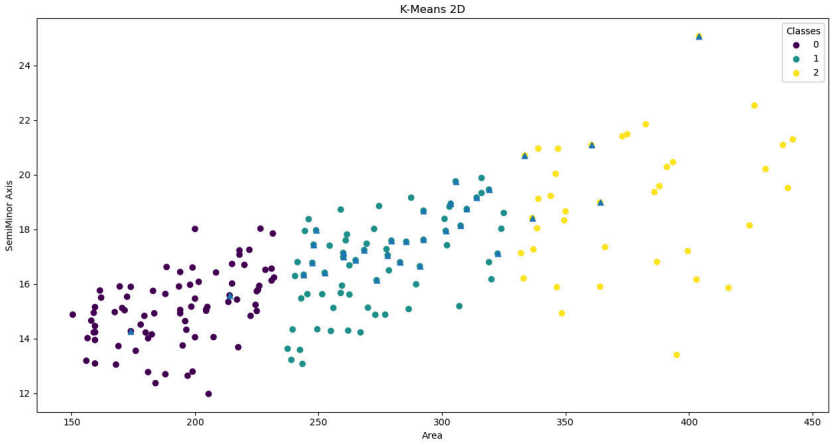

Figura 6.10: KMeans Área - Semieje Menor de la Imagen 1

Podemos ver como en la clase intermedia, la uno, se concentran la mayoría de las varroas, y solo unas pocas son clasificadas en las otras dos clases, por lo que obtenemos los siguiente valores 6.3

Clase	Precisión	Recall	Número de puntos estimados	Número de puntos reales
0	97 %	50 %	80	157
1	35 %	78 %	71	32
2	0 %	0 %	38	0

Tabla 6.3: Resultados de Kmeans con 3 clases

Puesto que la clase cero y dos representan lo mismo, se podrian unir en una sola, a efecto de calculos, quedando la tabla de la siguiente manera: 6.4

Clase	Precisión	Recall	Número de puntos estimados	Número de puntos reales
0	94 %	71 %	118	157
1	35 %	78 %	71	32

Tabla 6.4: Resultados de Kmeans con 3 clases reducido a 2

Podemos ver como la precisión en este caso ha aumentado así como el *recall*, por lo que podemos extraer que un gran número de varroas están bien clasificadas, aunque hay otros puntos irrelevantes que se siguen clasificando en la clase de varroas.

6.1.3. Combinación de datos

Para esta prueba se ha realizado el análisis de varias imágenes en concreto de las siguientes planchas:

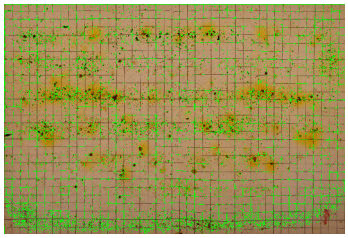

Figura 6.11: Imagen 1 con contornos señalados

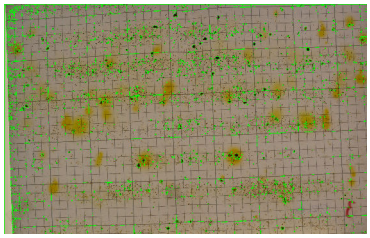

Figura 6.12: Imagen 3 con contornos señalados

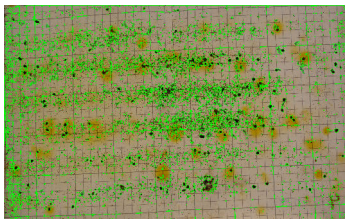

Figura 6.13: Imagen 4 con contornos señalados

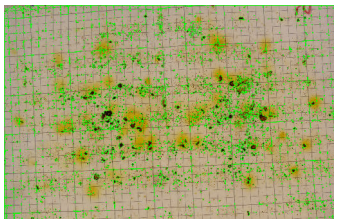

Figura 6.14: Imagen 5 con contornos señalados

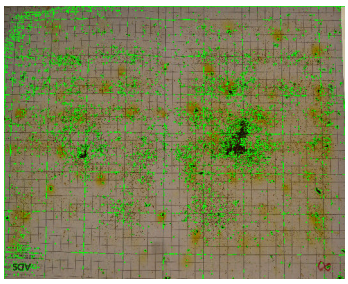

Figura 6.15: Imagen 8 con contornos señalados

Tras extraer los datos individualmente de cada uno de las fotografías, se han estandarizado estos datos, puesto que el tamaño de la imagen puede alterar el área y el valor de los semiejes de la elipse, ya que estos valores se miden en píxeles, además también se ha estandarizado el nivel de color, ya que el brillo y luminosidad entre imágenes cambian, con esto conseguimos que los valores de distintas imágenes sean comparables y pueden ser tratados juntos.

Posteriormente realizamos una representación de las propiedades que se han extraído para entender de una forma mas gráfica donde se localizan las varroas. Los más representativos son el área, el nivel del canal azul y la desviación del canal azul, es decir cuanto varía el nivel de color en el objeto. Las siguientes figuras (6.16, 6.17) ilustran lo comentado:

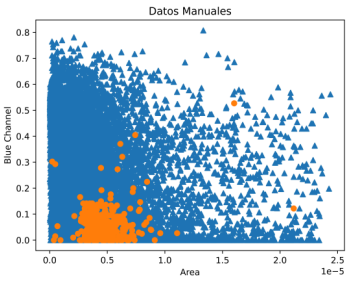

Figura 6.16: Área - Nivel de Canal Azul

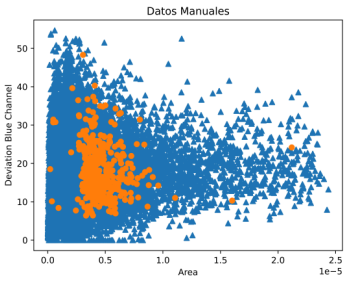

Figura 6.17: Área - Desviación Nivel de Canal Azul

Se pude observar que en ambas gráficas se identifica una región donde se concentran la mayoría de las varroas (representadas con los círculos naranjas). Los puntos mas dispersos pueden ser debidos a errores humanos en el proceso de identificación manual o por la calidad de la imagen que podría distorsionar la forma o el color de los objetos.

Una vez identificado todo, utilizamos el algoritmo KMeans para que agrupe los datos, se han hecho varias pruebas, que se detallarán a continuación:

Utilizando Área y Nivel de Canal Azul

Trás haber analizado los gráficos se comienza a utilizar KMeans para comprobar la eficacia de éste cuando es utilizado con todo el conjunto de datos. Primero se realiza una prueba dividiendo en dos clusters, o clases, y obtenemos los siguientes agrupamientos 6.18:

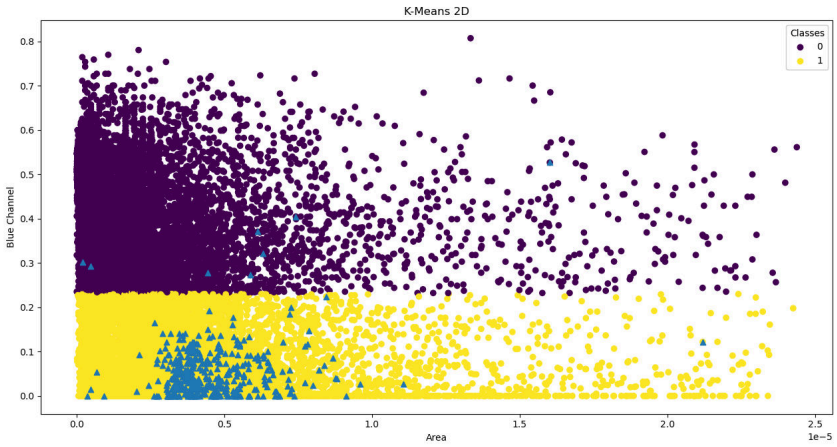

Figura 6.18: KMeans Área - Nivel Canal Azul

Se puede observar que practicamente la totalidad de las varroas son clasificadas bien, pero por el contrario es muy visible que en esa clase, la uno, hay muchos puntos correspondientes a objetos irrelevantes, pero para entenderlo mejor en la siguiente tabla se cuantifica esto 6.5:

Clase	Precisión	Recall	Número de puntos estimados	Número de puntos reales
0	100 %	38 %	7677	20388
1	3 %	98 %	13087	376

Tabla 6.5: Resultados de Kmeans con 2 clases

Como se comentaba en el párrafo anterior, aunque el *recall* de la clase uno es muy alto, obtenemos una muy baja precisión, que además este efecto se observa muy bien comparando el número de puntos estimados respecto a los reales, que es muy superior.

Se ha probado esta configuración pero indicando tres clases, en lugar de dos, pero los resultados obtenidos son prácticamente los mismos a los comentados anteriormente.

Utilizando Área y Desviación de Canal Azul

Otra prueba realizada con este conjunto de datos ha sido, como se ha comentado, utilizando el área y la desviación del canal azul, representado en la gráfica 6.17.

Tras aplicar KMeans se obtiene la siguiente separación de clases:

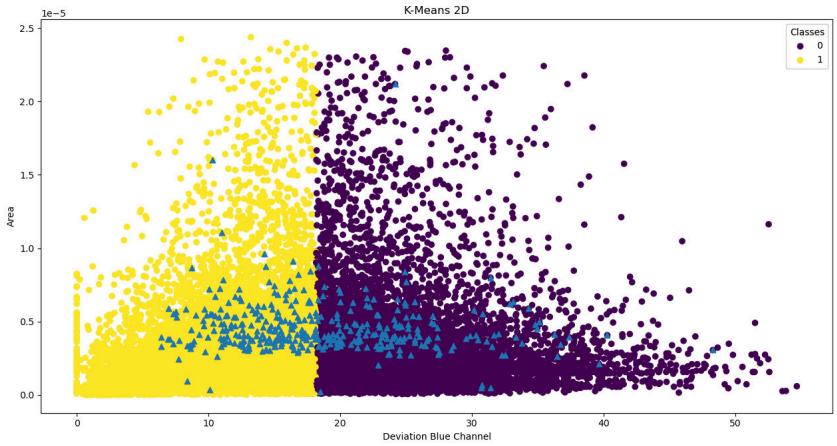

Figura 6.19: KMeans con dos clases Área - Desviación del Canal Azul

Como se puede observar la separación que encuentra no se ajusta a los valores reales, por lo que es de esperar que los valores de precisión y *recall* sean bajos, por lo que esta propuesta queda descartada.

6.1.4. Utilizando todas las propiedades

Por ultimo, se ha realizado una prueba pasando a KMeans todos los datos, es decir todas las propiedades. Con esto se ha obtenido la siguiente clasificación 6.20:

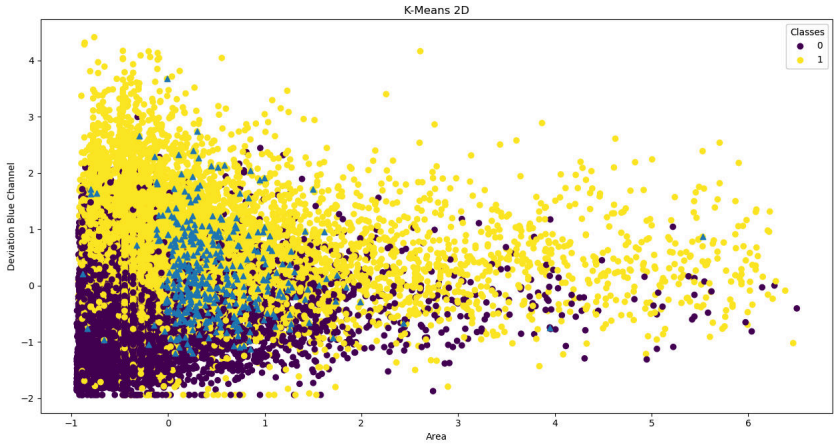

Figura 6.20: KMeans con dos clases Área - Desviación del Canal Azul

Como se puede observar utilizando solo dos agrupamientos se espera obtener una precisión muy baja debido al alto número de puntos irrelevantes en la clase uno, por ello directamente se va a hacer la prueba utilizando tres clases. Esto está representado en el siguiente gráfico 6.21

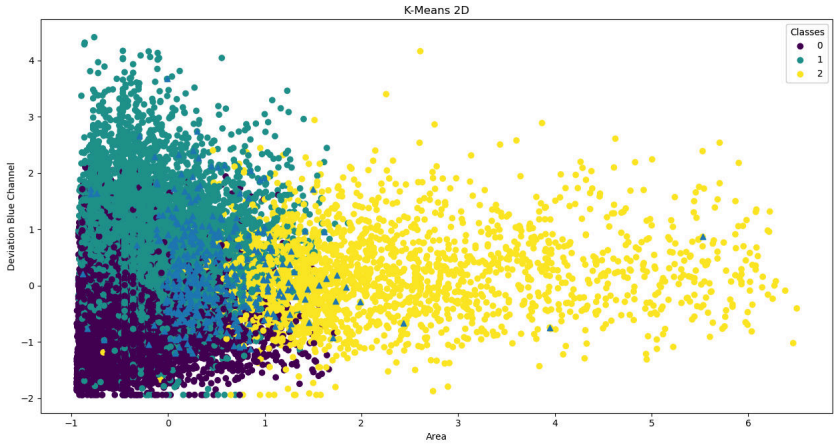

Figura 6.21: KMeans con tres clases Área - Desviación del Canal Azul

Y la correspondiente tabla de resultados 6.6:

Clase	Precisión	Recall	Número de puntos estimados	Número de puntos reales
0	100 %	57 %	11673	20388
1	4 %	85 %	9091	376

Tabla 6.6: Resultados de Kmeans con 3 clases

Como podemos ver en la tabla 6.6 se han representado solo dos clases, cuando KMeans esta clasificando en tres, esto es debido a que hay dos clases que representan lo mismo, los puntos irrelevantes, por lo que se pueden unir en una a efectos de cálculos, para que sea mas sencillo interpretar los datos.

Podemos observar que apenas ha mejorado la precisión respecto a los casos anteriores, y el *recall* ha disminuido. Por lo que esta solución tampoco es del todo satisfactoria.

Comparativa del número de agrupamientos

En este apartado se a va analizar como varía la precisión y el *recall* conforme se va aumentando el número de agrupaciones para KMeans. Para ello se han realizado unos gráficos de como evoluciona esto en valores entre 0 y 1 6.22:

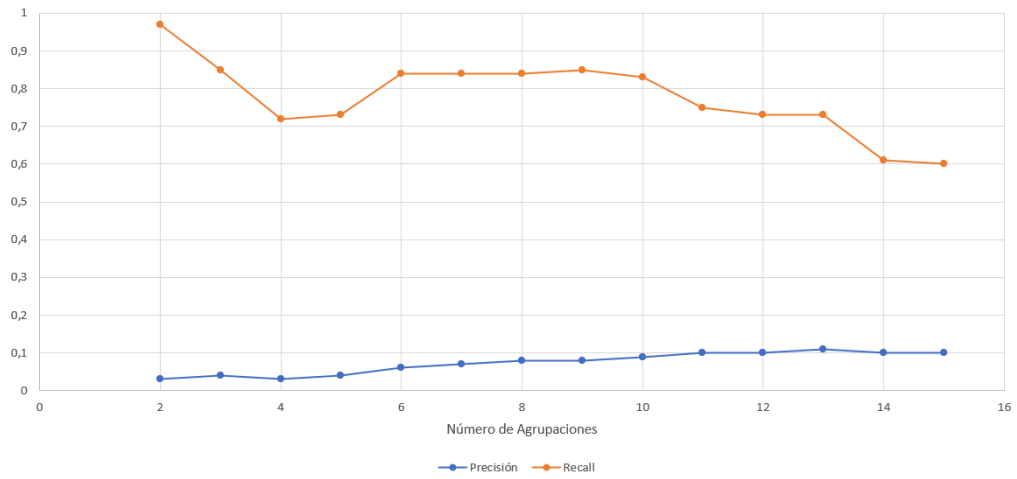

Figura 6.22: Comparativa número de agrupamientos

Como se puede ver, el máximo recall se obtiene con dos agrupamientos pero coincide con el punto de menor precisión. En cambio, a medida que se aumenta el número de agrupaciones, aumenta ligeramente la precisión pero disminuye el *recall*.

6.2 DBScan

Otro algoritmo de clustering, o agrupamiento, que se utilizó fue DBScan, en este caso es el propio algoritmo el que determina el número de clases en las que realizará las agrupaciones.

Para esta prueba se ha utilizado el conjunto de datos de las cinco imágenes comentado anteriormente. Para aplicar DBScan debemos establecer los parámetros de epsilon, que es la distancia entre puntos vecinos para que ambos puedan ser considerados de la misma clase, y el mínimo tamaño de muestra, es decir el mínimo número que debe tener una clase. Para ello primero obtenemos una gráfica de las distancias de los puntos 6.23

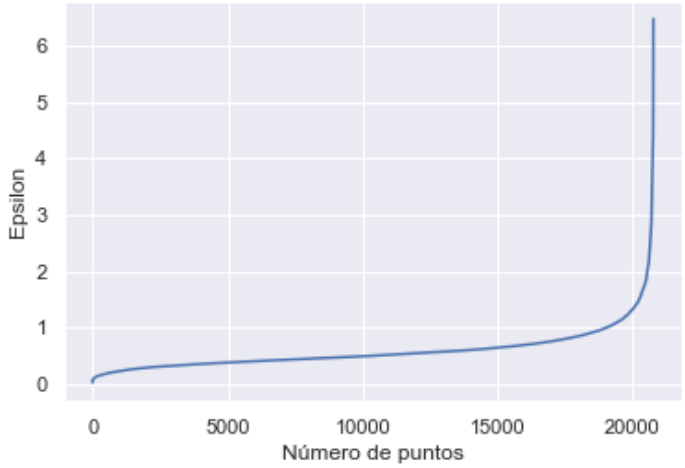

Figura 6.23: Distancia entre vecinos

La epsilon optima será aquella que se encuentra en el valle de la curva, en este caso será 1,2. Este método viene definido por un estudio realizado por "IOP Science"[5].

Una vez establecida la epsilon, ya se puede utilizar el algoritmo DBScan, con el cual obtenemos los siguientes resultados de clasificación:

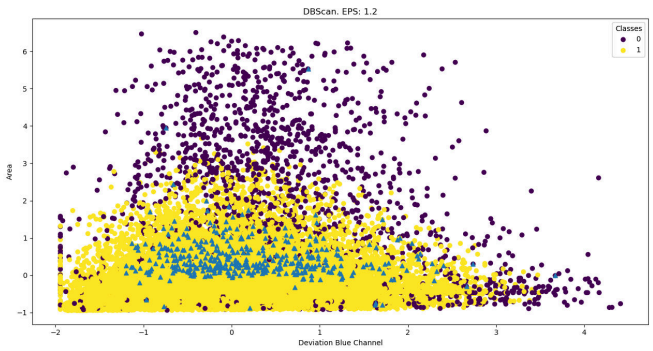

Figura 6.24: DBSCAN Desviación Color Azul - Área

Como vemos la clase uno, que es la de las varroas, abarca otros puntos que deberían pertenecer a la clase cero, esto se observa claramente en la siguiente matriz de confusión 6.25:

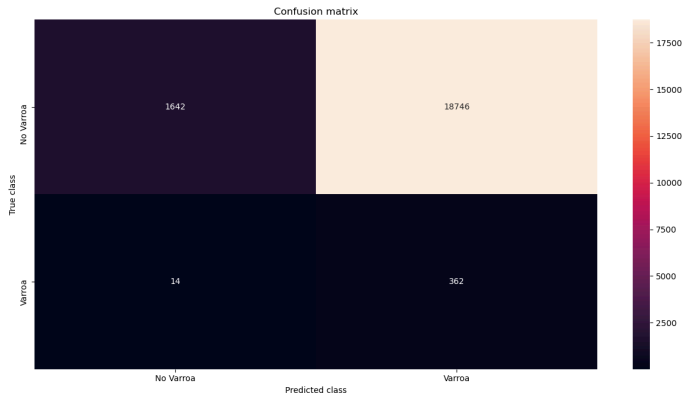

Figura 6.25: Matríz de confusion de DBSCAN con epsilon 1.2

Se observa que la mayoría de varroas están bien etiquetadas, pero tambíen etiqueta de forma erronea muchos otros puntos. Si calculamos la precisión y el *recall* obtenemos la siguiente tabla 6.7:

Clase	Precisión	Recall	Número de puntos estimados	Número de puntos reales
0	99 %	8 %	1642	20388
1	2 %	96 %	18746	376

Tabla 6.7: Resultados de Dbscan con 1.2 epsilon

Vemos que para la clase uno obtenemos un muy buen valor de *recall* pero por el contrario el valor de precisión es demasiado bajo.

6.3 Clasificadores

Otro experimento realizado ha sido utilizando clasificadores, tanto lineal como cuadráticos. Se ha utilizado el mismo conjunto de datos que el detallado en el apartado 6.1.3.

6.3.1. Lineal

Primero se va a analizar el clasificador lineal, que tras aplicarlo al conjunto completo, se obtiene la siguiente separación 6.26:

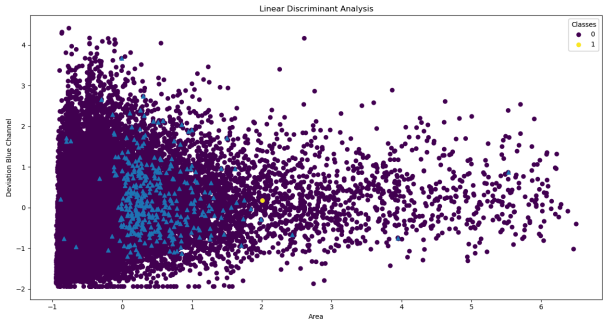

Figura 6.26: Clasificador Lineal Área - Desviación Canal Azul

Como se puede observar no ha clasificado ningún punto, eso es debido a que no ha encontrado ninguna función lineal que se ajuste a la distribución de los datos para separarlos en las clases.

6.3.2. Cuadrático

Puesto que con el anterior no se han obtenido resultados, se va a probar con una clasificador cuadrático, con el cual obtenemos la siguiente clasificación 6.27:

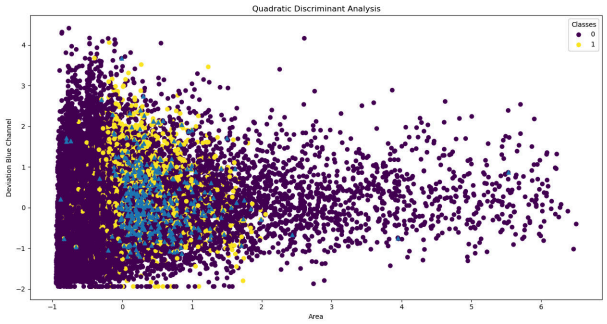

Figura 6.27: Clasificador Cuadrático Área - Desviación Canal Azul

Podemos ver que con este clasificador se obtienen mejores resultados, ya que los puntos de la clase uno se ajustan mejor a la clase de las varroas, representada con los triángulos azules.

Para analizarlo mejor, se muestra a continuación la representación de la matriz de confusión 6.28:

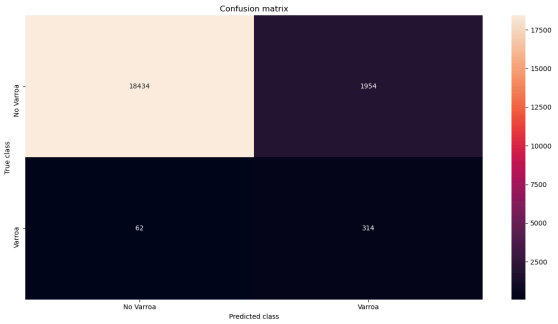

Figura 6.28: Matríz de Confusión del Clasificador Cuadrático

Como vemos se siguen clasificando muchos puntos irrelevantes en la clase de las varroas, pero muchos menos que con los métodos anteriores.

Si cuantificamos esto en porcentajes obtenemos la siguiente tabla 6.8

Clase	Precisión	Recall	Número de puntos estimados	Número de puntos reales
0	100 %	90 %	18496	20388
1	14 %	84 %	2268	376

Tabla 6.8: Resultados del clasificador cuadrático

Estos resultados nos acercan más al objetivo, aunque la precisión sigue siendo baja, lo cual penaliza al sistema. Por lo que hay que mejorarlo para conseguir un valor de precisión aceptable.

6.4 Affinity Propagation

Por último, se ha realizado un experimento con el algoritmo "Affinity Propagation", se ha realizado con dos conjuntos de datos, el primero es el utilizado en el apartado 6.1.2 y el segundo se utiliza el conjunto completo, el detallado en 6.1.3

6.4.1. Primer experimento

En este experimento como se ha comentado, se utiliza el conjunto de datos de la imagen de la plancha uno, al cual se le ha hecho un filtrado. Para ello se han utilizado las propiedades de área y valor del semieje menor de la elipse 6.7. Con ello se obtiene la siguiente separación 6.29:

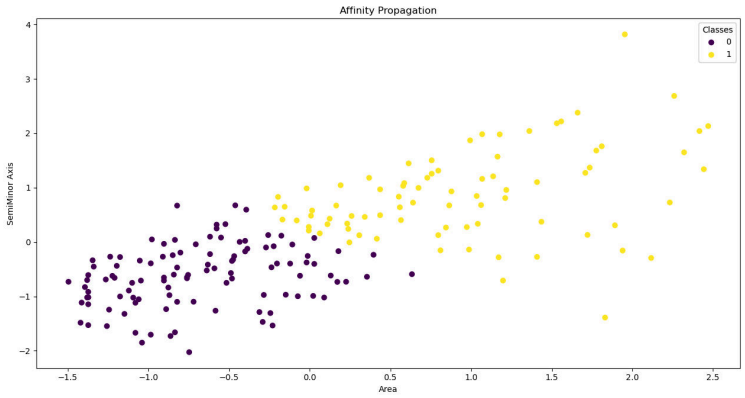

Figura 6.29: Affinity Propagation Área - Semieje Menor

Si analizamos la matriz de confusión obtenemos los siguientess valores 6.30:

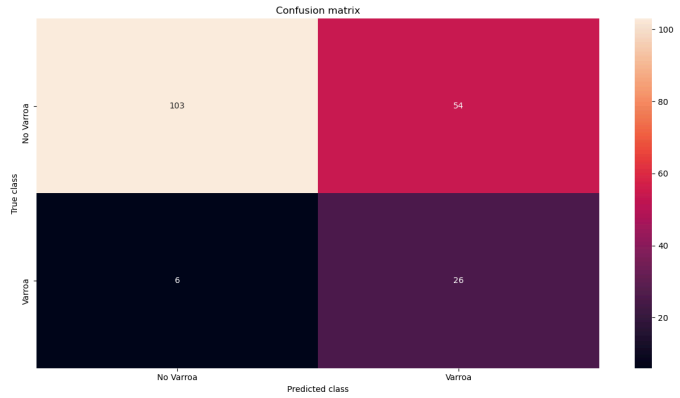

Figura 6.30: Matriz de Confusión Affinity Propagation Área - Semieje Menor

Vemos que la mayoría de los puntos pertenecientes a varroas están bien clasificados, aunque por otra parte también esta clasificando en esta clase a varios de los puntos que pertenecen a la clase de irrelevantes. Para tener una visión mas específica se presenta la siguiente tabla 6.9:

Clase	Precisión	Recall	Número de puntos estimados	Número de puntos reales
0	94 %	66 %	106	157
1	33 %	81 %	80	32

Tabla 6.9: Resultados del clasificador cuadrático

Observamos que tenemos un buen valor de *recall* y la precisión aun siendo baja es superior a muchas de las pruebas realizadas anteriormente.

6.4.2. Segundo experimento

Para este experimento como se comentaba anteriormente se va a utilizar el conjunto completo de datos, pero puesto que son más de 20000 datos, este algoritmo es muy costoso en tiempo, ya que tiene una función de tiempo de $\Theta(N^2T)$ [6] donde "N" es el número de muestras y "T" es el número de iteraciones antes de que converga. Por lo que teniendo en cuenta que el anterior experimento convergió en la iteración 53, se puede estimar que el coste de tiempo de éste, suponiendo que converga en la misma iteriación, sería de 21.200.000.000*t, por lo que se desestimó este experimento.

6.5 Resumen

A modo de resumen final, se adjunta una tabla donde se indica la precisión y el recall que se ha obtenido para cada experimento en la clase 1, es decir, en la clase varroa.

Experimento	Precisión	Recall
6.1.1	8 %	94 %
6.1.2	28 %	56 %
6.1.2	35 %	78 %
6.1.3	3 %	98 %
6.1.4	4 %	85 %
6.2	2 %	96 %
6.3.2	14 %	84 %
6.4.1	33 %	81 %

Tabla 6.10: Resumen resultados de los experimentos

CAPÍTULO 7
Conclusiones

Para desarrollar este trabajo se ha necesitado un aprendizaje continuo tanto en el ámbito de imágenes digitales, desde como está está formada una imagen hasta como detectar objetos en ella; como en el de análisis de datos, por ejemplo, el procesamiento de datos utilizando técnicas de clustering, usando clasificadores... Además ha sido fundamental el aprendizaje de un nuevo lenguaje de programación, como es python, asi como de la herramienta Jupyter, ya que todos estos conocimientos aprendidos han sido fundamentales para alcanzar los objetivos planteados inicialmente en este proyecto.

Se ha desarrollado un sistema de reconocimiento de formas, que se encarga de procesar una imagen y clasificar los objetos detectados en ésta. En este sistema además se incluye un web para que la interacción con el usuario sea lo más fácil posible, entre otras características.

Sin embargo, debido a retrasos en la obtención de imágenes con la suficiente calidad y los resultados de los conteos manuales, se ha dispuesto de menos tiempo para realizar pruebas y ajustes con las imágenes de mayor definición y realizar comparativas respecto a valores reales, ya que se ha tenido que desarrollar gran parte del proyecto con imágenes de baja calidad y sin la información de su conteo, por lo que no se sabía claramente si el sistema que se estaba desarrollando tenía suficiente precisión.

Por último, el sistema de identificación de las varroas no cumple del todo los requisitos que se esperaban ya que como se comenta en el capítulo de Evaluación 6, la precisión es demasiado baja. Esto se puede solucionar con mayor dedicación en tiempo para así poder mejorar la fase de procesado para obtener datos mas ajustados, con lo que ayudaría en la fase de clasificación. Ya que actualmente hay contornos que no se ajustan fielmente a la varroa en la imagen por lo que genera datos erróneos.

7.1 Relación con los estudios cursados

El desarrollo de este proyecto ha sido posible gracias a los conocimientos adquiridos en la asignatura de Bioinformática (BIO). Aunque la tecnología utilizada no ha sido la vista en clase, los conceptos obtenidos en esta materia han ayuda-

do bastante a comprender tanto los métodos de segmentación de datos como la interpretación posterior de los resultados.

Por otro lado, para el desarrollo de la página web se han usado conocimientos que se adquieren en la asignatura de Desarrollo Web (DEW), como el desarrollo HTML y JavaScript.

CAPÍTULO 8
Trabajos futuros

En este capítulo se va a comentar las líneas de desarrollo que se pretenden seguir tras la realización de este trabajo.

Actualmente, la precisión que se obtiene oscila entre el 30-40 % , la cual para el sistema que queremos es demasiado baja, por ello se pretende aumentarla hasta un 80-90 %. Para ello se van a mejorar el proceso de binarización de la imagen, mediante la aplicación de otros algoritmos que eliminen de forma mas exhaustiva el ruido que tenga la imagen, para así tener datos de los puntos de las varroas que se ajusten mejor, puesto que ahora tenemos contornos que no se están ajustando bien a los puntos, lo que nos genera datos erróneos. Además, se pretende conseguir mas imágenes, y por consiguiente mas datos, para así poder implementar redes neuronales en el proceso de clasificación, utilizando la librería TensorFlow.

Posteriormente, cuando se haya alcanzado la primera linea, se pretende ofrecer al público apicultor a través de la web, la cual, junto con el servidor, se alojarán en un sistema en la nube, e implantar un sistema de pago por uso.

Bibliografía

[1] Varroosis según el Ministerio de Agricultura, Pesca Y Alimentación. Consultado en https://www.mapa.gob.es/es/ganaderia/temas/sanidad-animal-higiene-ganadera/sanidad-animal/enfermedades/varroosis/Varroosis.aspx

[2] Aman Naimat. *The New Artificial Intelligence Market* O'Reilly Media,Inc., 2016

[3] Sector Apícola según el Ministerio de Agricultura, Pesca Y Alimentación. Consultado en https://www.mapa.gob.es/es/ganaderia/temas/produccion-y-mercados-ganaderos/sectores-ganaderos/apicola/default.aspx

[4] Entrevista a Simone Strey, CEO de Plantix, el día 14 de Junio de 2018. Consultado en https://www.code-brew.com/blog/2018/06/14/the-smart-crop-assistant-plantix/

[5] Nadia Rahmah and Imas Sukaesih Sitanggang *Determination of Optimal Epsilon (Eps) Value on DBSCAN Algorithm to Clustering Data on Peatland Hotspots in Sumatra* Consultado en https://iopscience.iop.org/article/10.1088/1755-1315/31/1/012012

[6] Descripción del algoritmo de Affinity Propagation de Scikit Learn Consultado en https://scikit-learn.org/stable/modules/clustering.html#affinity-propagation

[7] MÉTODOS DE UMBRALIZACIÓN DE IMÁGENES DIGITALES BASADOS EN ENTROPIA DE SHANNON Y OTROS Carlos A. Cattaneoa, Ledda I. Larchera, Ana I. Ruggerib, Andrea C. Herreraa, Enrique M. Biasonia Consultado en https://cimec.org.ar/ojs/index.php/mc/article/view/3951

[8] Encontrar la intensidad del gradiente de la imagen Consultado en https://es.wikipedia.org/wiki/Algoritmo_de_Canny

[9] Juan Humberto Sossa Azuela. *Visión Artificial*
Rasgos Descriptores para el Reconocimiento de Objetos. Ra-Ma Editorial., Madrid, primera edición, 2013.

[10] Joe Minichino. Joseph Howse. *Learning OpenCV 3 Computer Vision with Python*. GB: Packt Publishing, second edition, 2015.

APÉNDICE A
Imágenes utilizadas

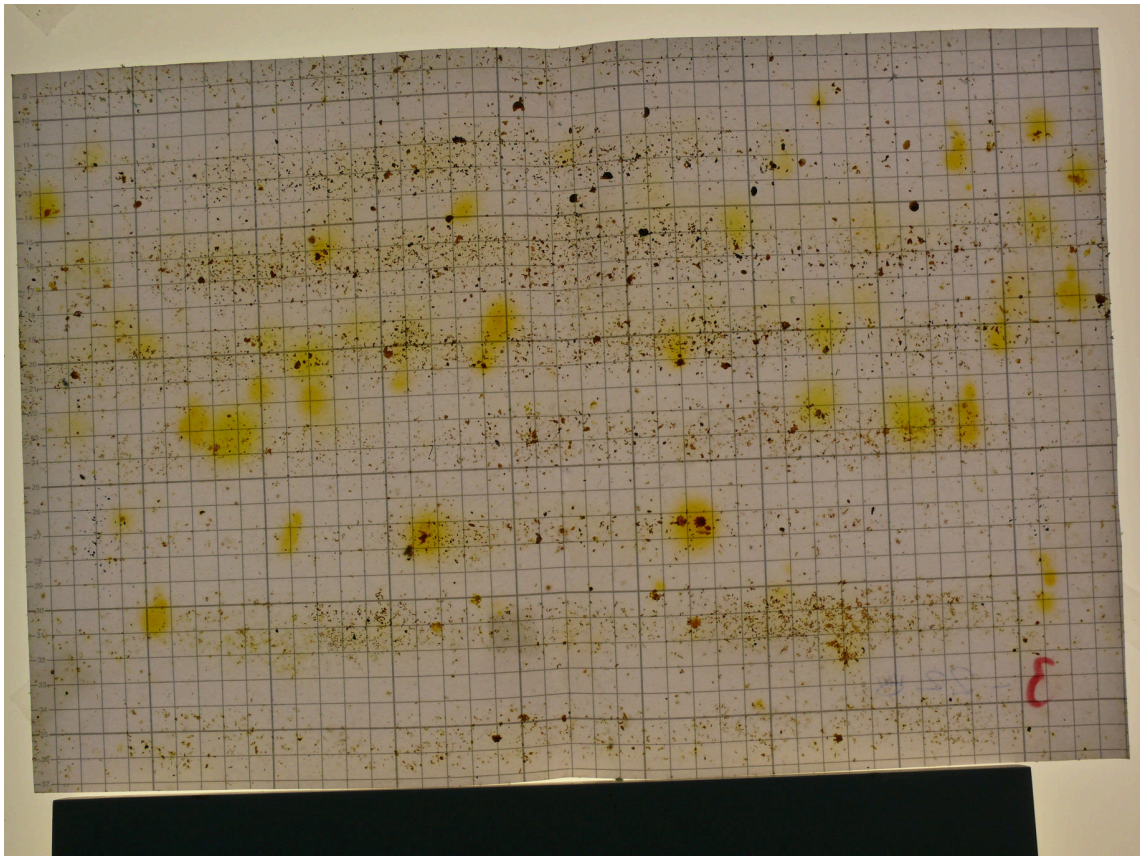

Figura A.1: Plancha tres

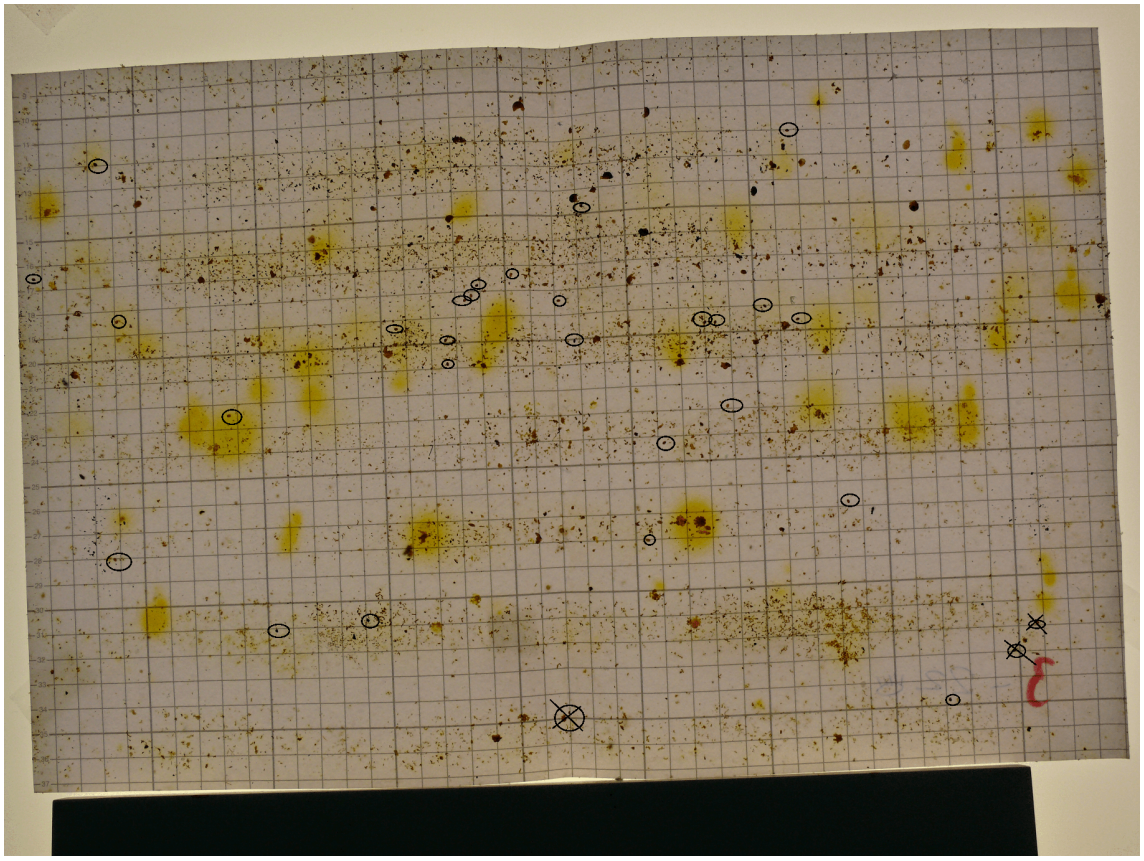

Figura A.2: Plancha tres con las varroas señaladas

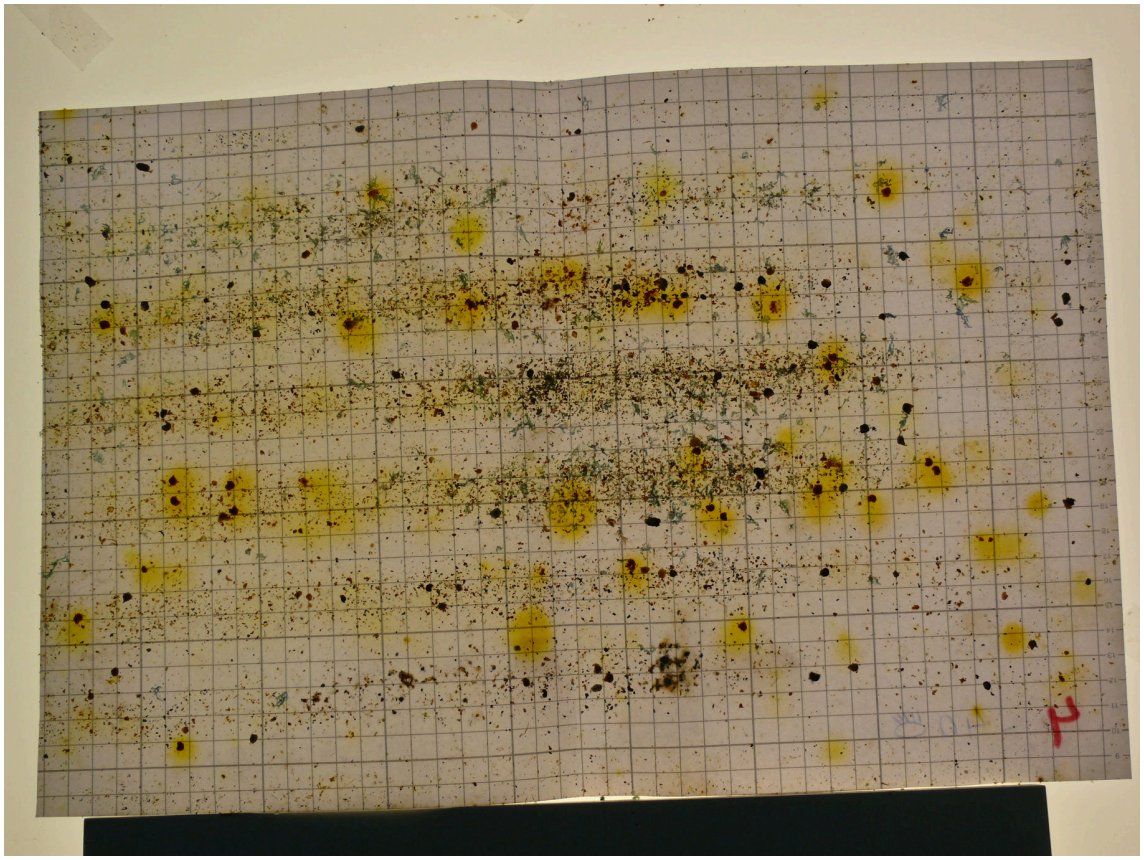

Figura A.3: Plancha cuatro

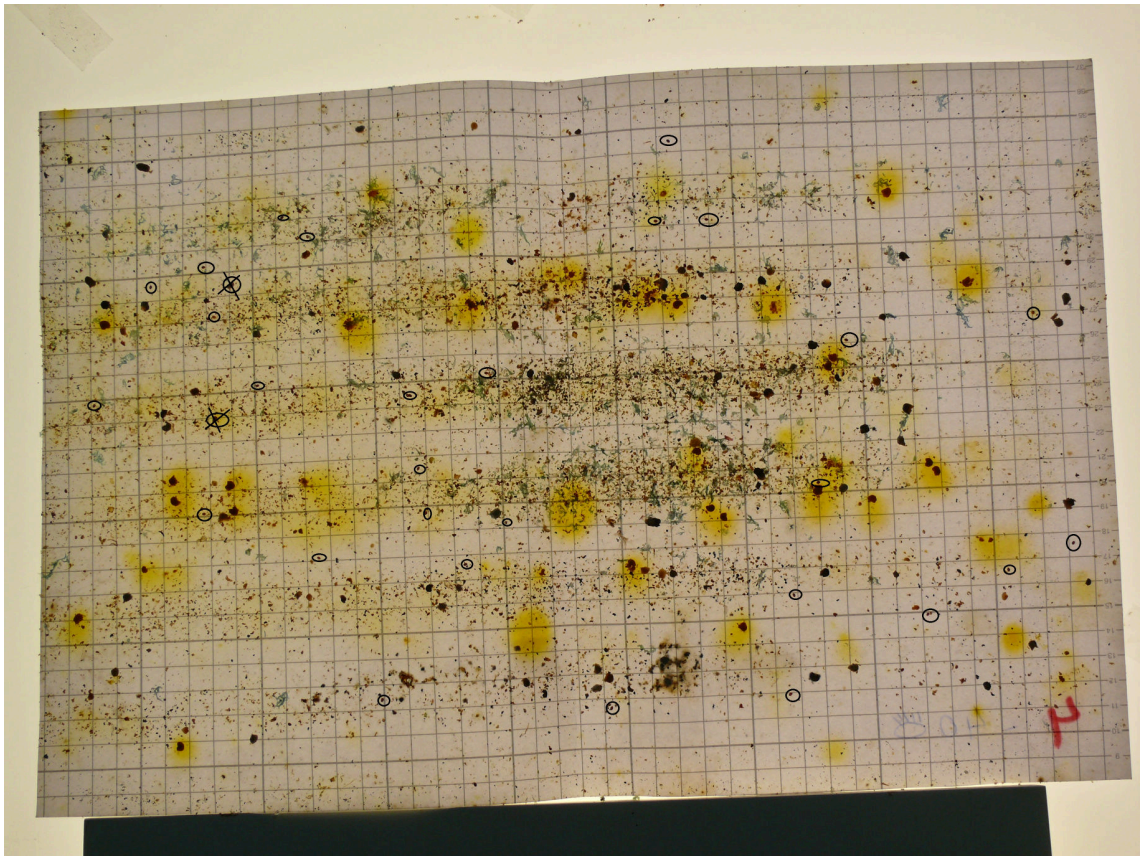

Figura A.4: Plancha cuatro con las varroas señaladas

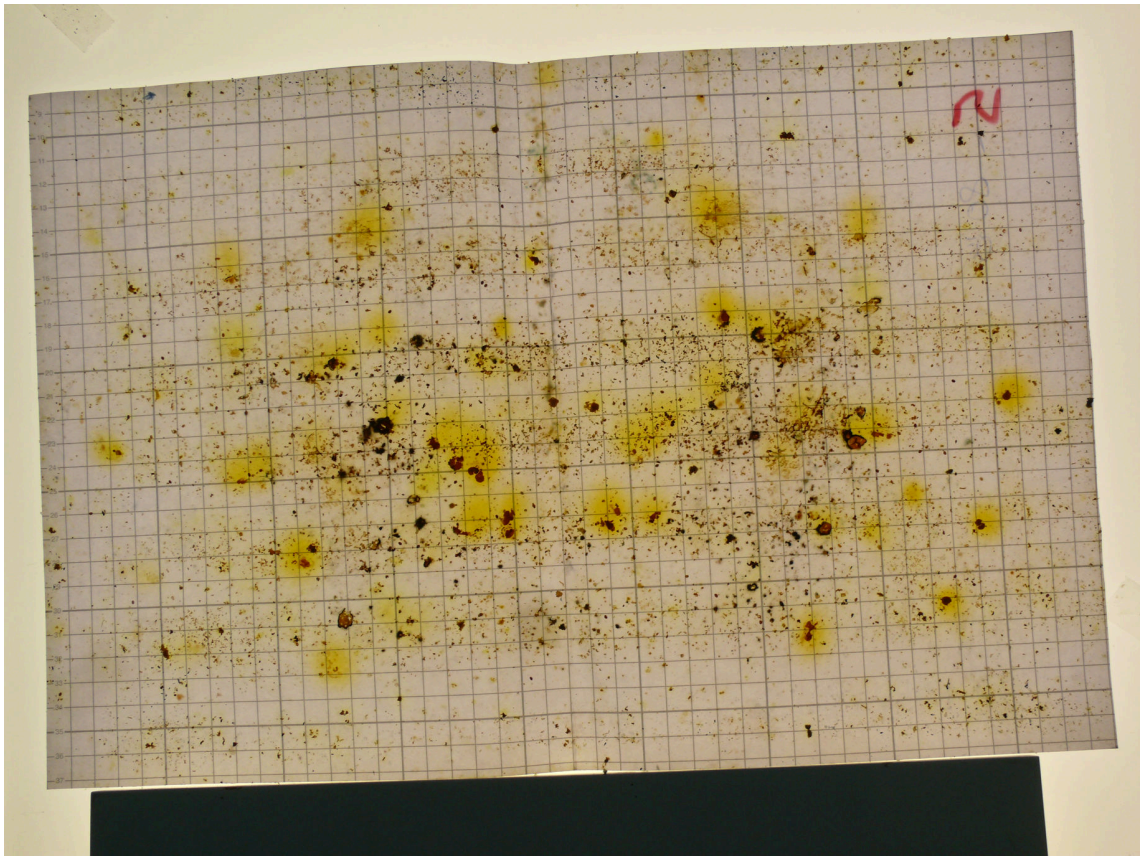

Figura A.5: Plancha cinco

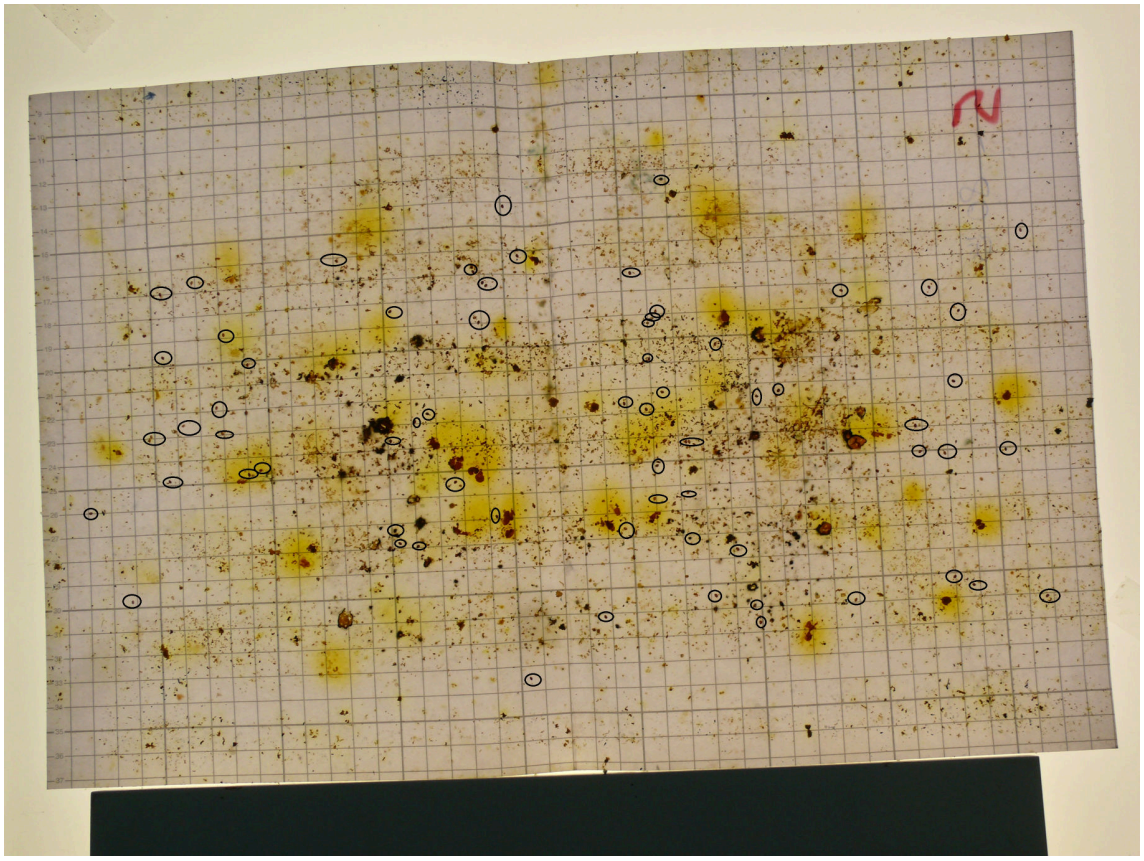

Figura A.6: Plancha cinco con las varroas señaladas

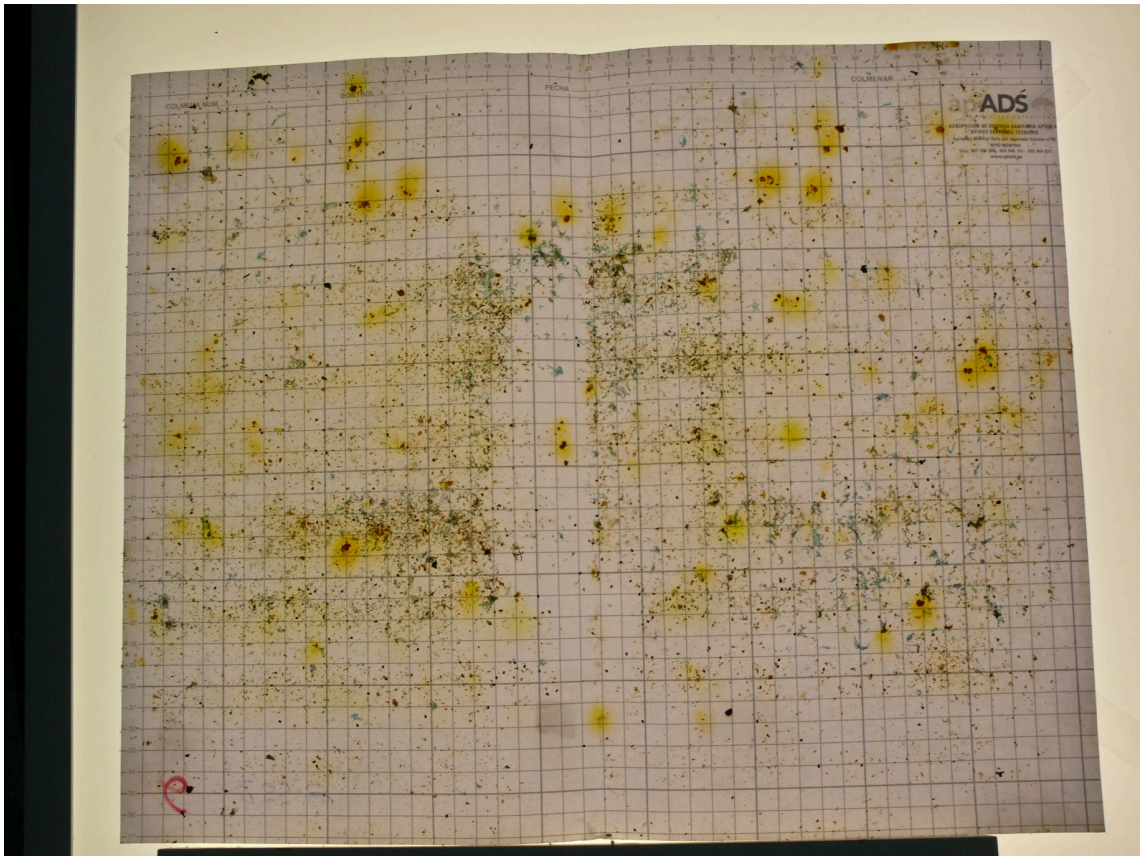

Figura A.7: Plancha seis

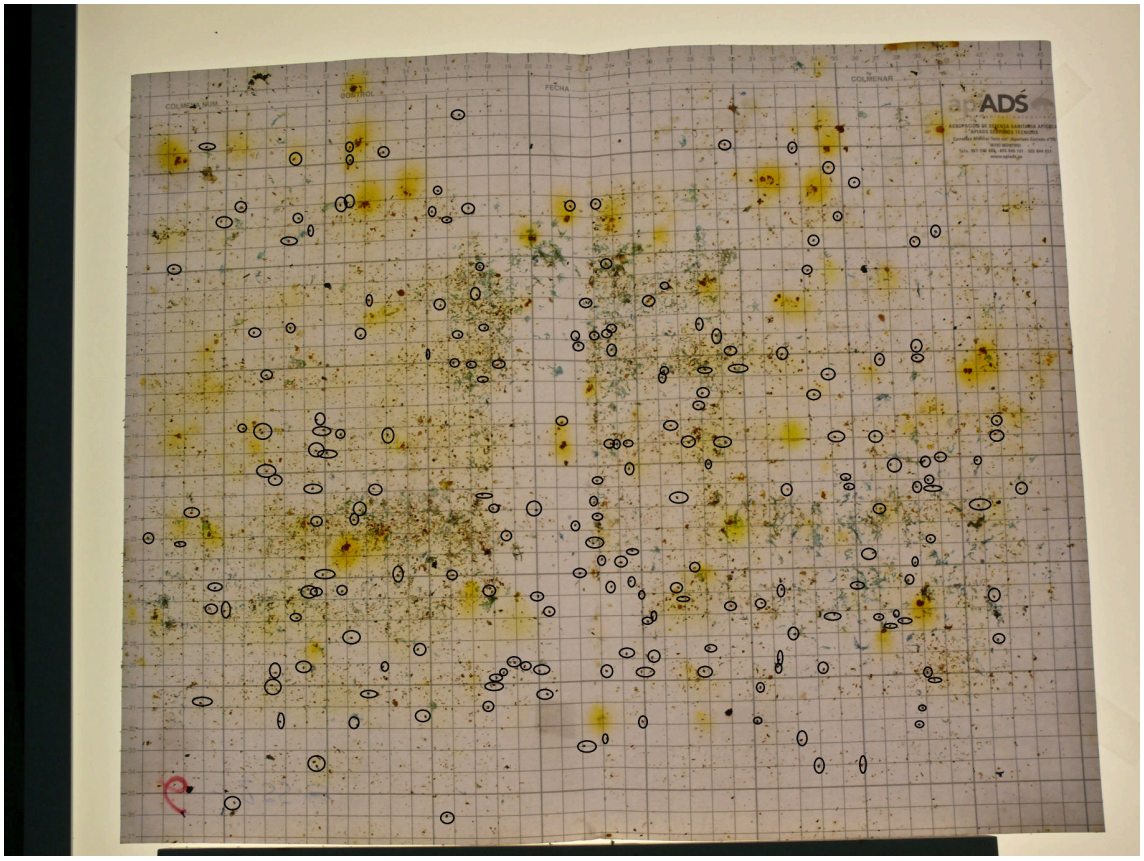

Figura A.8: Plancha seis con las varroas señaladas

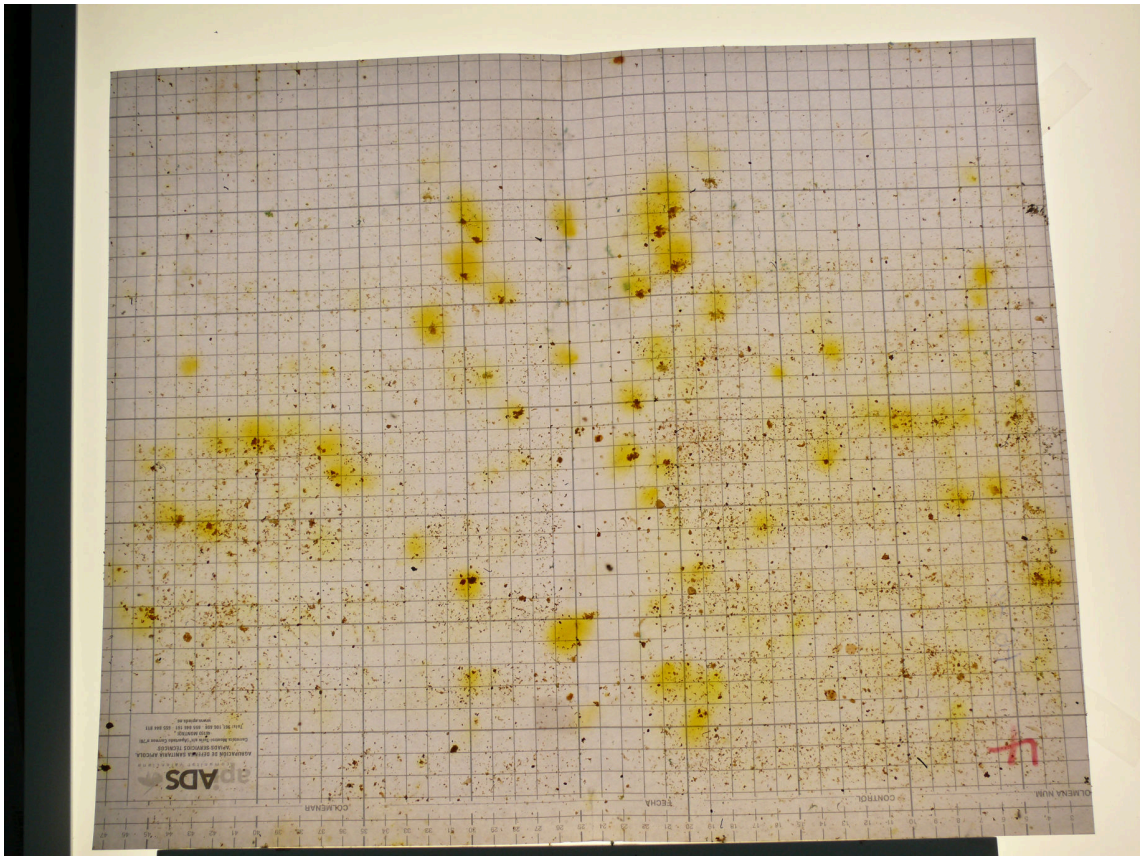

Figura A.9: Plancha siete

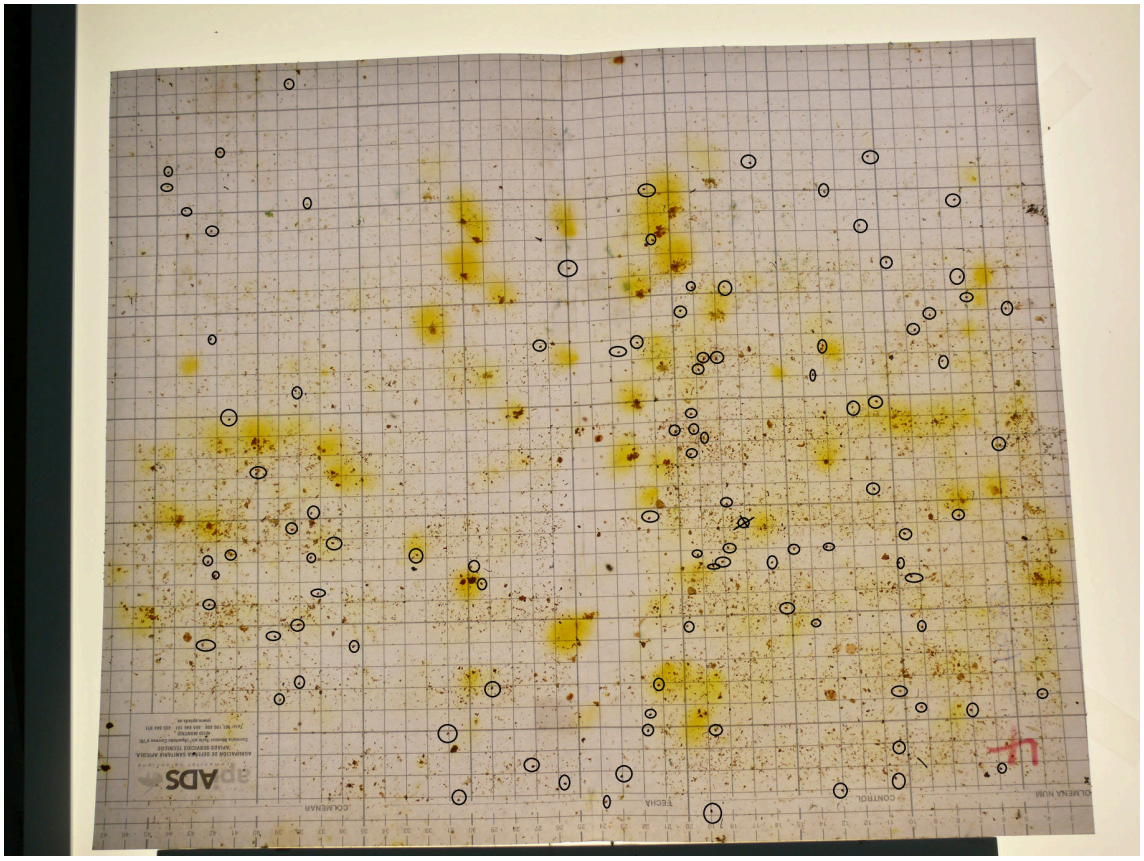

Figura A.10: Plancha siete con las varroas señaladas

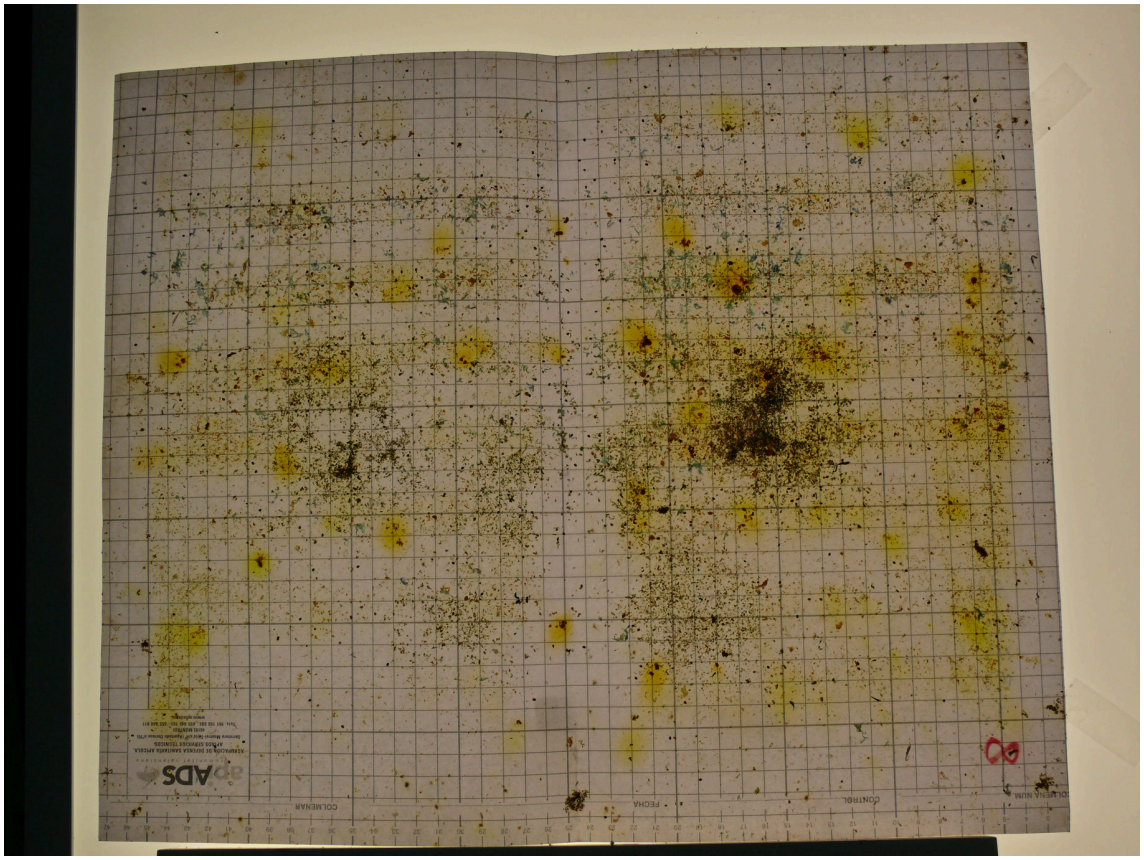

Figura A.11: Plancha ocho

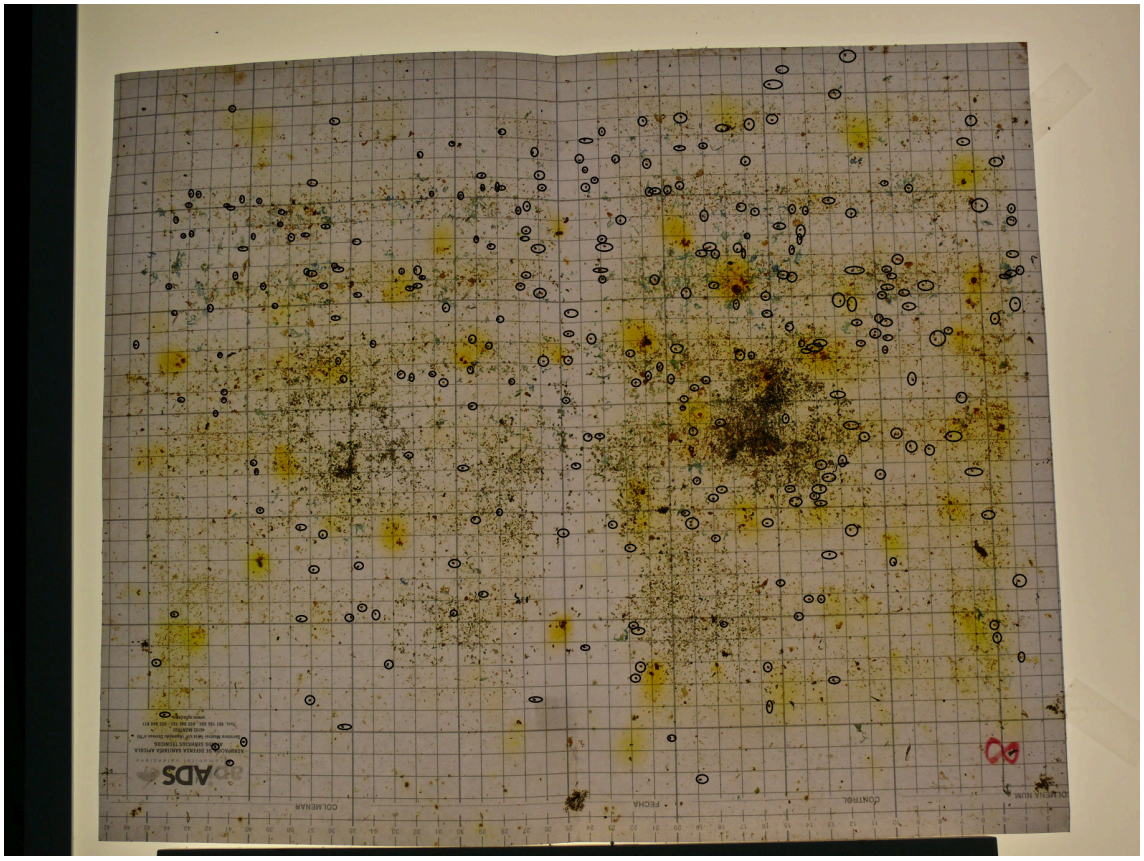

Figura A.12: lancha ocho con las varroas señaladas