UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

TFG del Grado en Ingeniería Informática

GoBees

Monitorización del estado de una colmena mediante la cámara de un smartphone.

Presentado por David Miguel Lozano
en Universidad de Burgos — 26 de enero de 2017
Tutores: José Francisco Díez Pastor
y Raúl Marticorena Sánchez

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

D. nombre tutor, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. David Miguel Lozano, con DNI dni, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 26 de enero de 2017

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. nombre tutor	D. nombre co-tutor

Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

Descriptores

Palabras separadas por comas que identifiquen el contenido del proyecto Ej: servidor web, buscador de vuelos, android ...

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

keywords separated by commas.

Índice general

Índice general III

Índice de figuras IV

Introducción 1
1.1. Estructura de la memoria 2
1.2. Materiales adjuntos 3

Objetivos del proyecto 4
2.1. Objetivos generales 4
2.2. Objetivos técnicos 4
2.3. Objetivos personales 5

Conceptos teóricos 6
3.1. Preprocesado 6
3.2. Substracción del fondo 8
3.3. Posprocesado 13
3.4. Detección y conteo de abejas 14

Técnicas y herramientas 17

Aspectos relevantes del desarrollo del proyecto 18

Trabajos relacionados 19

Conclusiones y Líneas de trabajo futuras 20

Bibliografía 21

Índice de figuras

3.1. Resultado de la fase de preprocesado. 7
3.2. Plataforma de desarrollo del algoritmo. 12
3.3. Resultado de la fase de substracción del fondo. 13
3.4. Resultado de la fase de posprocesado. 14
3.5. Resultado de la fase de detección y conteo de abejas. 16
3.6. Resultado del algoritmo en una imágen con moscas. 16

Introducción

Las abejas son una pieza clave en nuestro ecosistema. La producción de alimentos a nivel mundial y la biodiversidad de nuestro planeta dependen en gran medida de ellas. Son las encargadas de polinizar el 70 % de los cultivos de comida, que suponen un 90 % de la alimentación humana [9]. Sin embargo, la población mundial de abejas está disminuyendo a pasos agigantados en los últimos años debido, entre otras causas, al uso extendido de plaguicidas tóxicos, parásitos como la varroa o la expansión del avispón asiático [10].

Actualmente los apicultores inspeccionan sus colmenares de forma manual. Uno de los indicadores que más información les proporciona es la actividad de vuelo de la colmena (número de abejas en vuelo a la entrada de la colmena en un determinado instante) [2]. Este dato, junto con información previa de la colmena y conocimiento de las condiciones locales, permite conocer al apicultor el estado de la colmena con bastante seguridad, pudiendo determinar si esta necesita o no una intervención.

La actividad de vuelo de una colmena varía dependiendo de múltiples factores, tanto externos como internos a la colmena. Entre ellos se encuentran la propia población de la colmena, las condiciones meteorológicas, la presencia de enfermedades, parásitos o depredadores, la exposición a tóxicos, la presencia de fuentes de néctar, etc. A pesar de los numerosos factores que influyen en la actividad de vuelo, su conocimiento es de gran ayuda para la toma de decisiones por parte del apicultor. Ya que este posee información sobre la mayoría de los factores necesarios para su interpretación.

La captación prolongada de la actividad de vuelo de forma manual es muy costosa, tediosa y puede introducir una tasa de error elevada. Es por esto que a lo largo de los años se haya intentado automatizar este proceso de muy diversas maneras. Los primeros intentos se remontan a principios del siglo XX, donde se desarrollaron contadores por contacto eléctrico [17]. Otros métodos posteriores se basan en sensores de infrarrojos [26], sensores capacitivos [3], códigos de barras [6] o incluso en microchips acoplados a las abejas [7]. En

los últimos años, se han desarrollado numerosos métodos basados en visión artificial [2, 4, 5, 27].

Los métodos basados en contacto, sensores de infrarrojos o capacitivos tienen el inconveniente de que es necesario realizar modificaciones en la colmena, mientras que en los basados en códigos de barras o microchips es necesario manipular las abejas directamente. Estos motivos los convierten en métodos poco prácticos fuera del campo investigador. Por el contrario, la visión artificial aporta un gran potencial, ya que evita tener que realizar ningún tipo de modificación ni en el entorno, ni en las abejas. Además, abre la puerta a la monitorización de nuevos parámetros como la detección de enjambrazón (división de la colmena y salida de un enjambre) o la detección de amenazas (avispones, abejaruco, etc.).

Todos los métodos basados en visión artificial propuestos hasta la fecha utilizan hardware específico con un coste elevado. En este trabajo se propone un método de monitorización de la actividad de vuelo de una colmena mediante la cámara de un *smartphone* con Android.

El método propuesto podría revolucionar el campo de la monitorización de la actividad de vuelo de colmenas, ya que lo hace accesible al gran público. Ya no es necesario contar con costoso hardware, difícil de instalar. Solamente es necesario disponer de un *smartphone* con cámara y la aplicación GoBees. Además, esta facilita la interpretación de los datos, representándolos gráficamente y añadiendo información adicional como la situación meteorológica. Permitiendo a los apicultores centrar su atención donde realmente es necesaria.

1.1. Estructura de la memoria

La memoria sigue la siguiente estructura:

- **Introducción:** breve descripción del problema a resolver y la solución propuesta. Estructura de la memoria y listado de materiales adjuntos.
- **Objetivos del proyecto:** exposición de los objetivos que persigue el proyecto.
- **Conceptos teóricos:** breve explicación de los conceptos teóricos clave para la comprensión de la solución propuesta.
- **Técnicas y herramientas:** listado de técnicas metodológicas y herramientas utilizadas para gestión y desarrollo del proyecto.
- **Aspectos relevantes del desarrollo:** exposición de aspectos destacables que tuvieron lugar durante la realización del proyecto.
- **Trabajos relacionados:** estado del arte en el campo de la monitorización de la actividad de vuelo de colmenas y proyectos relacionados.

- **Conclusiones y líneas de trabajo futuras:** conclusiones obtenidas tras la realización del proyecto y posibilidades de mejora o expansión de la solución aportada.

Junto a la memoria se proporcionan los siguientes anexos:

- **Plan del proyecto software:** planificación temporal y estudio de viabilidad del proyecto.
- **Especificación de requisitos del software:** se describe la fase de análisis; los objetivos generales, el catálogo de requisitos del sistema y la especificación de requisitos funcionales y no funcionales.
- **Especificación de diseño:** se describe la fase de diseño; el ámbito del software, el diseño de datos, el diseño procedimental y el diseño arquitectónico.
- **Manual del programador:** recoge los aspectos más relevantes relacionados con el código fuente (estructura, compilación, instalación, ejecución, pruebas, etc.).
- **Manual de usuario:** guía de usuario para el correcto manejo de la aplicación.

1.2. Materiales adjuntos

Los materiales que se adjuntan con la memoria son:

- Aplicación para Android GoBees.
- Aplicación Java para el desarrollo del algoritmo.
- Aplicación Java para el etiquetado de fotogramas.
- JavaDoc.
- *Dataset* de vídeos de prueba.

Además, los siguientes recursos están accesibles a través de internet:

- Página web del proyecto [16].
- GoBees en Google Play [13].
- Repositorio del proyecto [15].
- Repositorio de las herramientas desarrolladas para el proyecto [14].

Objetivos del proyecto

A continuación, se detallan los diferentes objetivos que han motivado la realización del proyecto.

2.1. Objetivos generales

- Desarrollar una aplicación para *smartphone* que permita la monitorización de la actividad de vuelo de una colmena a través de su cámara.
- Facilitar la interpretación de los datos recogidos mediante representaciones gráficas.
- Aportar información extra a los datos de actividad que ayude en la toma de decisiones.
- Almacenar todos los datos generados de forma estructurada y fácilmente accesible.

2.2. Objetivos técnicos

- Desarrollar un algoritmo de visión artificial con OpenCV que permita contar el número de abejas en cada fotograma en tiempo real.
- Desarrollar una aplicación Android con soporte para API 19 y superiores.
- Aplicar la arquitectura MVP (*Model-View-Presenter*) en el desarrollo de la aplicación.
- Utilizar Gradle como herramienta para automatizar el proceso de construcción de software.
- Utilizar Git como sistema de control de versiones distribuido junto con la plataforma GitHub.
- Hacer uso de herramientas de integración continua como Travis, Codecov, Code Climate, SonarQube o VersionEye en el repositorio.

- Aplicar la metodología ágil Scrum junto con TDD (*Test Driven Development*) en el desarrollo del software.
- Realizar test unitarios, de integración y de interfaz.
- Utilizar ZenHub como herramienta de gestión de proyectos.
- Utilizar un sistema de documentación continua como Read the Docs.
- Distribuir la aplicación resultante en la plataforma Google Play.
- Realizar una página web para la difusión de la aplicación.

2.3. Objetivos personales

- Realizar una aportación a la modernización de la apicultura.
- Abarcar el máximo número de conocimientos adquiridos durante la carrera.
- Explorar metodologías y herramientas novedosas utilizadas en el mercado laboral.
- Adentrarme en el campo de la visión artificial.
- Profundizar en el desarrollo de aplicaciones Android.

Conceptos teóricos

La parte del proyecto con mayor complejidad teórica radica en el algoritmo de visión artificial, el cual se puede dividir en cuatro fases. En primer lugar, se realiza un preprocesado de la señal de entrada para optimizarla. En segundo lugar, se substrae el fondo para segmentar los objetos en movimiento. Posteriormente, se realiza un posprocesado de la señal para mejorar los resultados obtenidos de la substracción del fondo. Y por último, se clasifican y cuentan los contornos que pueden pertenecer a una abeja.

A continuación se exponen los conceptos teóricos que conlleva cada fase.

3.1. Preprocesado

Antes de aplicar el algoritmo de substracción del fondo, es recomendable realizar un preprocesado de los fotogramas para facilitar el procesado posterior, minimizar el ruido y optimizar los resultados. A continuación se explican las técnicas utilizadas.

Conversión de RGB a escala de grises

Los fotogramas captados por la cámara se devuelven en formato RGB. En este formato se poseen tres matrices, una por cada canal (*Red-Green-Blue*). La suma aditiva de los tres canales resulta en una imagen a color.

Sin embargo, el color no proporciona ninguna información relevante en nuestra tarea de identificación de abejas. Es por esto que se pueden convertir los fotogramas de RGB a escala de grises. De esta manera, se trabajará solamente con una matriz de píxeles en lugar de tres. Simplificando, en gran medida, el número de operaciones a realizar y por tanto, aumentado el rendimiento de nuestro algoritmo final.

OpenCV utiliza la conversión colométrica a escala de grises [22]. Esta técnica se basa en principios colométricos para ajustar la luminancia de la imagen a

color y la imagen resultante. Devolviendo una imagen con la misma luminancia absoluta y la misma percepción de luminosidad [30].

Utiliza la siguiente fórmula para calcular la luminancia resultante:

$$\text{RGB[A] to Gray:} \quad Y \leftarrow 0{,}299 \cdot R + 0{,}587 \cdot G + 0{,}114 \cdot B$$

La fórmula calcula la luminancia de una forma no lineal, sin necesidad de realizar una expansión gamma. Los coeficientes intentan imitar la percepción de intensidad medida por un humano tricromático, siendo más sensible al color verde y menos al color azul [30].

Desenfoque Gaussiano

Las imágenes captadas pueden contener ruido que puede dificultar su procesamiento. El ruido son variaciones aleatorias del brillo o del color de una imagen. Una técnica que permite reducirlo es el desenfoque.

En nuestro caso hemos utilizado desenfoque Gaussiano, un filtro de paso bajo que reduce las componentes de alta frecuencia de la imagen utilizando para ello una convolución con una función Gaussiana [29]. Se diferencia del desenfoque promedio en que da más peso a los vecinos cercanos, siendo estos más influyentes en el resultado.

El kernel utilizado en la convolución es una muestra discreta de la función Gaussiana. En nuestro caso utilizamos un kernel 3x3 que se corresponde con: [11]

$$M = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

En la siguiente imagen se puede ver el resultado de aplicar esta fase a la imagen de entrada:

Figura 3.1: Resultado de la fase de preprocesado.

3.2. Substracción del fondo

En un sistema de monitorización por vídeo resulta de gran interés el poder extraer los objetos en movimiento del resto de la imagen. Esto se conoce como extracción del fondo, en inglés *background subtraction* o *foreground detection*, y consiste en clasificar todos los píxeles de un determinado fotograma bien como fondo, o como primer plano [28]. En primer plano se engloban todos los objetos en movimiento, mientras que en el fondo se encuentran todos los objetos estáticos junto con posibles sombras, cambios de iluminación u otros objetos en movimiento que no son de interés, como puede ser la rama de un árbol balanceándose por el viento. Se trata de un paso crítico, ya que algoritmos posteriores dependen en gran medida de los resultados de este.

En nuestro proyecto, la toma de imágenes se realiza mediante una cámara estática. Esto facilita en parte la detección del fondo, ya que este se corresponderá con todos los píxeles estáticos. Sin embargo, como trabajamos en un entorno al aire libre, tenemos que lidiar también con cambios de iluminación, sombras, u otros objetos móviles que no son de nuestro interés (falsos positivos).

Con OpenCV para Android podemos implementar varios algoritmos básicos de extracción del fondo y además, nos proporciona la implementación de dos algoritmos más sofisticados: `BackgroundSubtractorMOG` y `BackgroundSubtractorKNN`. Tras realizar un estudio de todos ellos, nos decantamos finalmente por `BackgroundSubtractorMOG2`. A continuación, explicamos el funcionamiento de todos los algoritmos que se probaron, así como los resultados que nos proporcionaron.

Substracción con imagen de referencia

Se parte de una imagen de referencia del fondo, en la que no haya ningún objeto en movimiento. A partir de esta, se obtienen los elementos en movimiento substrayendo a cada fotograma la imagen tomada como referencia.

Este método, al tomar un modelo del fondo tan sencillo y estático, es muy vulnerable a cambios en la escena (iluminación, sombras, objetos del fondo con ligeros movimientos, pequeñas oscilaciones de la cámara, etc.). Sin embargo, ofrece muy buenos resultados cuando se trabaja en una escena con la iluminación y los elementos controlados, ya que al ser tan simple, es muy eficiente [8].

Para implementar este algoritmo con OpenCV, se hace uso de la función `Core.absdiff()`.

En nuestro problema, al trabajar al aire libre nos es imposible utilizar este algoritmo. Ya que el modelo del fondo cambia constantemente.

Substracción del fotograma anterior

En este método, el modelo del fondo se extrae del fotograma anterior. De tal manera, que a cada nuevo fotograma se le substrae el anterior.

De esta manera se mejora la respuesta a cambios en la escena, como los cambios de iluminación. Sin embargo, si un objeto en movimiento se queda estático en la imagen, este deja de ser detectado [1].

La implementación se realiza como en la técnica anterior, variando el modelo del fondo.

Tras probarlo en nuestro problema específico, vimos que no nos era de utilidad. Ya que el fotograma resultante de la diferencia contenía las abejas por duplicado (la abeja en el fotograma actual y misma abeja en el fotograma anterior en otra posición).

Substracción del acumulado de los fotogramas anteriores

Una mejora interesante del algoritmo anterior supone tomar como modelo del fondo un acumulado de los fotogramas anteriores de acuerdo a un ratio de aprendizaje. De esta forma, se puede lidiar con cambios en el fondo de la imagen dinámicamente. El modelo se calcula de acuerdo a la siguiente fórmula:

$$u_t = (1 - \alpha)u_{t-1} + \alpha\ p_t$$

Donde p_t es el nuevo valor del píxel, u_{t-1} es la media del fondo en el instante $t - 1$, u_t es la nueva media del fondo y α es el ratio de aprendizaje (cómo de rápido olvida los frames anteriores) [1].

OpenCV provee la función `Imgproc.accumulateWeighted()` que implementa la fórmula anterior por nosotros. Haciendo uso de esta función y de la utilizada en la sección anterior podemos implementar este algoritmo.

Tras probarlo, vimos que tenía una eficiencia muy buena y se adaptaba a los cambios correctamente. Sin embargo, no era capaz de diferenciar las sombras de las abejas, por lo que se obtenían falsos positivos.

BackgroundSubtractorKNN

Se trata de un método que se basa en el algoritmo de clasificación supervisada *K Nearest Neighbors* (k-nn). El algoritmo fue propuesto en el artículo [33]. Y de acuerdo con sus conclusiones, es muy eficiente cuando el número de píxeles que se corresponden con el primer plano es bajo.

La clase de OpenCV que lo implementa es `BackgroundSubtractorKNN`. Los parámetros más importantes son:

- `history`: número de fotogramas recientes que afectan al modelo del fondo.
- `dist2Threshold`: umbral de la distancia al cuadrado entre el píxel y la muestra para decidir si un píxel está cerca de esa muestra.
- `detectShadows`: con un valor verdadero detecta las sombras (aumenta considerablemente el tiempo de procesado).

En nuestras pruebas, el algoritmo proporcionaba unos resultados buenos pero su tiempo de ejecución era muy elevado (entorno a 25ms/frame). Como el tiempo de ejecución es un factor clave en nuestro proyecto, se descartó el uso de este algoritmo.

BackgroundSubtractorMOG2

`BackgroundSubtractorMOG2` es una mejora del algoritmo `BackgroundSubtractorMOG`. En la versión original de OpenCV se encuentran implementados ambos, sin embargo, en los *wrappers* para Android solo disponemos de la revisión.

`BackgroundSubtractorMOG` está basado en el modelo *Gaussian Mixture* (GMM). Se trata de un modelo compuesto por la suma de varias distribuciones Gaussianas que, correctamente elegidas, permiten modelar cualquier distribución [18].

El algoritmo de substracción del fondo fue propuesto en el artículo [31] y modela cada píxel del fondo como la mezcla de K distribuciones Gaussianas. Los pesos de la mezcla representan las proporciones de tiempo que el color de ese píxel se ha mantenido en la escena. Siendo los colores de fondo más probables los que más permanezcan y sean más estáticos [21].

`BackgroundSubtractorMOG2` se basa en los mismos principios que su antecesor pero implementa una mejora sustancial. Es el propio algoritmo el que selecciona el número adecuado de distribuciones Gaussianas necesarias para modelar cada píxel. De esta manera, se mejora notablemente la adaptabilidad del algoritmo a variaciones en la escena. Fue propuesto en los artículos [32] y [33].

El código fuente de este algoritmo está disponible en [19] (interfaz) y [20] (implementación).

La clase de OpenCV que lo implementa es `BackgroundSubtractorMOG2`. Posee los siguientes parámetros configurables: [25]

- `history`: número de fotogramas recientes que afectan al modelo del fondo. Se representa en la literatura como `T`. Por defecto, 500 fotogramas. Nosotros hemos obtenido buenos resultados con valores de entorno a 50.

- `learningRate`: valor entre 0 y 1 que indica como de rápido aprende el modelo. Si se establece un valor de -1 el algoritmo elige automáticamente el ratio. 0 significa que el modelo del fondo no se actualiza para nada, mientras que 1 supone que el modelo del fondo se reinicializa completamente cada nuevo fotograma. En la literatura podemos encontrar este parámetro como `alfa`. Si el intervalo que se quiere considerar es `history`, se debe establecer `alfa=1/history` (valor por defecto). También se pueden mejorar los resultados iniciales estableciendo `alfa=1` en el instante 0 e ir decrementándolo hasta `alfa=1/history`. De esta manera, en el inicio aprende rápidamente, pero una vez estabilizada la situación las variaciones afectan menos al modelo. En nuestro caso, el valor por defecto ha funcionado correctamente.
- `backgroundRatio`: si un pixel del primer plano permanece con un valor semi-constante durante `backgroundRatio*history` fotogramas, es considerado fondo y se añade al modelo del fondo como centro de una nueva componente Gaussiana. En los artículos se hace referencia a este parámetro como `TB`. `TB=0.9` es el valor por defecto. Este parámetro nos permite decidir cuando dejar de contar una abeja que se ha quedado inmóvil o un objeto nuevo en la escena como podría ser una hoja que se acaba de caer de un árbol.
- `detectShadows`: con un valor verdadero (valor por defecto) detecta las sombras (aumenta ligeramente el tiempo de procesado). Nos permite despreciar las sombras de las abejas con muy buenos resultados.
- `shadowThreshold`: el algoritmo detecta las sombras comprobando si un píxel es una versión oscurecida del fondo. Este parámetro define cómo de oscura puede ser la sombra como máximo. Por ejemplo, un valor de 0.5 (valor por defecto) significa que si un píxel es más del doble de oscuro, entonces no se considerará sombra. En los artículos se representa como `Tau`.
- `shadowValue`: es el valor utilizado para marcar los píxeles de sombras en la máscara resultante. El valor por defecto es 127. En la máscara devuelta, un valor de 0 siempre se corresponde con un pixel del fondo, mientras que un valor de 255 con un píxel del primer plano.
- `nMixtures`: número máximo de componentes Gaussianas para modelar el modelo del fondo. El número actual se determina dinámicamente para cada píxel. Hemos utilizado el valor por defecto, 5.
- `varThreshold`: umbral utilizado en el cálculo de la distancia cuadrada de Mahalanobis entre el píxel y el modelo del fondo para decidir si una muestra está bien descrita por el modelo o no. Este parámetro no afecta a la actualización del modelo del fondo. Se representa como `Cthr`. Por defecto, 16. Se han obtenido mejores resultados con valores de entorno a 40.
- `varThresholdGen`: umbral sobre la distancia cuadrada de Mahalanobis entre el píxel y el modelo para ayudar a decidir si un píxel está cercano

a alguna de las componentes del modelo. Si no es así, es considerado como primer plano o añadido como centro de una nueva componente (dependiendo del `backgroundRatio`). Se representa como `Tg` y su valor por defecto es 9. Un valor menor genera más componentes Gaussianas, mientras que un valor mayor genera menos.

- `complexityReductionThreshold`: este parámetro define el número de muestras necesarias para probar que una componente existe. Se representa como `CT`. Su valor por defecto es `CT=0.05`. Si se establece su valor a 0 se obtiene un algoritmo similar al de Stauffer & Grimson (no se reduce el número de componentes).
- `varInit`: varianza inicial de cada componente Gaussiana. Afecta a la velocidad de adaptación. Se debe ajustar teniendo en cuenta la desviación estandar de las imágenes. Por defecto es 15.
- `varMin`: varianza mínima. Por defecto, 4.
- `varMax`: varianza máxima. Por defecto, `5*varInit`.

De todos ellos, los parámetros más importantes a ajustar son `history` o `learningRate`, `varThreshold` y `detectShadows`.

La parametrización correcta de este algoritmo es clave para su buen funcionamiento. Por ello, durante las pruebas se integró en nuestra aplicación de desarrollo, permitiendo variar todos estos parámetros en tiempo real. De esta manera. se pudo elegir la mejor configuración para nuestro problema concreto.

En la siguiente imagen se puede ver una captura de nuestra plataforma de desarrollo en la pestaña correspondiente a esta fase:

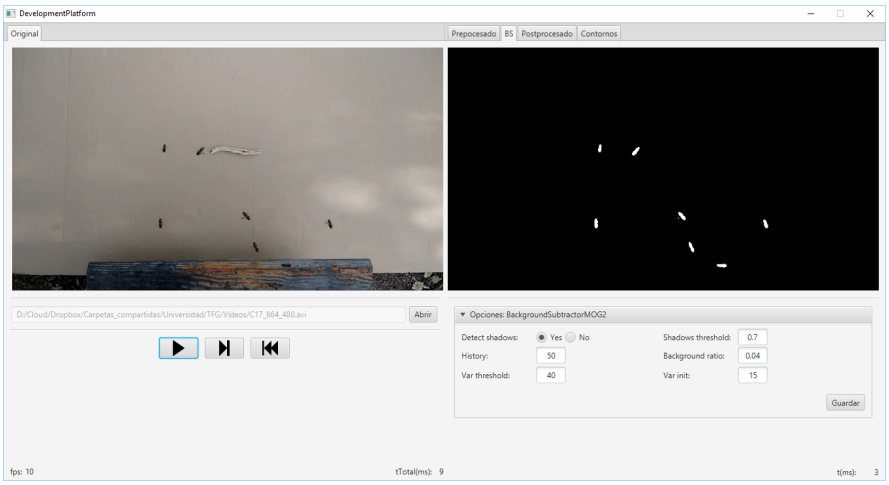

Figura 3.2: Plataforma de desarrollo del algoritmo.

Una vez parametrizado correctamente, vimos como este algoritmo era el que mejores resultados nos proporcionaba. Con un tiempo de ejecución en

nuestro equipo de pruebas de entorno a 4ms/frame, mucho menor que el proporcionado por `BackgroundSubtractorKNN`, de entorno a 25ms/frame. El algoritmo detectaba correctamente las abejas, era resistente al ruido, y además, era capaz de diferenciar una abeja de su sombra. Por todos estos motivos, se seleccionó para la fase de substracción del fondo.

Otros algoritmos

La implementación original de OpenCV implementa otros dos algoritmos más que no están disponibles a través de los *wrappers* de Android.

- `BackgroundSubtractorGMG` es un algoritmo que combina una estimación estadística del fondo de la imagen junto con una segmentación Bayesiana píxel a píxel [21].
- `BackgroundSubtractorFGD` está disponible en la versión para CUDA. Utiliza la regla de decisión de Bayes para clasificar los elementos del fondo y los del primer plano atendiendo a sus vectores de características [12].

En la siguiente imagen se puede ver el resultado de aplicar `BackgroundSubtractorMOG2` a la salida de la fase anterior:

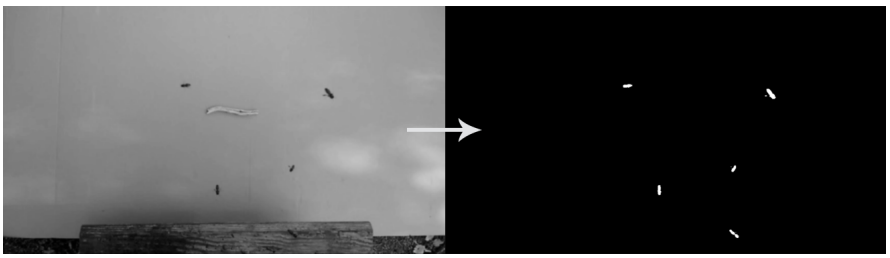

Figura 3.3: Resultado de la fase de substracción del fondo.

Se puede apreciar como ha descartado correctamente las sombras en movimiento de los árboles y se ha quedado únicamente con los objetos en movimiento.

3.3. Posprocesado

Para mejorar los resultados de la extracción de fondo y preparar la imagen para la búsqueda de contornos, se han aplicado las siguientes técnicas:

Dilatación

Se trata de una operación morfológica por la cual se expanden las regiones luminosas de una imagen. Esto se consigue mediante la sustitución de cada píxel por el más brillante de los vecinos considerados por el *kernel* (matriz utilizada para la convolución). De esta manera se consiguen unir las regiones de abejas que podían haberse roto [11].

Erosión

Se trata de la operación contraria a la anterior, expande las regiones oscuras de la imagen. Para ello se coge el valor mínimo de los valores considerados por el *kernel* [11].

La dilatación nos permite reconstruir las abejas, pero también aumenta su tamaño, aumentando el riesgo de solapamientos. Para evitar esto, se vuelve a reducir el tamaño de estas mediante una erosión.

En nuestro algoritmo aplicamos tres operaciones morfológicas seguidas:

1. **Erosión (3x3)**: elimina las piernas de las abejas.
2. **Dilatación (2x2)**: junta la cabeza de las abejas con su cuerpo que en numerosas ocasiones es separado durante la substracción de fondo.
3. **Erosión (3x3)**: recupera el tamaño inicial.

A continuación podemos ver el resultado de esta fase:

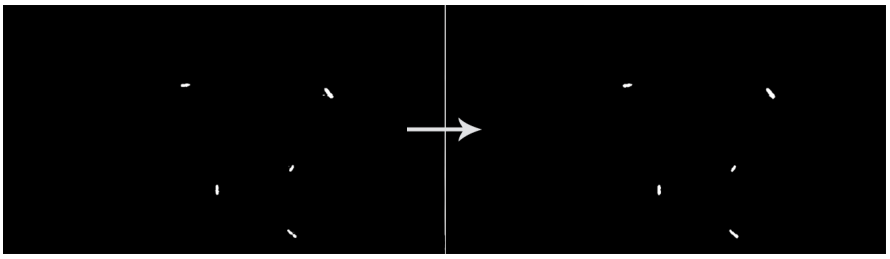

Figura 3.4: Resultado de la fase de posprocesado.

3.4. Detección y conteo de abejas

El último paso que realiza nuestro algoritmo de visión artificial es detectar cuáles de las regiones obtenidas en la fase anterior se correponden con abejas. Para ello, se realiza una búsqueda de contornos y se filtran por área.

Entendemos por contorno una línea curva que une todos los puntos continuos del borde de una región de un mismo color o intensidad.

La salida de la fase anterior es una imagen binaria con los objetos en movimiento en blanco y el fondo en negro. Por lo tanto, el objetivo de esta fase es detectar todas las regiones blancas que puedan corresponderse con una abeja.

OpenCV provee la función `Imgproc.findContours()` para realizar la búsqueda de contornos. Esta toma una imagen binaria y devuelve una lista con todos los contornos encontrados. Para entender la función se necesita comprender una serie de conceptos: [23]

- **Jerarquía**: los contornos pueden ser independientes unos de otros, o poseer una relación padre-hijo cuando un contorno está dentro de otro. En la jerarquía se especifican las relaciones entre contornos.
- **Modo de obtención del contorno**: define cómo se van a obtener los contornos en cuestión de jerarquía [24].
  - `RETR_LIST`: devuelve todos los contornos en una lista, sin ninguna información de jerarquía entre ellos.
  - `RETR_EXTERNAL`: devuelve todos los contornos externos. Si algún contorno tiene contornos hijo, estos son ignorados.
  - `RETR_CCOMP`: devuelve los contornos agrupados en dos niveles de jerarquía. Un primer nivel en el que se encuentran todos los contornos exteriores. Y un segundo nivel con los contornos correspondientes a agujeros en los primeros.
  - `RETR_TREE`: devuelve todos los contornos creando un árbol completo con la jerarquía.
- **Método de aproximación de los contornos**: define el método que utiliza la función para almacenar los contornos [24].
  - `CHAIN_APPROX_NONE`: almacena todos los puntos del borde del contorno.
  - `CHAIN_APPROX_SIMPLE`: almacena sólo los puntos relevantes del contorno. Por ejemplo, si el contorno es una línea no se necesita almacenar todos los puntos de esta, con el punto inicial y el final basta. Esto es lo que realiza este método, eliminar todos los puntos redundantes y comprimirlos para que ocupe menos espacio.
  - `CV_CHAIN_APPROX_TC89_L1` y `CV_CHAIN_APPROX_TC89_KCOS`: aplican el algoritmo de aproximación de cadena de Teh-Chin, simplificando los polígonos que forman los contornos.
  - `CV_CHAIN_CODE`: almacena los contornos utilizando el código de cadenas de Freeman.

En nuestro caso, la configuración más adecuada es utilizar `RETR_EXTERNAL` y `CHAIN_APPROX_SIMPLE`. Ya que no nos interesa ningún contorno interno que

pueda tener la abeja (y que en principio no debería tener) y tampoco nos es relevante el cómo se almacenan estos, sólo nos interesa el número.

Para evitar posibles falsos positivos, establecemos un umbral mínimo y máximo en el área del contorno. De esta manera, evitamos que contornos diminutos o grandes generados por ruidos o por objetos del entorno (moscas, pájaros, roedores...) sean contados cómo abejas.

En la siguiente imagen podemos ver la salida del algoritmo:

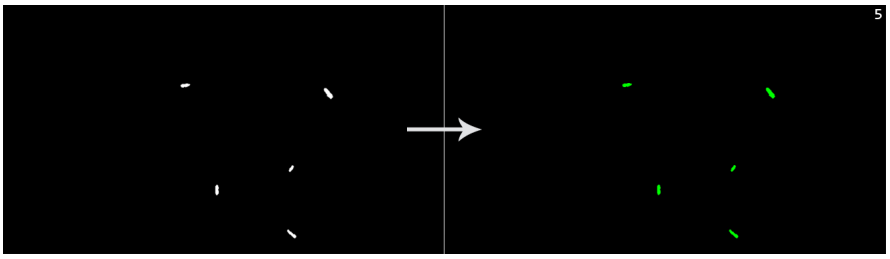

Figura 3.5: Resultado de la fase de detección y conteo de abejas.

En esta otra se puede apreciar como se descartan las tres moscas que hay en la imagen ya que su área es inferior al área mínima:

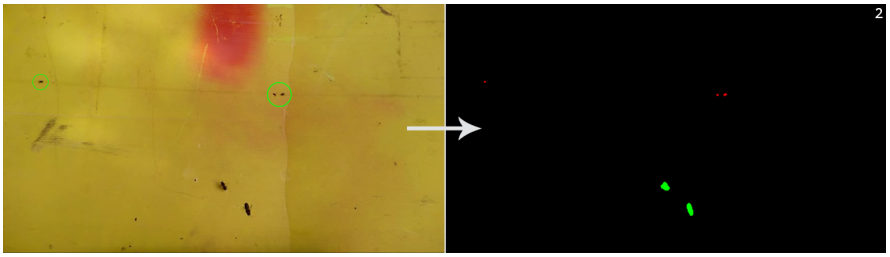

Figura 3.6: Resultado del algoritmo en una imágen con moscas.

Técnicas y herramientas

Esta parte de la memoria tiene como objetivo presentar las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto. Si se han estudiado diferentes alternativas de metodologías, herramientas, bibliotecas se puede hacer un resumen de los aspectos más destacados de cada alternativa, incluyendo comparativas entre las distintas opciones y una justificación de las elecciones realizadas. No se pretende que este apartado se convierta en un capítulo de un libro dedicado a cada una de las alternativas, sino comentar los aspectos más destacados de cada opción, con un repaso somero a los fundamentos esenciales y referencias bibliográficas para que el lector pueda ampliar su conocimiento sobre el tema.

Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo. Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación. Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales. Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas de la base de datos, normalización y desnormalización, distribución en ficheros3, reglas de negocio dentro de las bases de datos (EDVHV GH GDWRV DFWLYDV), aspectos de desarrollo relacionados con el WWW... Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.

Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliografía

[1] Daniel Lélis Baggio. *OpenCV 3.0 Computer Vision with Java*. Packt Publishing, July 2015.

[2] Jason Campbell, Lily Mummert, and Rahul Sukthankar. Video Monitoring of Honey Bee Colonies at the Hive Entrance. *Visual observation analysis of animal insect behavior ICPR*, 8:1–4, 2008. URL http://homepages.inf.ed.ac.uk/rbf/VAIB08PAPERS/vaib9{_}mummert.pdf.

[3] Jennifer M Campbell, Douglas C Dahn, and Daniel A J Ryan. Capacitance-based sensor for monitoring bees passing through a tunnel. *Measurement Science and Technology*, 16(12):2503–2510, 2005. ISSN 0957-0233. doi: 10.1088/0957-0233/16/12/015. URL http://stacks.iop.org/0957-0233/16/i=12/a=015https://docs.google.com/file/d/0Bz7WbjRT9mlmTW1ac3R0azEycDA/view.

[4] Guillaume Chiron, Petra Gomez-Krämer, and Michel Ménard. Detecting and tracking honeybees in 3D at the beehive entrance using stereo vision. *EURASIP Journal on Image and Video Processing*, 2013(1):59, 2013. ISSN 1687-5281. doi: 10.1186/1687-5281-2013-59. URL http://jivp.eurasipjournals.com/content/2013/1/59https://hal.inria.fr/hal-00923374/document.

[5] Guillaume Chiron, Petra Gomez-Krämer, Michel Ménard, and Fabrice Requier. 3D tracking of honeybees enhanced by environmental context. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8156 LNCS, pages 702–711. Springer Berlin Heidelberg, 2013. ISBN 9783642411809. doi: 10.1007/978-3-642-41181-6_71. URL http://link.springer.com/10.1007/978-3-642-41181-6{_}71.

[6] Express Label Company. Barcode labels many uses, 2009. URL http://www.uprintlabels.com/barcode-labels-uses.html. [Online; Accedido 02-Noviembre-2016].

[7] Axel Decourtye, James Devillers, Pierrick Aupinel, François Brun, Camille Bagnis, Julie Fourrier, and Monique Gauthier. Honeybee tracking with microchips: A new methodology to measure the effects of pesticides. *Ecotoxicology*, 20(2):429–437, 2011. ISSN 09639292. doi: 10.1007/s10646-011-0594-4.

[8] Luis del Valle Hernández. Detección de movimiento con opencv y python, 2016. URL http://programarfacil.com/blog/vision-artificial/deteccion-de-movimiento-con-opencv-python/. [Online; Accedido 5-Octubre-2016].

[9] Greenpeace Laboratories Research. Bees in Decline put pollinators and agriculture. page 48, 2013. URL http://www.greenpeace.org/switzerland/Global/international/publications/agriculture/2013/BeesInDecline.pdf.

[10] J. K. Kaplan. Colony Collapse Disorder - A Complex Buzz. *Agricultural Research*, (June):8–11, 2008. ISSN 00027626. URL https://agresearchmag.ars.usda.gov/AR/archive/2008/May/colony0508.pdf.

[11] Salil Kapur. *Mastering OpenCV Android Application Programming*. Packt Publishing, July 2015.

[12] Liyuan Li, Weimin Huang, Irene Y. H. Gu, and Qi Tian. Foreground Object Detection from Videos Containing Complex Background. In *Proceedings of the Eleventh ACM International Conference on Multimedia*, MULTIMEDIA '03, pages 2–10, New York, NY, USA, 2003. ACM. ISBN 978-1-58113-722-4. doi: 10.1145/957013.957017. URL http://doi.acm.org/10.1145/957013.957017.

[13] David Miguel Lozano. Gobees en google play, 2016. URL https://play.google.com/store/apps/details?id=com.davidmiguel.gobees. [Online; Accedido 25-Enero-2017].

[14] David Miguel Lozano. Repositorio de las herramientas desarrolladas para gobees en github, 2016. URL https://github.com/davidmigloz/go-bees-prototypes. [Online; Accedido 25-Enero-2017].

[15] David Miguel Lozano. Repositorio de gobees en github, 2016. URL https://github.com/davidmigloz/go-bees/. [Online; Accedido 25-Enero-2017].

[16] David Miguel Lozano. Sitio web de gobees, 2016. URL http://gobees.io/. [Online; Accedido 25-Enero-2017].

[17] A. E. Lundie. The flight activities of the honeybee. *United States Department of Agriculture*, 1328:1–38, 1925. ISSN 0717-6163. doi: 10.1007/s13398-014-0173-7.2. URL https://archive.org/details/flightactivities1328lund.

[18] University of Pennsylvania. Robotics: Estimation and learning: Gaussian mixture model (gmm), 2016. URL https://www.coursera.org/learn/robotics-learning/lecture/XG0WD/1-4-1-gaussian-mixture-model-gmm/. [Online; Accedido 5-Octubre-2016].

[19] OpenCV. Interfaz backgroundsubtractormog2 (background_segm.hpp), 2016. URL https://github.com/opencv/opencv/blob/master/modules/video/include/opencv2/video/background_segm.hpp. [Online; Accedido 12-Octubre-2016].

[20] OpenCV. Implementación backgroundsubtractormog2 (bgfg_gaussmix2.cpp), 2016. URL https://github.com/opencv/opencv/blob/master/modules/video/src/bgfg_gaussmix2.cpp. [Online; Accedido 12-Octubre-2016].

[21] OpenCV. Opencv background subtraction tutorial, 2016. URL http://docs.opencv.org/master/db/d5c/tutorial_py_bg_subtraction.html. [Online; Accedido 26-Septiembre-2016].

[22] OpenCV. Color conversions, 2016. URL http://docs.opencv.org/3.1.0/de/d25/imgproc_color_conversions.html. [Online; Accedido 18-Octubre-2016].

[23] OpenCV. Contours in opencv, 2016. URL http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_contours/py_table_of_contents_contours/py_table_of_contents_contours.html. [Online; Accedido 19-Octubre-2016].

[24] OpenCV. Structural analysis and shape descriptors, 2016. URL docs.opencv.org/3.0-beta/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html. [Online; Accedido 19-Octubre-2016].

[25] OpenCV. cv::backgroundsubtractormog2 class reference, 2016. URL http://docs.opencv.org/3.1.0/d7/d7b/classcv_1_1BackgroundSubtractorMOG2.html. [Online; Accedido 12-Octubre-2016].

[26] M H Struye, H J Mortier, G Arnold, C Miniggio, and R Borneck. Microprocessor-controlled monitoring of honeybee flight activity at the hive entrance. *Apidologie*, 25(4):384–395, 1994. ISSN 0044-8435. doi: 10.1051/apido:19940405. URL http://cat.inist.fr/?aModele=afficheN{&}cpsidt=4192550https://hal.archives-ouvertes.fr/hal-00891170/documenthttp://www.apidologie.org/10.1051/apido:19940405.

[27] Rahman Tashakkori and Ahmad Ghadiri. Image processing for honey bee hive health monitoring. In *SoutheastCon 2015*, pages 1–7, Fort Lauderdale, FL, apr 2015. IEEE. ISBN 978-1-4673-7300-5. doi: 10.1109/SECON.2015.7133029. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7133029.

[28] Wikipedia. Background subtraction — wikipedia, the free encyclopedia, 2016. URL https://en.wikipedia.org/w/index.php?title=Background_subtraction&oldid=741230478. [Online; Accedido 23-Septiembre-2016].

[29] Wikipedia. Gaussian blur — wikipedia, the free encyclopedia, 2016. URL https://en.wikipedia.org/w/index.php?title=Gaussian_blur&oldid=739396767. [Online; Accedido 18-Octubre-2016].

[30] Wikipedia. Grayscale — wikipedia, the free encyclopedia, 2016. URL https://en.wikipedia.org/w/index.php?title=Grayscale&oldid=742147487. [Online; Accedido 18-Octubre-2016].

[31] Li Yao and Miaogen Ling. An Improved Mixture-of-Gaussians Background Model with Frame Difference and Blob Tracking in Video Stream. *The Scientific World Journal*, 2014, April 2014. ISSN 2356-6140. doi: 10.1155/2014/424050. URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4000660/.

[32] Zoran Zivkovic. Improved Adaptive Gaussian Mixture Model for Background Subtraction. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 2 - Volume 02*, ICPR '04, pages 28–31, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 978-0-7695-2128-2. doi: 10.1109/ICPR.2004.479. URL http://dx.doi.org/10.1109/ICPR.2004.479.

[33] Zoran Zivkovic and Ferdinand van der Heijden. Efficient Adaptive Density Estimation Per Image Pixel for the Task of Background Subtraction. *Pattern Recogn. Lett.*, 27(7):773–780, May 2006. ISSN 0167-8655. doi: 10.1016/j.patrec.2005.11.005. URL http://dx.doi.org/10.1016/j.patrec.2005.11.005.