DRON polinizador de cultivos. Detección de flores con algoritmos de visión artificial

CASTELLANOS-SERRANO, Luis Tonatiuh, CERVANTES-BAZÁN, Josué Vicente y LÓPEZ-CHIMIL, Humberto

L. Castellanos´, J. Cervantes´´, H. López´´´

´ Profesor-Investigador del Tecnológico de Estudios Superiores de Ixtapaluca (TESI)
´´Profesor-Investigador de la Universidad Autónoma del Estado de México
´´´ Profesor-Investigador de la Universidad Autónoma Chapingo
procesoslcce@hotmail.com

F. Pérez, E. Figueroa, R. García, L. Godínez (eds.) Ciencias de la Biología, Agronomía y Economía. Handbook T-I.-

Abstract

Drones have become a technology in the vogue of the 21st century, its first applications are merely military, but today its range of application is unlimited. One of the important topics of the discussion has been the low population decline of bees, colloquially named "Bee Colony Syndrome", showing different factors in the different populations of the country, there are different methods to counteract this deplorable desertion of The bees, so in this article we present technical conceptualization and engineering, for the design of a DRON crop pollinator. The construction of a DRON pollinator is not a simple task and involves multiple branches of engineering, because the article focuses on the discussion of the artificial vision aspect for the detection of flowers, explaining in detail the methodology and procedures carried out by a cable To obtain a cascade file classifier by the Viola-Jones algorithm, better known as haartraining. Topics such as the OpenCV library, the process for creating a classifier for object recognition, evaluation tests for object recognition and the results obtained have been discussed.

10 Introducción

Las tecnologías agronómicas son hoy en día una discusión que han tomado suma importancia en los temas mundiales, la hambruna, desnutrición, la mala calidad de alimentos, etc. Son indicios graves que deben atenderse a la brevedad, uno de los sectores de alta importancia, son los cultivos dependientes de los polinizadores de cultivos, Berenice González Durand en su artículo “México pierde cultivos por falta de polinizadores” nos dice:

> Se pierde casi la mitad del valor total de los cultivos dependientes de la polinización. El calentamiento global y el uso de pesticidas han mermado sus poblaciones. (Durand, 2015)

Esto nos hace realizar el ejercicio de la reflexión valorando la importancia de las especies polinizadoras, mostrándonos estadísticas sorprendentes que nos deben conducir al ejercicio de la reflexión, mostrándonos lo importante que son las especies polinizadoras en nuestra vida diaria, y el caos que la humanidad está llegando al amenazar unos de los eslabones de la cadena alimenticia.

Un DRON en su forma básica se define como una aeronave no tripulada, hoy en día sus aplicaciones son multipropósito, que van desde hobbistas hasta aplicaciones militares. El presente documento propone el diseño de un dron polinizador de cultivos, el cual en su estructura general para poder ser concebirlo debe de desglosarse en diferentes fases de desarrollo de ingeniería, entre las etapas más importante podemos enlistar:

- Sistema de control.
- Hardware embebido.
- Algoritmos genéticos de inteligencia artificial.
- Procesamiento de visión artificial.
- Diseño mecánico de fuselaje.
- Sistema aerodinámico.

El presente artículo pretende aborda la discusión técnica del proceso de visión artificial, mostrando la metodología de diseño de algoritmos computacionales para crear sistemas inteligentes que detecten flores de tiempo real, esto con la premisa de poder incorporarlo en un futuro no muy lejano y crear un prototipo funcional de dron polinizador de cultivos.

10.1 Metodología de detección de flores con visión artificial

10.1.1 Detección de objetos tiempo real

La librería OpenCV es una recopilación de instrucciones de software libre para el diseño de algoritmos de visión artificial.

La librería OpenCV en su página oficial 2016 nos dice: "OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

"The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms" (Copyright, 2016)

10.1.2 Entrenamiento haartraining

Una de las herramientas que OpenCV ofrece, se llama Haartraining, es un complejo algoritmo para el reconocimiento de imágenes. En la documentación que se puede consultar de forma gratuita en la Web de OpenCV, explican de manera completa el fundamento matemático en que se basa el algoritmo para su justificación y además los parámetros que se deben de ingresar para ejecutar los comandos. (Rezaei, 2011) .

10.1.3 Entrenamiento haartraining de un objeto de prueba

Para corroborar en una primera fase los pasos necesarios para realizar un entrenamiento haartraining, se procede a detectar un objeto de prueba que sirva como experimento para obtener la metodología adecuada de cómo llevar con éxito un entrenamiento de este tipo.

El primer paso es seleccionar un objeto de prueba, en este caso se escogió una flor artificial de girasol, vista de la parte enfrente, como se muestra a continuación:

Figura 10 Objeto de prueba

Para realizar un entrenamiento que sirva para detectar objetos en tiempo real, es necesario contar con muestras positivas y negativas. Las muestras positivas son imágenes donde se contiene el objeto que se desea detectar y las imágenes negativas son imágenes que no contienen el objeto que se desea detectar. Para ello se crea un catálogo amplio de muestras positivas y muestras negativas, que no es más que una colección de imágenes que puede estar en formato .jpg o .bmp, o cualquier otro formato soportado por el entrenador de OpenCV (consultar documentación), para el presente caso se ocupan imágenes en formato .jpg, ya que el hecho de usar imágenes en .bmp aumenta el peso de KB de las imágenes, dándoles mejor calidad y en consecuencia haciendo más tardado el entrenamiento por el aumento de resolución.

Para el entrenamiento de prueba se recolectan 100 imágenes positivas y 300 imágenes negativas que se procede a almacenar en 2 carpetas diferentes.

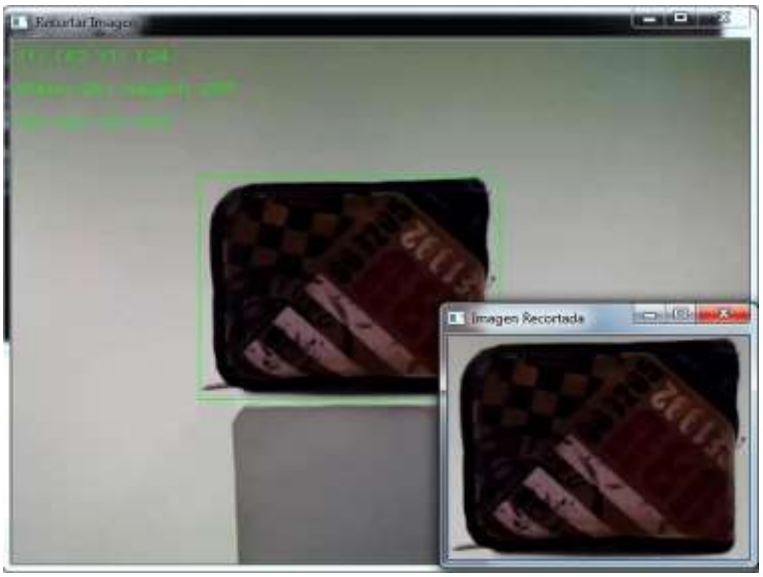
Una vez obtenidas las muestras positivas y negativas es necesario crear un fichero de direccionamiento que le sirva a el entrenador de OpenCV como guía de donde están resguardadas las imágenes, solo que dichos ficheros llevan información diferente, en cualquiera de los dos casos estos ficheros son archivos de Bloc de notas con extensión ".txt".

Para el caso de las imágenes positivas se debe agregar la misma información que el fichero de las muestras negativas, más, la cantidad de objetos detectados, el punto "x, y" el ancho y largo donde se encuentra el objeto, su estructura es la siguiente:

Ruta_de_la_carpeta/Nombre_imagen.Extención - #Objetos Coordenadas - x - y (Punto Inicial) - Ancho - Largo

En la siguiente Figura puede apreciarse mejor la idea:

Figura 10.1 Programa ObjectMarker diseñado para la obtención de posiciones del objeto



El punto es tener una imagen que posea el objeto que se desea detectar, obtener el número de objetos que tiene, sus coordenadas de punto de inicio, su ancho y largo, para que el programa pueda trazar un rectángulo que le ayude a delimitar donde está el objeto que se desea detectar.

Para poder realizar el trabajo de una forma más sencilla y dinámica, se procede a diseñar una herramienta que se nombra "ObjectMarker", un programa diseñado en Visual C++, que tiene como objetivo facilitar la obtención del fichero de muestras positivas, en la imagen 3 se puede apreciar cómo se ejecuta. (Comunidad de Programadores de Software Libre del Ecuador, 2010)

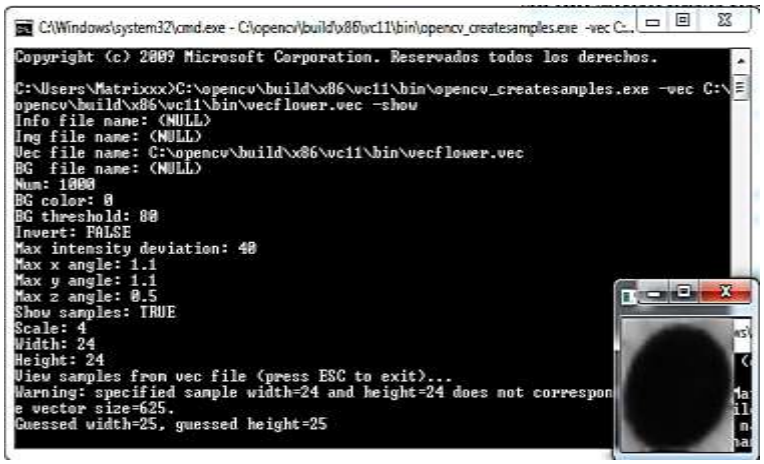
Una vez obtenidos los ficheros de las muestras positivas y negativas, se recomienda copiar las 2 carpetas que contienen las imágenes y también los dos archivos .txt a la dirección C:\opencv\build\x86\vc11\bin, lugar donde se encuentran los programas para el entrenamiento.

El siguiente paso es ejecutar un programa llamado "opencv_createsamples" mismo que se encuentra en la ruta ya antes mencionada, el programa debe ejecutarse desde la consola, por lo que es necesario abrir el CMD de Windows, y ejecutar el siguiente comando:

opencv-createsamples -info -vec -num -w -h (Comunidad de Programadores de Software Libre del Ecuador, 2010)

Una vez ejecutado el programa, no demora mucho en crear un archivo ".vec" con el nombre y la ruta que se le específica, para corroborar que el programa haya realizado de forma exitosa la operación puede ejecutar el siguiente comando:

Figura 10.2 Ejecución del comando para verificar muestras vectoriales desde CMD



Como se observa en la figura anterior al momento de ejecutar el comando, se despliega una nueva ventana, que muestra en escala de grises las partes que delimitan con el "ObjectMarker" de las imágenes positivas que interesa detectar, con ello el resultado es un archivo del tipo vectorial que contiene todas las muestras positivas en un solo ancho, largo y en escala de grises.

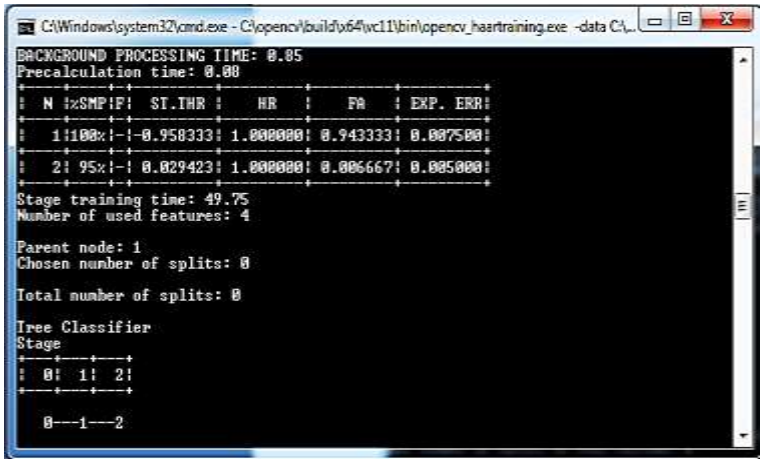
Teniendo el archivo vectorial se ejecuta el entrenamiento de cascada, en el presente caso con el programa "opencv_haartraining", los parámetros que se deben de tener en cuenta para invocar este comando son:

opencv-haartraining -data -vec -bg -nstages -nsplit -minhitrate -maxfalsealarm -npos -nneg -w -h -nonsym -mem -mode ALL (IOSR Journal of Computer Engineering, 2014)

Con los datos anteriores se ejecuta el programa desde CMD.

Al ejecutar el haartraining, se puede apreciar en la consola, como se van estructurando los arboles de entrenamiento del clasificador, en la siguiente imagen se puede ver como el clasificador va ensamblando los árboles. (Serrano, 2015)

Figura 10.3 Estructuración de los árboles de entrenador haartraining



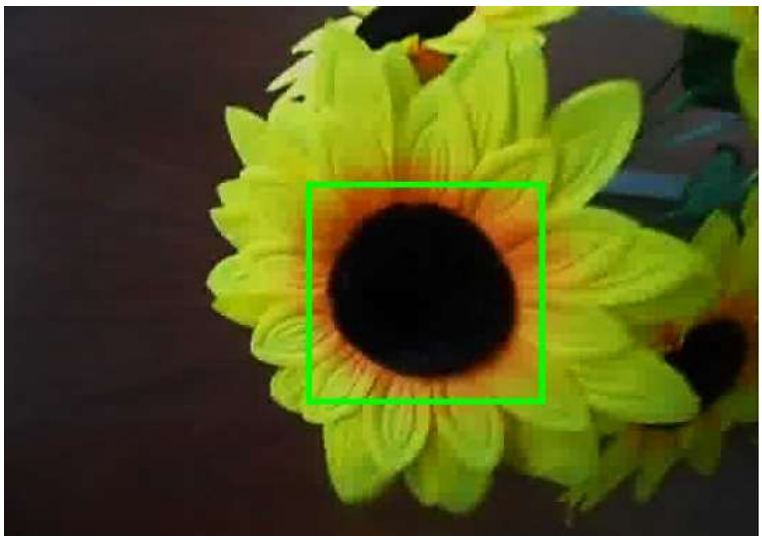
Una vez concluido el entrenamiento se crea una carpeta que contiene subcarpetas con el índice total de etapas que se usan para clasificar el entrenador, en el presente caso solo se necesitan 8 etapas de las 25 que se indexan en los parámetros.

El entrenador puede quedarse atorado en una cierta etapa y no completar el entrenamiento, esto se debe a que el programa encuentra incongruencias en la comparación de las muestras negativas con la muestras vectoriales, quedando encerrado en bucle infinito que lo detiene, por ello es necesario forzar el cierre del programa, y una vez terminado el entrenamiento con el número de etapas correspondientes se tiene que ejecutar un archivo que proporciona OpenCV llamado "Converte_cascade.c" la ejecución del programa en Windows exige cargarlo desde una plataforma que soporte programación en C, invocarlo desde Linux es una opción más rápida y sencilla para realizar dicha tarea, solo tiene que direccionarle donde está la carpeta de nuestro entrenamiento, el programa ejecuta el algoritmo para crear un archivo ".xml", que es proporcional al producto final del entrenamiento generado.

Finalmente se obtiene el archivo de entrenamiento vertido en un clasificador de extensión ".xml" que hace referencia a archivos de tipo cascada.

Los resultados de detección no son tan buenos, ya que se usan pocas muestras positivas y negativas, pero el algoritmo logra detectar en tiempo real el centro de un girasol artificial con una tasa de fallo que pueden mejorarse con el aumento de las muestras. En la siguiente Figura se presencia el resultado de la fase de detección del objeto de prueba:

Figura 10.4 Detección de objeto de prueba en tiempo real



10.1.4 Entrenamiento haartraining de flores

Los pasos listados en el apartado anterior sirven como punto de partida para realizar el entrenamiento deseado, en el presente proyecto se propone dotar al dron de visión artificial para poder detectar flores en tiempo real.

Como fase 1 del proyecto se ha ensamblado un ambiente artificial para poder operar al dron dentro de un campo de prueba, que permita calibrar los errores y realizar los ajustes necesarios, en la siguiente Figura se puede apreciar una vista detallada de la maqueta:

Figura 10.5 Ambiente artificial para pruebas

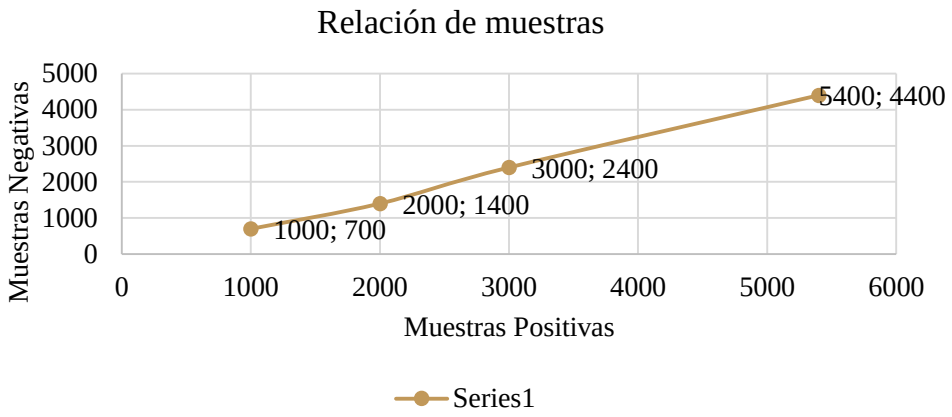
Este ambiente artificial, sirve como punto de partida, para poder simular las condiciones de un invernadero, lugar donde se pretende poner a prueba el dron. Se realizan 4 entrenamientos haartraining tomando diferente número de muestras positivas y negativas, en la siguiente tabla se observan los resultados.

Tabla 10 Datos obtenidos de entrenamientos realizados

Entrenamiento	Muestras Positivas	Muestras Negativas	Numero de Etapas	Arboles creados
1	1000	700	20	14
2	2000	1400	20	18
3	3000	2400	20	19
4	5400	4400	25	18

Como se observa el número muestras positivas es mayor al número de muestras negativas, al momento de ejecutar el entrenador se obtienen errores que bloquean el programa, enviando en la consola un código de error que hace referencia a la falta de muestras positivas (Navarro, 2004), tras varios intentos se detecta que la relación de muestras positivas y negativas obedece una reciprocidad, que influye como error del algoritmo Haar, en el siguiente gráfico se puede observar la respuesta de la correlación.

Gráfico 10 Gráfico representativo de la correlación de muestras positivas y negativas, para un eficiente entrenamiento



Fuente: (Serrano, 2015)

Si se realizan los cálculos pertinentes el gráfico muestra que entre mayor sean las muestras positivas mayor deben ser el número de muestras negativas y también con el aumento de muestras positivas se debe aumentar el número de etapas del entrenamiento, para ello se establece el siguiente argumento, obtenido a partir de métodos gráficos y experimentales:

"La cantidad de muestras negativas deben de ser del 70%-80% de las muestras positivas usadas en el archivo .vec creado"

Esta relación puede observarse en el gráfico de arriba, ya que el eje de las abscisas influye un aumento poco abrupto en el eje de las ordenadas de manera lineal, cambiando la pendiente de las tres rectas consecutivas, que representan la relación.

Otra forma puede ser valiéndose del gráfico (ver Gráfico 10) para calcular el número de muestras negativas, dibujando una línea perpendicular a la variable que se desea conocer.

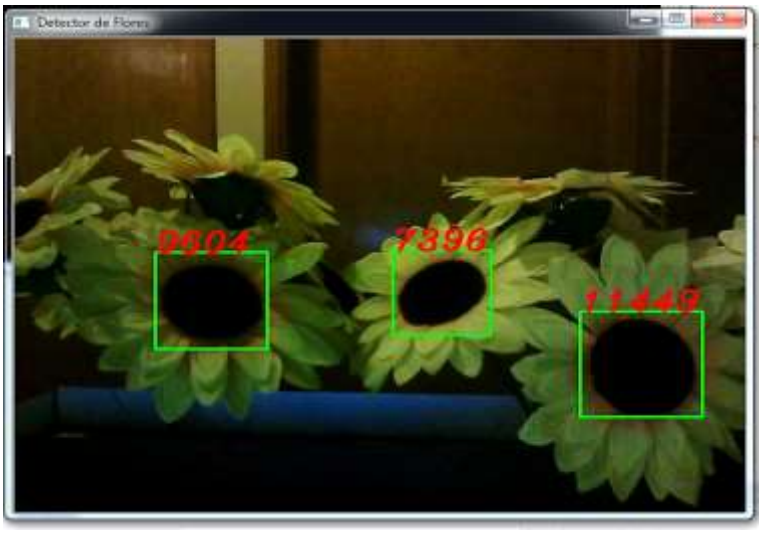
Obedeciendo esto, se tiene éxito en la ejecución del entrenamiento y evita que el programa haartarinig envié un código de error perteneciente a que las muestras positivas sean insuficientes.

Es importante que cuando se ejecute el entrenador haartraining, la opción de -nonsym se cargue dentro del comando, ya que esto indica que las imágenes vectoriales del entrenador no son simétricas, cuando esta simetría no se especifica, el entrenador toma las muestras vectoriales como imágenes simétricas, por lo tanto solo toma en cuenta la mitad de la imagen para analizar y duplicar el resultado, en el presente caso al no ser simétricas afecta de manera considerable el entrenamiento, aunque también requiere muchos más recursos del sistema. (Asenjo & Cabeza Laguna, 2011).

Finalmente, al cargar el archivo ".xml" obtenido se puede apreciar cómo, conforme se van aumentando el número de muestras la clasificación mejora redituablemente.

En la siguiente Figura se puede apreciar la detección de flores en tiempo real del último entrenamiento realizado en el laboratorio.

Figura 10.6 Prueba de detección de flores en tiempo real



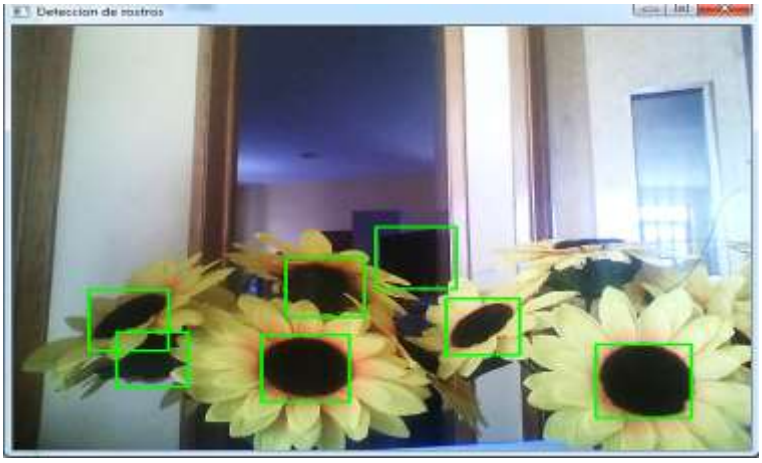
10.1.5 Mejoramiento de detección de objetos

10.1.5.1 Falsos positivos

En tabla 10 se puede observar que se crean 4 clasificadores, estos clasificadores mejoran su respuesta en función del aumento de las muestras, pero la razón importante por el aumento de las muestras, es por la detección de cosas que no pertenecen al catálogo de objetos del archivo ".vec" y también por la falla de detección de objetos que supuestamente el programa debe detectar.

Cuando un objeto que es detectado no debe de ser reconocido por el clasificador se le nombran *falsos positivos*, errores de detección provocados por el clasificador, como se puede apreciar en la siguiente Figura:

Figura 10.7 El clasificador detecta objetos fuera del catálogo de muestras, como los tallos de las flores y partes del fondo



Como se explica en la imagen, el clasificador detecta objetos que no forman parte de las muestras positivas, por lo tanto estos errores son *falso positivos*, para pulir dichos errores es necesario ampliar el catálogo de muestras negativas colocando imágenes de estos objetos indeseables, con ello, el clasificador mejora redituablemente la detección.

10.1.5.2 Falsos negativos.

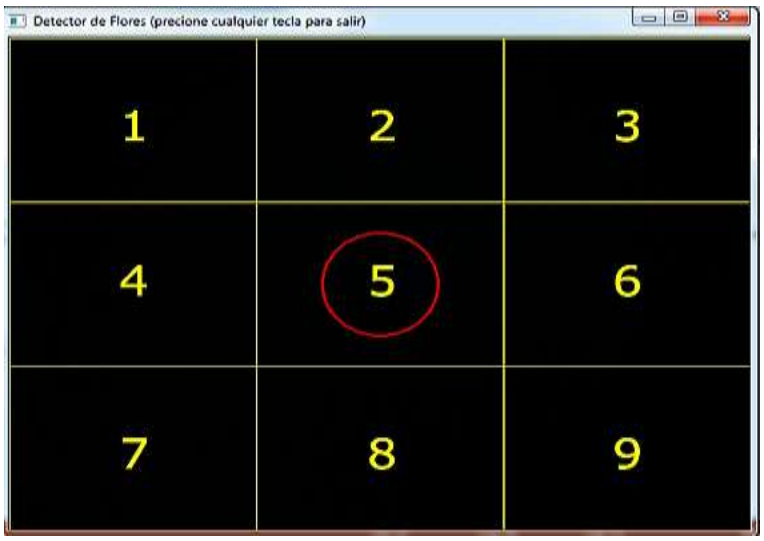
En el caso contrario cuando el clasificador no logra detectar un objeto que forma parte de las muestras positivas del archivo ".vec", estos se nombran *falsos negativos* ya que los objetos que supuestamente deben ser detectadas por el clasificador, son tomadas como imágenes negativas. Para contrarrestar esta carencia en el clasificador, es necesario aumentar el número de muestras positivas, para crear un catálogo más grande y mejorar la calidad de detección de los objetos positivos.

10.1.6 Algoritmo de detección de flores en C++

Con el diseño de un buen clasificador se obtiene como producto final un archivo ".xml" que contiene los índices necesarios del algoritmo Haar para la detección del objeto. Con ello se procede a diseñar una interfaz en C++ que permita la detección del objeto y poder enviar los comandos necesarios de la posición del objeto para ser enviados al dron.

Para ello lo primero que se implementa es cuadricular la pantalla en 9 cuadrantes:

Figura 10.8 Distribución de los cuadrantes de operación

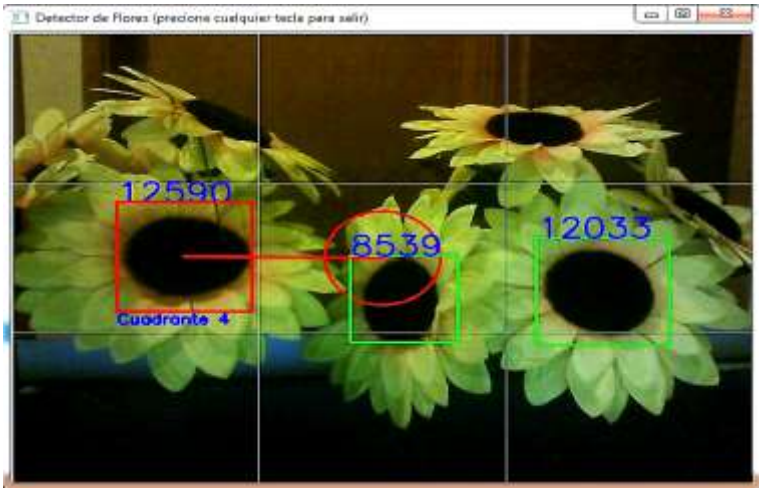


A partir de ello se tienen 9 cuadrantes donde puede estar el objeto que se examina (en la imagen anterior se enumeraron la lógica de los cuadrantes donde el programa realiza la distribución).

Lo siguiente es detectar el cuadrante donde se encuentra dicho objeto y además cual flor es mejor candidato para extraer la muestra de polen, en una ramo de flores puede haber más de una flor, por lo tanto el programa no solo detecta el cuadrante donde está la flor sino también el área del rectángulo de detección más grande para determinar cuál objeto está más cerca, y con ello, el dron en base a esta información pueda tomar decisiones y ajustar el vuelo para guiar al objeto en el centro del círculo rojo, preparándolo de esta manera, para poder tomar la muestra de polen con el utensilio de extracción.

Finalmente, la visión artificial del dron está dotada, con un complejo algoritmo computacional, que da como resultado la siguiente respuesta visual:

Figura 10.9 Detección de flores en tiempo real



Como se pude apreciar en la imagen anterior, el programa detecta las flores que cumplen con las características de muestras positivas del clasificador ".xml" (en el presente caso detectaron tres objetos) calcula el área de los rectángulos que los delimitan mostrándolo en letras azules, y evalúa cual es el objeto con mayor área siendo este el más cercano. Cuando el programa sabe cuál es el más grande, pinta de rojo el objeto detectado, procede a dibujar una línea que permite al usuario ver la distancia del centro con respecto al objeto y finalmente muestra en pantalla el cuadrante donde se encuentra el objeto.

10.2 Interfaz Gráfica

La plataforma diseñada en C++, cumple como propósito principal depurar las librerías de OpenCV para realizar el procesamiento de imagen y obtener como resultado, la visión artificia para el dron. Para la comunicación directa con el Dron, se ha implementado un sistema de conexión USB, para mandar la salida de las señales equivalentes a los comandos con los que opera el dron.

En primera instancia, la interfaz gráfica es diseñada y programada en Visual Basic 2012, la función principal que se pretende con la implementación de la interfaz gráfica es:

- Anexar un módulo que permite el envío de comandos por vía USB por interfaz HID (*Human Interface Device*), o su traducción directa *Dispositivos de Interfaz Humana*, que son todos aquellos dispositivos que generan una interfaz directa con los seres humanos.
- Proveer al usuario una interfaz gráfica amigable con la cual pueda interactuar de forma sencilla sin la necesidad de adentrarse en la programación.
- Proporcionar una consola que permita operar el dron y tener un status visual y dinámico de su estado.
- Generará un archivo ejecutable que instale toda la paquetería necesaria de forma sencilla.

10.2.1 Intercomunicación de C++ con Visual Basic

Existen muchos métodos y formas para comunicar diferentes lenguajes de programación, Visual Basic se encuentra en un nivel alto de programación usando un enfoque orientado a objetos para su lógica de declaración, mientras que C++ es encuentra en un nivel medio, empelando programación estructurada para el diseño de sus algoritmos.

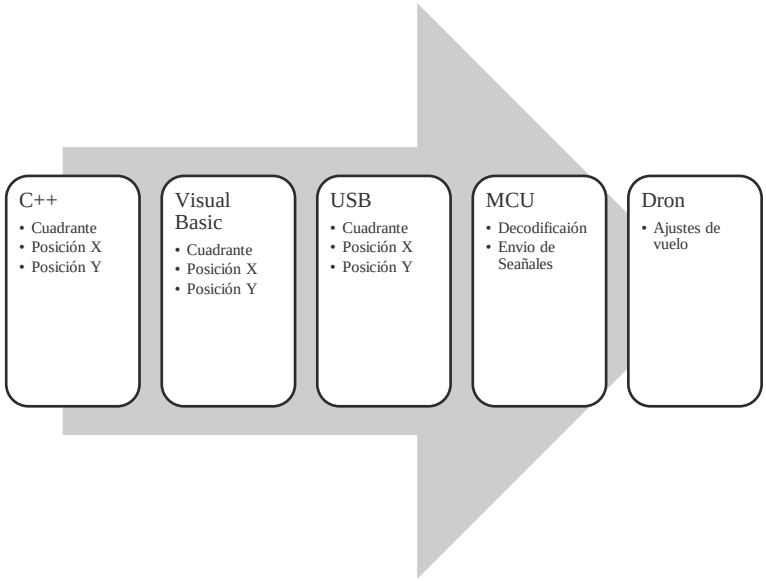
Como Visual y C++ con dos plataformas de comunicación completamente distintas, se tiene que indagar sobre los métodos de importación de datos de un programa a otro.

Al ejecutar la visión artificial de C++, y permitiendo que el compilador realice todas las rutinas programadas, como resultado final de todos los algoritmos obtienen tres variables de suma importancia:

1. El número de cuadrante donde se detectó el objeto más grande.
2. La posición "X" del objeto.
3. La posición "Y" del objeto.

Esas tres variables, son la información necesaria que el dron necesita para realizar los ajustes competentes del vuelo, por ello es necesario hacerle llegar la información a la interfaz de Visual Basic, para que a su vez Basic, envíe los datos por medio del USB hasta un microcontrolador que decodifique los comandos y puedan ser enviados al dron, en el siguiente diagrama se aprecia mejor el flujo de información.:

Figura 10.10 Flujo de información de las variables de ajuste de vuelo



10.3 Resultados Finales. Interfaz gráfica de usuario (GUI)

Finalmente, en la siguiente imagen se puede apreciar la interfaz gráfica diseñada, misma plataforma es el medio donde el operador interactúa con el software y hardware de todo el sistema esbozado, y poder operar de una forma fácil y dinámica el dron polinizador:

Figura 10.11 Consola de Principal, interfaz gráfica final diseñada en Visual Studio 2012

Como se puede observar la interfaz gráfica es una consola simple con recuadros de status para monitorear los cambios efectuados durante la depuración del programa, dos botones de operación y un contenedor para visualizar el video obtenido de la fuente de videograbación.

La interfaz gráfica es un software sencillo y dinámico, cuenta con un botón para empezar el proceso de reconocimiento de visión artificial, al igual que el botón para terminar el proceso de reconocimiento, el cuadro de detección ubica el cuadrante donde ha sido detectado el centro de la flor, como se mencionó anteriormente la plataforma cuenta con 9 cuadrantes de detección. Finalmente se pueden apreciar dos ventanas de Status, donde en la primera ventana en un objeto "ListBox" se revisa la escritura de los ficheros con los índices de detección, y el segundo recuadro checa que la conexión del sistema hardware embebido vía USB sea exitosa, el cual será el medio para él envió de información de manera remota con las instrucciones del vuelo del DRON.

10.4 Conclusiones

La viabilidad del clasificador obtenido para la detección del cultivo de prueba supera las expectativas esperadas, el proyecto debe encaminarse en pruebas de campo para la recolección de muestras positivas y negativas de cultivos reales. Los contextos de estudio del clasificador obtenidos son ejecutados en condiciones óptimas, ya que los entornos de las pruebas realizadas tienen por ejemplo la cantidad de luz requerida, una clara pigmentación en las flores, anulación de factores externos en la lente de la cámara, el habitad de estudio sin agentes terceros, etc. En esta primera fase de estudios los resultados fueron satisfactorios, se obtuvo una detección aceptable con pocos falsos positivos, permitiendo comprobar el principio de funcionamiento del sistema y poder argumentar, que, con mayor tiempo e inversión en esta investigación, en un futuro próximo, no cabe duda que será posible crear un DRON polinizador de cultivos, volviéndose una herramienta activa del sector de agricultura.

Cabe mencionar que el presente proyecto no viene a sustituir a los polinizadores, ni tampoco coadyuvar como alternativa para la extinción de las especies naturales, el lazo entre hombre naturaleza es una correspondencia biunívoca que pende de un hilo muy delicado entre una simbiosis que garantice el equilibrio del ecosistema, la discusión de la dicotomía entre tecnología y preservación, es un argumento ético que se debe hacerse valido, y no buscar que la era tecnológica arrase con los principios éticos y morales, que en vez de llevarnos a la prosperidad de la humanidad, sean partícipes de nuestra propia destrucción.

10.5 Referencias

Asenjo, R. S., & Cabeza Laguna, R. (31 de Enero de 2011). *EVALUACIÓN EXPERIMENTAL DE DETECTORES DE.* Obtenido de http://academica-e.unavarra.es/bitstream/handle/2454/3143/0000577399.pdf?sequence=1

Comunidad de Programadores de Software Libre del Ecuador. (8 de Abril de 2010). *Blog de Palichis.* Obtenido de Entrenar opencv : http://coplec.org/?q=2010%2F08%2F05%2Fentrenar-opencv
Copyright. (23 de Diciembre de 2016). *OpenCV team* . Obtenido de http://opencv.org/about.html

Durand, B. G. (26 de Enero de 2015). *El Universal*. Obtenido de http://archivo.eluniversal.com.mx/ciencia/2015/mexico-cultivos-polinizadores-100570.html

ELECTRONICS AND ELECTRICAL ENGINEERING . (29 de Junio de 2011). *SYSTEM ENGINEERING, COMPUTER TECHNOLOGY.* Obtenido de http://www.ee.ktu.lt/journal/2011/10/18__ISSN_1392-1215_Inspection%20System%20based%20on%20Computer%20Vision.pdf

IOSR Journal of Computer Engineering. (Mayo-Junio de 2014). *Traffic Event Detection using Computer Vision.* Obtenido de Www.iosrjournals.org: http://www.iosrjournals.org/iosr-jce/papers/Vol16-issue3/Version-2/F016322532.pdf

Navarro, R. U. (Julio de 2004). *Aplicaci´on de t´ecnicas de Boosting.* Obtenido de http://www.gpiv.upv.es/publications/pdf/pfc_raquel.pdf

Rezaei, M. (2011). *www.MahdiRezaei.com.* Obtenido de https://www.cs.auckland.ac.nz/~m.rezaei/Tutorials/Creating_a_Cascade_of_Haar-Like_Classifiers_Step_by_Step.pdf

Serrano, L. T. (2015). *DISEÑO DE DRON POLINIZADOR DE CULTIVOS. DETECCIÓN DE CULTIVOS CON VISIÓN ARTIFICIAL.* Ixtapaluca, Estado de México: Teconologico de Estudios Superiores de Ixtapaluca.