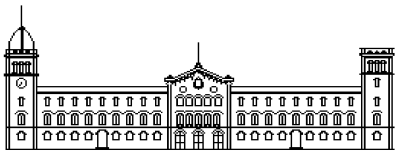


Trabajo fin de carrera

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

Facultad de Matemáticas
Universidad de Barcelona

RECONOCIMIENTO AUTOMÁTICO DE MODELOS DE AVIÓN EN SECUENCIAS DE VIDEO

Francesc Xavier Muñoz Romero

Directores: Sergio Escalera Guerrero
Carlos Gallardo García
Alberto Escudero Pardo
Realizado en: Departamento de Matemática Aplicada y Análisis. UB

Barcelona, 19 de Febrero de 2010

Agradecimientos

Éste apartado de agradecimientos quiero dedicarlo en primer lugar a mi tutor en este trabajo, Sergio Escalera Guerrero, cuyas explicaciones en cada reunión semanal han sido fundamentales para la realización del proyecto.

En segundo lugar agradecer a Marc Garcia i Ramis por su inestimable aportación en la elaboración de la parte de los métodos de extracción del avión, a Carlos Gallardo García por su esfuerzo en proporcionar el material necesario como vinculo vivo entre el proyecto y el aeropuerto, y a Alberto Escudero Pardo que sin su dedicación y constancia no hubiera sido posible que éste proyecto viera la luz.

Y finalmente a todos mis amigos y seres queridos que me han apoyado durante todos estos meses y que sin su guía me hubiera perdido.

Gracias a todos.

Resumen

Dentro de la nueva terminal del aeropuerto de Barcelona se llevan a cabo una media de 400 vuelos diarios, una vez en tierra, todos estos aviones deben ser dirigidos hacia distintos lugares dentro del aeropuerto.

En estos momentos este trabajo está siendo realizado por operadores a pie de pista, esto conlleva un retraso, ya que deben desplazarse e identificar cada modelo uno por uno y luego comprobar si existe una pista disponible para ese modelo.

Se propone que éste trabajo sea realizado automáticamente por un sistema inteligente capaz de distinguir los modelos y asignarles un lugar valido donde dirigirse.

Después de un estudio previo del terreno se ha visto factible implantar un sistema automatizado de reconocimiento de modelos de avión, aprovechando las instalaciones disponibles.

Resum

Dins de la nova terminal de l'aeroport de Barcelona es porten a terme una mitja de 400 vols diaris, una vegada a terra, tots aquests avions han de ser dirigits cap a diferents llocs dins l'aeroport.

En aquest moments aquest treball està sent realitzat per operadors a peu de pista, això comporta un retard, ja que han de desplaçar-se i identificar cada model un per un i després comprovar si existeix una pista disponible per quest model.

Es proposa que aquest treball sigui realitzat automàticament per un sistema intel·ligent capaç de distingir els models i assignar a cadascun un lloc vàlid on dirigir-se.

Després d'un estudi previ del terreny s'ha vist factible implantar un sistema automatitzat de reconeixement de models d'avió, aprofitant les instal·lacions disponibles.

Abstract

Inside the new terminal at Barcelona airport are being conducted an average of 400 flights per day, once on land, all these aircrafts should be directed to different places inside the airport.

Currently this work is being done by operators on the tarmac, this means a delay, as they must go and identify each model one by one and then see if there is a track available for that model.

It is proposed that this work will be done automatically by an intelligent system capable of distinguishing the models and assign them a valid place where to go.

After a preliminary study has been feasible to implement an automated pattern recognition of aircrafts using the facilities available.

Índice

1.- Introducción

Dentro de este apartado se realizará un estudio del problema planteado dando una introducción de todos los métodos a utilizar para resolverlo y la propuesta que mejor cumpla con todos los requisitos.

1.1.- Contextualización del Problema

La automatización dentro de las empresas del siglo 20 fue el gran empuje que hizo posible un nuevo desarrollo de la industria y un mayor confort en el puesto de trabajo del operador, pudiendo dedicar sus esfuerzos a otros trabajos. Para finales del siglo 20 hasta la actualidad, esta tendencia se ha extendido a nuestros hogares haciendo del trabajo diario algo en lo que no preocuparse. Un ejemplo lo vemos en las nuevas casas domóticas, donde hasta el mismo hecho de hacer la lista de la compra es un trabajo automatizado.

Con esta premisa, después de la inauguración de la nueva terminal del aeropuerto de Barcelona se ha querido seguir con este espíritu innovador y automatizar un trabajo que hoy día está siendo realizado por los operadores de forma manual. En estos momentos los operadores de pista en la nueva terminal del aeropuerto de Barcelona deben dirigir más de 400 aviones diarios. Cuando un avión acaba de aterrizar, el operador de pista al cargo debe aproximarse al avión y distinguir de qué modelo de avión se trata, revisar en un computador donde se puede dirigir y avisar al piloto del avión hacia donde debe dirigirse.

Si nos paramos a pensar en cómo automatizar todo este proceso, una primera aproximación sería analizar más detalladamente el proceso que realiza un operador humano para reconocer diferentes modelos de avión. El operador humano se vale de sus sentidos y experiencia para lograr realizar su trabajo, de manera que para alcanzar nuestro objetivo se quiere dotar de las mismas características a nuestro operador máquina.

Para lograrlo substituiremos el factor humano por un concepto computable para la máquina, descomponiendo el proceso humano en 3 pasos:

1.- Visualización del avión.

- Para este punto nos valdremos de una cámara para registrar los aviones a analizar.

2.- Observación de sus características.

- Para registrar las características de los diferentes modelos de aviones se utilizarán descriptores generados por distintos métodos. Los descriptores proveen de características entendibles por métodos automáticos computacionales, ya sea por ejemplo registrando el contorno del objeto como los puntos que lo componen.

3.- Con las características observadas discernir de qué modelo se trata.

- Para este punto se dotará al programa de una máquina de aprendizaje que sea capaz de aprender a distinguir los diferentes modelos a partir de los descriptores extraídos en el paso anterior. Como resultado se obtendrá un clasificador de rendimiento “equiparable” a la experiencia humana.

1.2.- Estado del arte

Hasta el momento no se conoce otra iniciativa parecida, lo que hace más atractivo conseguir solucionar el problema, ya que puede revolucionar el concepto de trabajo para miles de trabajadores alrededor del mundo dedicados a la gestión de los aviones a pie de pista.

Para lograr este objetivo a continuación se detallan los conceptos y las diferentes fórmulas matemáticas utilizadas:

1.2.1.- Descriptor

Un descriptor [1] almacena algunas características de una imagen, entendibles por métodos computacionales, ya sea por ejemplo registrando el contorno del objeto o los puntos que lo componen. Se podría decir que los descriptores son los encargados de describir el contenido de la imagen.

Existen diferentes métodos para describir una imagen, a continuación se explican los métodos utilizados en éste proyecto.

1.2.2.- Métodos descriptores

Para la creación de los descriptores se han escogido por su alto rendimiento los siguientes métodos:

- HOG (Histograms of oriented gradient)
- Zernike [2]
- Zoning

Una vez obtenido el descriptor de la imagen podemos conseguir el clasificador [3] que nos reconozca ese determinado modelo de avión.

1.2.3.- Clasificadores

Un clasificador [3] es un método para determinar la posible clase de un objeto desconocido, o evento sobre la base de un número de casos de cada una de las clases, conocido como el conjunto de entrenamiento.

El algoritmo utilizado para obtener los clasificadores en este proyecto será Adaboost [4], ya que ofrece adaptabilidad y baja susceptibilidad al sobreentrenamiento que otros algoritmos.

1.2.3.- Evaluación

En la etapa de evaluación se utilizarán los clasificadores extraídos mediante Adaboost. Para combinarlos se ha pensado en utilizar el método Voting que es una forma de extender los clasificadores binarios extraídos a multi-clase.

1.3.- Propuesta

Este proyecto se ha pensado como una primera aproximación hacia un sistema totalmente automatizado, donde a los aviones que van aterrizando se les indique donde deben dirigirse únicamente en el tiempo que tardan en aproximarse a la cámara que captura sus movimientos, como se muestra en la ilustración 1:

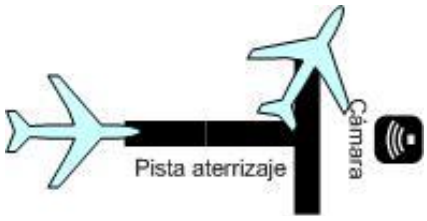


Ilustración 1 Diagrama técnico del aterrizaje y decisión automatizada.

Para más información sobre el aterrizaje de los aviones y las maniobras en la terminal 1 del Aeropuerto del Prat, se presenta material informativo en el apartado de Anexos, en el punto 6.2.

Para ello se propone crear un algoritmo capaz de reconocer y distinguir entre distintos modelos de avión e informar al operador sin necesidad de que éste abandone su puesto de trabajo. Para conseguir un programa ágil y eficaz se utilizará C++ como lenguaje de programación, ya que dispone de potentes librerías para la manipulación de imágenes, y un entrono elegante de programación.

Para la obtención del avión en una imagen encuadrada se han pensado en los siguientes métodos que extraen información de movimiento de una secuencia de imágenes:

- Eigen
- Wren
- Adaptative
- Grimson
- Zivkovik
- Prati
- Mean

Para la creación de los clasificadores de las imágenes de vídeo se ofrecen 3 métodos distintos:

- HOG
- Zernike
- Zoning

Al empezar el proceso de reconocimiento el usuario puede escoger entre los diferentes métodos.

Para el entrenamiento del clasificador se utilizara Adaboost [4], que es un algoritmo de aprendizaje formulado por Yoav Freund y Robert Schapier [5]. Este algoritmo es adaptable, en el sentido que la creación de los siguientes clasificadores se ajusta a favor de las instancias clasificadas erróneamente por los anteriores clasificadores. Es sensible al ruido en los datos y a los valores atípicos. Sin embargo, es menos susceptible al sobreentrenamiento que la mayoría de algoritmos.

En una primera versión estable del programa el usuario podrá ser capaz de saber qué modelo de avión se trata sin necesidad de desplazarse.

1.4.- Estructura de la memoria

A continuación se pasarán a desarrollar los siguientes puntos:

- Análisis: Este capítulo contiene todo el análisis del proyecto basándonos en sus requerimientos funcionales y planificación.
- Diseño: Este capítulo detalla la parte de diseño e implementación del proyecto.
- Resultados: En este capítulo se describen las pruebas y test que se han realizado, y se analizarán los resultados obtenidos.
- Conclusiones: Como indica su nombre se discuten las propuestas y los resultados obtenidos.
- Referencias: En este capítulo se puede encontrar referencias utilizadas durante el desarrollo de los puntos anteriores.
- Anexos: En este capítulo se puede encontrar material suplementario, como información del contenido del CD adjunto y descripción de la tecnología del aeropuerto de Barcelona.

2.- Metodología

A continuación se explica todo el análisis realizado para la ejecución del proyecto, tal como la distribución de tareas, los costes relacionados, y los casos de uso implicados en la aplicación.

2.1.- Análisis

2.1.1.- Planificación y costes

Dentro de este apartado se pasa a detallar los tiempos dedicados y el presupuesto necesario en cada etapa del proyecto.

Para realizar la planificación y los costes se ha definido un equipo de 3 analistas y 1 analista-programador con un plazo de 3 meses para realizar el proyecto.

En la tabla 1 que se muestra a continuación se puede observar el diagrama de Gantt previsto para el cumplimiento de los plazos.

Id.	Nombre de tarea	Comienzo	Fin	Duración
1	Lectura del problema	19/10/2009	21/10/2009	3d
2	Propuesta de soluciones (análisis)	22/10/2009	26/10/2009	3d
3	Diseño	27/10/2009	23/11/2009	20d
4	Implementación	02/11/2009	27/11/2009	20d
5	Escritura	30/11/2009	08/01/2010	30d
6	Test de pruebas	11/01/2010	15/01/2010	5d
7	Documentación	11/01/2010	29/01/2010	15d

Tabla 1 Diagrama de Gantt previsto

Se da como fecha de comienzo del proyecto el día 19 de Octubre de 2009, esto nos deja como fecha de entrega el día 19 de enero de 2010 con unos días de holgura para posibles retrasos en el transcurso del proyecto.

El primer paso a realizar es la lectura del problema, para ello se realizará una entrevista con un trabajador que opera a pie de pista para observar las condiciones de trabajo y poder realizar una lista de requerimientos necesarios a satisfacer por el programa. Esta rarea conlleva 3 días.

El segundo paso es analizar la lista de requerimientos obtenidos y proponer las mejores soluciones para el problema existente. Para ello se proponen 3 días de plazo.

En el tercer paso se realizará el análisis y diseño. Dentro de esta etapa se recogerá de una forma más formal la información extraída en los dos pasos anteriores, generando los diagramas de uso necesarios. Esta parte es muy importante dentro del proyecto, ya que el resto trabajo y esfuerzos se fundamenta en lo aquí acordado. Para esta etapa se proponen 20 días de plazo.

El cuarto paso es la implementación. Dentro de esta etapa se realizará tanto la escritura del pseudocódigo, como todos los diagramas necesarios para la mejor comprensión en la etapa de

escritura. Dado que la etapa anterior es incremental, se ha pensado en realizar en paralelo tanto diseño como implementación, para que vayan creciendo conjuntamente. Para esta etapa se proponen 20 días de plazo.

En el quinto paso se realizará la escritura de todo lo realizado en la etapa de implementación, para esta etapa se proponen 30 días de plazo.

En el sexto paso se realizarán los test de prueba y se analizarán los resultados para comprobar que se ajustan a los requerimientos recogidos en la primera etapa de lectura del problema. Si alguna parte no funciona como es debido se deberá volver al paso 2 donde se realizó la propuesta de soluciones y modificar todo lo necesario para ajustarlo. Para esta etapa se proponen 5 días, ampliables a 10 días en caso necesario.

En el séptimo y último paso se realizará la documentación donde se incluirá todo el trabajo realizado hasta el momento. Para esta última etapa se proponen 15 días.

Para calcular los costes relacionados no sólo se deben contar los gastos derivados de los analistas y programadores, sino también de las licencias de los programas utilizados. A continuación, en la tabla 2, se pasa a detallar los costes:

Concepto	Cantidad	Precio	IVA	Precio final
Licencia Matlab Student Version 2009	1	60,26 €	9,64 €	69.90 €
Licencia Microsoft Visual Studio 2008	1	713,15 €	114,1 €	827.25 €
Equipo informático	1	600 €	96 €	696 €
Horas de Trabajo	270	25 €/h	Exento	6750 €
		TOTAL	**219,74€**	**8343,15€**

Tabla 2 Relación de gastos del proyecto

2.1.2.- Análisis de métodos

El diseño del algoritmo para reconocer los diferentes modelos de aviones se ha querido orientar al mínimo gasto computacional, ya que trabajar con imágenes es muy costoso tanto en tiempo computacional como en recursos del sistema.

Para ello se ha escogido como lenguaje de programación C++, ya que provee un entorno rápido y dinámico, ideal para la gestión de grandes paquetes de datos con el mínimo coste. Otra razón para escoger C++ como lenguaje han sido las librerías ofrecidas por la comunidad OpenCV [5], que han sido desarrolladas bajo C++. Estas librerías proveen las funciones necesarias para la captura, análisis, modificación y tratamiento de imágenes, entre otras funciones.

Antes de adentrarnos en los pormenores del programa de reconocimiento se debe proporcionar al programa una imagen del avión obviando todas las partes irrelevantes de la imagen obtenida. Esto quiere decir que si en la imagen extraída del vídeo en el segundo indicado contiene otros elementos fuera del avión, éstos no harán otra cosa que entorpecer el aprendizaje del programa ya que son partes no características del modelo de avión. Este proceso también se denomina segmentación. Para solucionarlo se ha pensado en distintos métodos para encontrar el avión dentro de la imagen y ser capaces de extraerlo dando como resultado una nueva imagen donde se muestra únicamente el avión.

Para conseguirlo nos hemos valido de los siguientes métodos:

- Eigen [9]
- Wren [9]
- Adaptative [9]
- Grimson [9]
- Zivkovik [9]
- Prati [9]
- Mean [9]

Eigen Background-substraction

Eigen proviene del alemán y significa propio. Dado que un frame contiene su información de una manera matricial, en este método nos hemos servido de los eigenvectors y sus eigenvalues que se van obteniendo de cada paso. Un eigenvector (o vector propio) es aquel que no se ve afectado por las transformaciones lineales y que en cualquier caso no varía su dirección todo y ser multiplicado por un escalar. Al mismo tiempo, un eigenvalue (o valor propio) será aquel escalar por el cual se multiplica el eigenvector asociado. Entonces, este entorno es conocido como un eigenspace.

Una imagen digital [6] es una matriz con una información determinada. Entonces iremos obteniendo frame a frame esta matriz donde se delimita un eigenspace. Cabe señalar que la mayor parte del peso de una imagen está contenida en pocos eigenvalues.

Al iniciar el método, en Eigen.cpp se establecen los parámetros comunes a todas las otras funciones y además declaramos HistorySize, que nos indica con cuantos frames estableceremos la imagen de fondo. EnbeddedDim indica con cuanta información nos quedamos de los frames, por defecto la contenida en los 20 eigenvalues de más peso.

Finalmente, un vez que obtenemos la imagen de background iremos proyectando los nuevos frames sobre ésta. Pixel a pixel comprobaremos la distancia euclidiana entre la nueva imagen y la guardaremos como imagen de fondo.

Método de AdaptiveMedian

Esta funcionalidad aprovecha los cambios en la luminosidad y las múltiples superficies que aparecen en un pixel para determinar cuáles corresponden al fondo y cuáles no. A cada iteración los valores Gaussianos adaptados serán evaluados de forma heurística y los píxeles que no coinciden con los del fondo "Gaussiano" serán clasificados como en primer plano.

Por otro lado, si de todos estos cambios recordamos algunos de los últimos nos permitirá ir obteniendo una media, y por lo tanto, un historial para saber si los cambios que van sufriendo los píxeles son significativos y por tanto si merecen o no ser background.

En el código, *TFC_Base->Adaptive.cpp*, podemos observar como inicializamos la varialbe específica SamplingRate donde se nos indica cuantos frames se utilizan para ir renovando esta media.. Mientras que en LearningFrames serán el número de frames que utilizaremos para conseguir el primer background. Por defecto se utiliza los primeros 30 frames para el fondo inicial que iremos modificando cada 7 en caso de que se considere oportuno según la algorítmica ya anunciada.

Uno de los pasos a seguir para conseguir un algoritmo inteligente capaz de distinguir entre diferentes modelos de avión es indicarle que es un avión, para ello nos hemos valido de 3 métodos descriptores diferentes:

- HOG

- Zernike

- Zoning

HOG (Histograms of oriented gradient)

Cada descriptor obtenido por esta técnica se define en función de la magnitud y orientación del gradiente de los bordes de la imagen.

Zernike

Los polinomios de Zernike se propusieron por primera vez en 1934 por Zernike. Su formulación por momentos parece ser uno de los más populares, superando las alternativas existentes (en términos de capacidad de resistencia de ruido, la redundancia de información y la capacidad de reconstrucción). La formulación pseudo-Zernike propuesta por Bhatia y Wolf [6] fue mejorar aún más estas características.

Zoning

Aunque se trata de una técnica en su concepto sencilla es eficaz en el caso que nos ocupa. Una vez transformada la imagen para dejar únicamente los bordes de ésta, se trata de contar los puntos que componen estos bordes. La imagen debe componerse únicamente de puntos.

Ejecutar estos métodos da como resultado un descriptor de la imagen, que indica de qué forma está compuesta la imagen. Cada método describe la imagen de distinta forma. Esto nos es muy útil a la hora de indicarle al programa como es un avión para que posteriormente pueda distinguirlo de otro modelo.

El siguiente paso es hacer que el programa aprenda cómo es un avión de un determinado modelo, para ello se ha utilizado el algoritmo Adaboost [4], una máquina de aprendizaje que actúa combinando gran cantidad de clasificadores sencillos para conseguir un elevado grado de precisión en problemas de clasificación. Un clasificador [3] es un método para determinar la posible clase de un objeto desconocido, o evento, sobre la base de un número de casos de cada una de las clases, conocido como el conjunto de entrenamiento.

A continuación se puede observar en la ilustración 2 un diagrama explicativo del proceso a seguir para la obtención de los clasificadores mediante Adaboost:

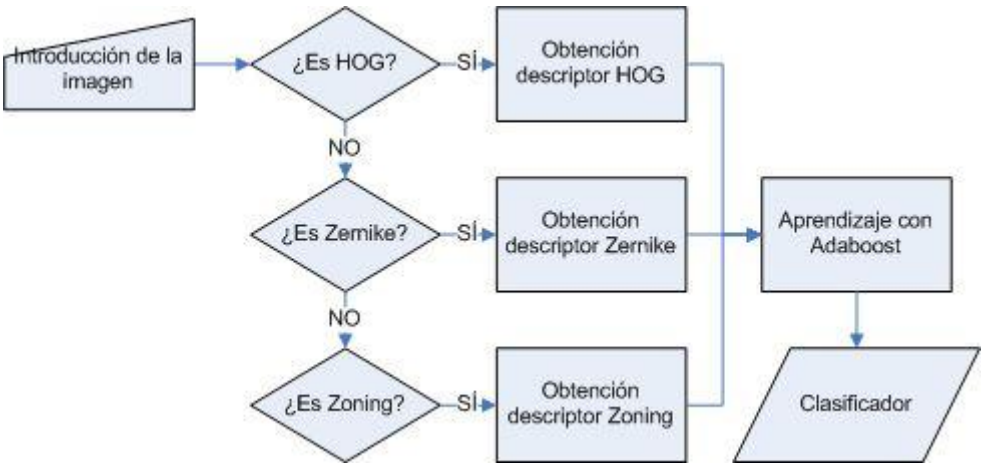


Ilustración 2 Diagrama de flujo para la obtención de los clasificadores

Una vez conseguidos los clasificadores se necesita crear un algoritmo capaz de diferenciar entre varios modelos de avión diferentes. Esto es así porque el proceso de aprendizaje es binario, es decir, que se comparan los modelos de avión uno contra otro, esto seria suficiente si nuestro escenario unicamente contara con 2 modelos de avión, pero éste no es el caso. Para conseguir un escenario multi-clase se utlizará el método Voting que consiste en comparar el descriptor de entrada con cada uno de los clasificadores de cada modelo de avión almacenando el resultado obtenido. El modelo elegido será aquel que mayores aciertos haya conseguido.

2.1.3.- Análisis de requerimientos funcionales

En este apartado se pasará a detallar los módulos a desarrollar y una explicación de cada uno de ellos.

En la ilustración 3 que podemos ver a continuación se puede observar de modo general las diferentes opciones que tienen los distintos usuarios para interactuar con el programa:

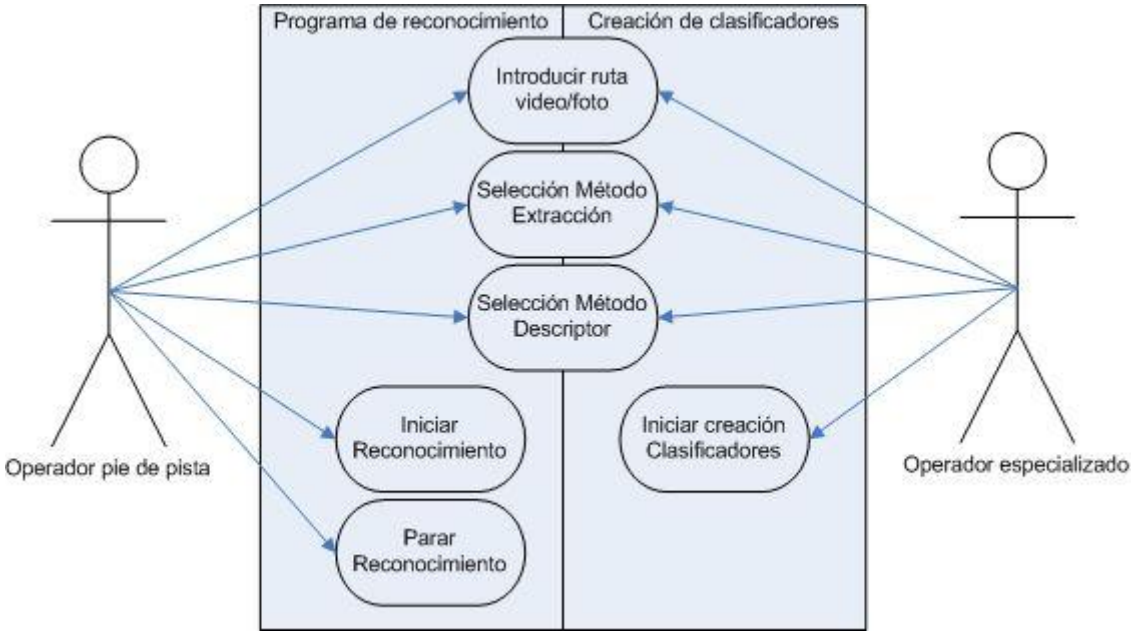


Ilustración 3 Diagrama casos de uso visión general

Como podemos observar la solución está compuesta por dos programas, uno para uso de los usuarios a pie de pista, encargados de dirigir a los diferentes modelos de avión, cuyo fin es el de informar al operador del modelo correspondiente de avión. Y un segundo programa que se encarga de crear los clasificadores. Éste segundo programa se utilizara en una primera fase para generar los clasificadores de los distintos modelos y en etapas posteriores cuando se quiera mejorar éstos clasificadores.

A continuación se pasa a desarrollar paso a paso cada uno de los casos de uso mostrados en la ilustración anterior, cabe indicar que los casos de uso compartidos por los dos programas serán explicados una vez siendo totalmente representativos para los dos programas:

Descripción casos de uso

Los dos programas propuestos como solución tienen como partes comunes 3 casos de uso que se pasarán a describir a continuación bajo el titulo “Casos Comunes”. Los casos de uso diferentes para cada programa se desarrollaran bajo el titulo propio del programa:

1.-Casos Comunes

1.1.- Introducción ruta Vídeo/Foto

- El usuario podrá introducir la ruta donde se encuentra el vídeo o la foto que desea que sea descrita por el programa

1.1.1.- Flujo

- Mediante un botón en la interfaz gráfica, el usuario podrá ser capaz de buscar el archivo deseado con el mismo explorador de Windows.

1.1.2.- Precondiciones

- Queda como responsabilidad del usuario tener en el formato de vídeo y/o foto adecuado el archivo a analizar.

1.1.3.- Postcondiciones

- En pantalla aparecerá la ruta introducida por el usuario.

1.2.- Selección del Método de Extracción

- El programa deberá dar la opción al usuario de escoger entre diferentes métodos de encuadre de un avión a partir de un vídeo.

1.2.1.- Flujo

- Mediante una lista desplegable en la interfaz gráfica, el usuario podrá ser capaz de elegir entre los diferentes métodos de extracción.

1.2.2.- Precondiciones

- El usuario deberá haber escogido una ruta donde se encuentre un archivo válido.

1.2.3.- Postcondiciones

- Como resultado se obtendrá una imagen encuadrada obtenida a partir de un frame del vídeo introducido anteriormente.

1.3.- Selección del Método Descriptor

- El programa deberá dar la opción al usuario de escoger entre diferentes métodos de describir el contenido de la imagen.

1.3.1.- Flujo

- Mediante una lista desplegable en la interfaz gráfica, el usuario deberá ser capaz de elegir entre los diferentes métodos capaces de describir la imagen.

1.3.2.- Precondiciones

- Para que el descriptor resultante sea lo más eficaz posible se deberá pasar primero por el método extractor del avión para evitar objetos no relacionados.

1.3.2.- Postcondiciones

- Como resultado se obtiene un archivo con extensión ".dat" donde se encuentra almacenado el descriptor obtenido. El título del archivo resultante proviene del nombre del vídeo introducido por el usuario, seguido por los valores introducidos para las subimágenes. Un archivo nombrado por ejemplo "MODXXX_HOG_H4W6.dat" se ha construido utilizando el nombre del vídeo introducido que se llamaba "MODXXX", el método descriptor utilizado, en este caso HOG y los valores dados para recortar la imagen en trozos más pequeños, que han sido 4 para el eje de las "y" y 6 para el eje de las "x".

A partir de este punto los dos programas propuestos difieren el uno del otro para lograr resultados diferentes. Se empezará detallando el programa creador de clasificadores ya que el programa de reconocimiento de modelos de avión necesita de estos clasificadores para funcionar.

2.-Creación de clasificadores

- Dentro de esta fase del programa se crean los clasificadores, para ello se utilizara el algoritmo Adaboost [4] descrito anteriormente. Esta parte de la solución a diferencia del resto del programa está programada bajo Matlab.

2.1.- Iniciar creación Clasificadores

- Mediante el programa Matlab se ejecutarán las funciones generadas para la creación de los clasificadores.

2.1.1.- Flujo

- Desde la ventana de comandos de Matlab introducimos "Main();".

- La función "Main" carga los descriptores para cada modelo de avión y los compara uno contra otros dando como resultado los clasificadores.

2.1.2.- Precondiciones

- Es necesario haber creado los descriptores mediante alguno de los métodos anteriormente descritos.

- Queda como responsabilidad del usuario asignar como carpeta de trabajo dentro del programa Matlab, el mismo donde se encuentran los descriptores.

2.1.3.- Postcondiciones

- En el mismo directorio de trabajo se crearán los clasificadores entrenados mediante Adaboost.

3.-Programa de reconocimiento

- A partir de esta parte del programa empieza el reconocimiento del modelo de avión introducido por el usuario. Para ello se hace requisito imprescindible que se hayan creado con anterioridad los descriptores para todos los modelos de avión.

3.1.- Iniciar Reconocimiento

- En este caso de uso se engloba todo el procedimiento que conlleva el reconocimiento.

3.1.1.- Flujo

- El usurario dispondrá de un botón en la interfaz gráfica donde podrá indicar que el proceso de reconocimiento se inicie.

3.1.2.- Precondiciones

- Para que el proceso de inicie correctamente será indispensable que el usuario haya introducido la ruta donde se encuentra el archivo multimedia, el método de extracción de la imagen, el segundo dentro del vídeo que desea que sea examinado, el método descriptor, en cuantos recuadros quiere seccionar la imagen en el caso de métodos como HOG o Zoning y los momentos requeridos en el caso de tratarse de Zernike. Deben haberse creado con anterioridad los archivos clasificadores con extensión ".boost" dentro de la carpeta "boost".

3.1.3.- Postcondiciones

- Una vez finalizado el proceso se muestra por pantalla el modelo de avión que el sistema cree haber reconocido. Por el proceso se habrán creado en la carpeta "descriptores", los descriptores de las imágenes analizadas.

3.2.- Parar Reconocimiento

- En este caso de uso se analiza todo lo ocurrido cuando el usuario decide parar el proceso de reconocimiento del modelo de un avión.

3.2.1.- Flujo

- El usuario dispondrá de un botón en la interfaz gráfica donde podrá parar el proceso de reconocimiento una vez éste está iniciado.

3.2.2.- Precondiciones

- Para que al usuario se le permita parar el proceso será imprescindible que se haya iniciado el proceso de reconocimiento.

3.2.3.- Postcondiciones

- Se mostrará un mensaje indicando que el usuario ha terminado con el proceso y se procederá a inicializar todos los procesos en marcha, dejando la interfaz lista para un nuevo uso.

2.2.- Diseño

Como parte del diseño del software se pasará a explicar la arquitectura elegida. Se ha querido diferenciar entre la parte de interfaz y la parte de cálculo y evaluación. Para ello se creará una librería "TFC_Base" donde se encontrarán todas las funciones necesarias para la descripción de las imágenes, la creación de clasificadores mediante Adaboost [4] y la evaluación en un entorno tanto de testeo como para un usuario final. Para hacer la librería más accesible se creará una clase denomina "Controlador" que ofrecerá todas las funciones que la librería es capaz de ofrecer. También se ha querido diferenciar entre la parte de evaluación y aprendizaje del programa. Para ello se crearán dos programas uno donde se evalúan las imágenes obtenidas y otro donde se crean los clasificadores necesarios para el aprendizaje del programa.

El software resultante está definido por los siguientes pseudocódigos:

2.2.1.- Pseudocódigos

Pseudocódigo 1: Interfaz de usuario

El programa no finalizará hasta que el usuario pare el proceso inicializado o indique que quiere salir del programa

1 Búsqueda y carga de la imagen o vídeo a analizar

2 Carga del vídeo o imagen

3 Selección del método de detección de una imagen en el vídeo (si se trata de un vídeo)

4 Selección del método del descriptor con sus características

5 Iniciar Proceso

Pseudocódigo 2: Proceso inicializado (línea 5 del pseudocódigo 1)

En esta parte se debe diferenciar entre parte de aprendizaje y parte de evaluación.

Pseudocódigo 2.1: Parte de aprendizaje

1 Se obtiene el descriptor a partir de la imagen introducida mediante un método escogido por el usuario

2 Se utiliza la herramienta Matlab para la creación de los clasificadores

3 Se guarda el clasificador en formato de archivo .boost

4 Final del proceso

Pseudocódigo 2.2: Parte de evaluación

1 Se inicia la reproducción del vídeo en la interfaz del usuario

2 Se obtiene la imagen del vídeo (si se trata de un vídeo) mediante un método escogido por el usuario

3 Se obtiene el descriptor a partir de la imagen anterior mediante un método escogido por el usuario

4 Se recupera el descriptor creado del sistema de archivos en formato .dat

5 Se evalúa el descriptor a partir de los clasificadores creados anteriormente .boost y se retorna el nombre del modelo coincidente

6 Se carga la imagen del modelo resultante

Los pseudocódigos descritos a continuación pertenecen a la parte de la librería creada para el programa para realizar los cálculos y evaluaciones.

Pseudocódigo 2.3: Obtención del descriptor

Esta parte del código la comparten ambos programas y se corresponden con el pseudocódigo 2.1 línea 1 y el pseudocódigo 2.2 línea 3.

1 Obertura del archivo con nombre y formato adecuado en modo escritura

2 Obtención del descriptor

3 Si el proceso ha finalizado correctamente se guarda en el archivo abierto anteriormente

4 Se cierra el archivo

Pseudocódigo 2.3.1: Métodos de obtención del descriptor (línea 2 pseudocódigo 2.3)

A continuación se explica mediante pseudocódigo las tres posibles opciones para obtener el descriptor.

- HOG

1 Se trocea la imagen (se estudia cada porción por separado)
2 Se analizan todos los trozos uno por uno
  2.1 Se aplica HOG al trozo
  2.2 Se introduce el resultado en una estructura
3 Si se han analizado todos los trozos salimos del loop
4 Si todo ha salido bien se retorna OK

- Zernike

1 Se aplica Zernike a la imagen
2 Se introduce el resultado en una estructura
3 Si todo ha salido bien se retorna OK

- Zoning

1 Se trocea la imagen (se estudia cada porción por separado)
2 Se analizan todos los trozos uno por uno
2.1 Se aplica Zoning al trozo
2.2 Se introduce el resultado en una estructura
3 Si se han analizado todos los trozos salimos del loop
4 Si todo ha salido bien se retorna OK

Los pseudocódigos descritos a continuación pertenecen únicamente al programa de evaluación.

Pseudocódigo 2.4: Recuperación del descriptor (Pseudocódigo 2.2 línea 4)

1 Obertura del archivo con nombre y formato adecuado en modo lectura

2 Lectura del descriptor (Se recorre todo el archivo)

3 Se cierra el archivo

4 Si el proceso ha finalizado correctamente se retorna el descriptor

Pseudocódigo 2.5: Evaluación (Pseudocódigo 2.2 línea 5)

1 Se genera la matriz con todos los clasificadores existentes

2 Se recorre la matriz hasta haber recorrido todas las celdas

2.1 Se evalúa el descriptor con los datos del clasificador contenido en cada celda

2.2 Se guarda el resultado en una estructura

3 Una vez obtenido el resultado de la comparación se busca el nombre del modelo coincidente.

4 Se retorna el nombre del modelo coincidente

Algoritmo de aprendizaje del clasificador

A continuación se pasa a explicar más detalladamente el algoritmo de aprendizaje Adaboost escogido para este proyecto.

• Dadas las imágenes de ejemplo $(x_1, y_1), \ldots . (x_n, y_n)$ donde $y_1 = 0,1$ para los ejemplos negativos y positivos respectivamente.

• Inicializamos los pesos $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ para $y_1 = 0,1$ respectivamente, Donde *m* y *l* son el número de negativos y positivos respectivamente.

• Para t=1,…,T:

1. Normalizar los pesos, $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$

De manera que w_t es una distribución de probabilidad.

2. Para cada característica, *j*, entrenar un clasificador *hj* que está restringido a usar una sola característica. El error es evaluado con respecto a

$$w_t, e_j = \sum_i w_i \, |h_j(x_i) - y_i \,|$$

3. Escoger el clasificador, $h_j j$, con el menor error e_t

4. Actualizar los pesos: $w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$

Donde $e_i = 0$ si el ejemplo x, se ha clasificado correctamente $e_i = 0$, en caso contrario, y $\beta_t = \frac{e_i}{1-e_i}$

• El clasificador fuerte final es:

$$h(x) = \begin{cases} 1 \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 \; En \; caso \; contrario \end{cases}$$

Donde $\alpha_t = \log \frac{1}{\beta_t}$

Una vez obtenemos un clasificador, entrenamos los siguientes teniendo en cuenta los casos negativos no descartados por los clasificadores anteriores.

2.2.2.-Estructuras

Se han utilizado estructuras para el manejo de información procedente de archivos guardados en el sistema ya que es un modo fácil y rápido de manejar la información sin necesidad de complicar la lógica y estructura del programa.

```
typedef struct NomFichero{
char nombre[260];
NomFichero* sig;
}NomFichero;
```

Ésta estructura es utilizada por el programa para gestionar los archivos contenidos en las carpetas utilizadas durante la ejecución del programa. La variable *nombre* puede contener un nombre de hasta 260 caracteres, y la variable *sig* es un puntero que apunta al siguiente archivo. Esta estructura se utiliza por ejemplo cuando queremos guardar los archivos contenidos dentro de una carpeta por ejemplo.

```
typedef struct MatrixIA{
char *nombre;
unsigned int codigo;
float* alpha;
float* feature;
float* p;
float* thr;
MatrixIA* sig;
}MatrixIA;
```

Ésta estructura es utilizada por el programa para crear de forma dinámica una matriz de estructura lineal donde se guardan todas las característica del clasificador. Toda la información se almacena en tiempo de ejecución por éste motivo todas las variables deben ser punteros. En la variable *nombre* se almacena el nombre del avión evaluado, las variables *alpha, feature, p, thr* son las variables obtenidas como clasificadores y se utilizan para evaluar. La misión de la variable *código* es la de dotar a los distintos clasificadores de un mismo modelo de un código inequívoco que lo represente a la hora de evaluar. La variable *sig* apunta la dirección de donde se encuentra el siguiente clasificador.

2.2.3.-Diagramas

Diagramas para la interfaz grafica

Los diagramas de secuencias que se encuentran dentro de esta sección, muestran el funcionamiento interno del software.

En la ilustración 4 podemos observar el diagrama de clases de la interfaz del programa:

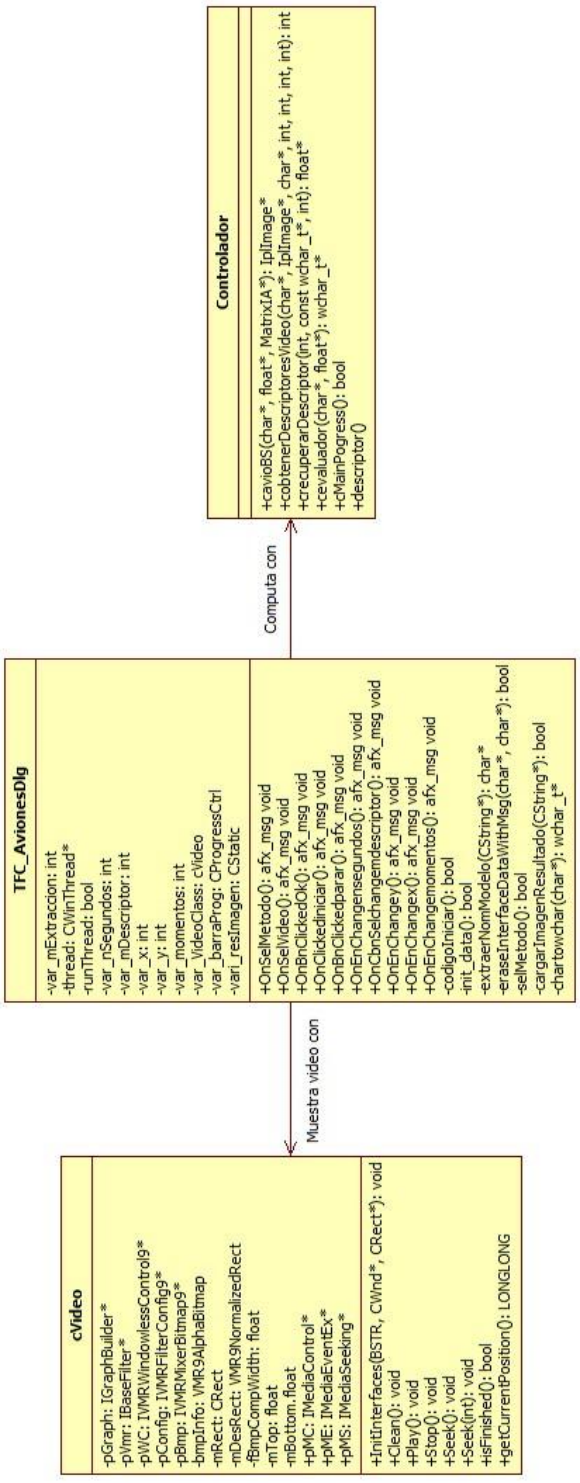


Ilustración 4 Diagrama de clases de la Interfaz de usuario

La interfaz del programa se compone principalmente de 3 clases para funcionar correctamente. La clase principal y que actúa a modo de columna vertebral es la clase *TFC_AvionesDlg.* En ella podemos encontrar las funciones necesarias para gestionar todas las funcionalidades de la interfaz así como todas las funciones precursoras encargadas de iniciar los procesos.

La clase *cVídeo* es la encargada de gestionar todo lo referente a la gestión del vídeo que se va mostrando mientras se extrae la imagen del vídeo y se evalúa.

La clase *Controlador* contiene las funcionalidades principales de la librería *TFC_Base* y sirve de nexo de unión entre la interfaz y la librería.

Con tal de comprender un poco mejor el funcionamiento de esta última clase se muestra a continuación en la ilustración 5 un diagrama de componentes con la finalidad de mostrar de una forma clara el comportamiento entre librerías dentro del programa.

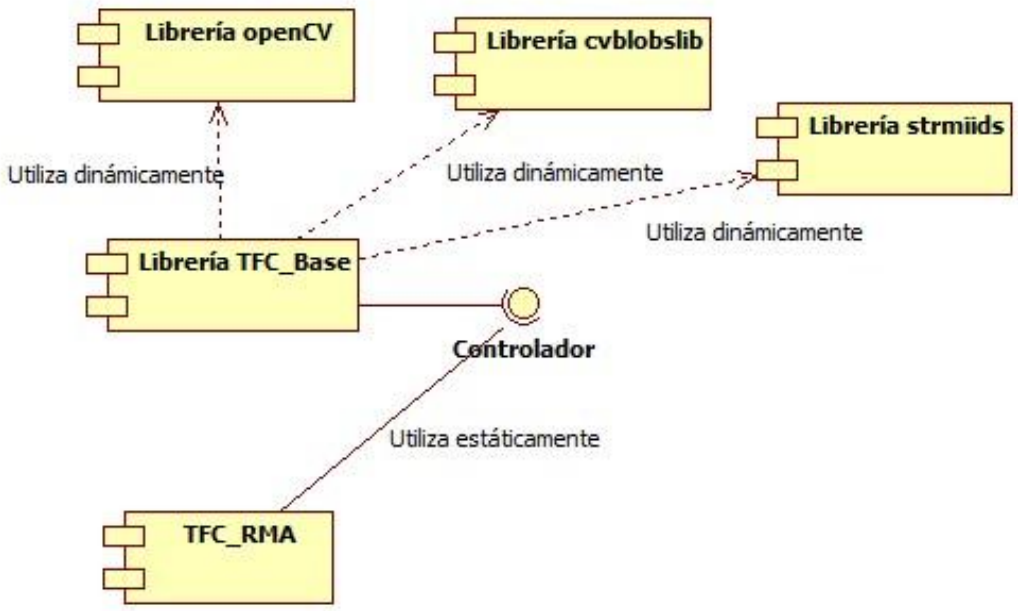


Ilustración 5 Diagrama de componentes

Como se puede observar todo el proyecto *TFC_RMA* hace servir de modo estático y haciéndose valer de la clase Controlador la librería *TFC_Base*, donde se encuentran las todas las funciones encargadas de realizar los cálculos dentro del proyecto. A su vez, la librería *TFC_Base*, utiliza dinámicamente las funciones de la librería OpenCV.

La ilustración 6 muestra el diagrama de secuencia de la parte de interfaz del programa. Cabe destacar que se describe el escenario en el caso de que no se produzca ningún error.

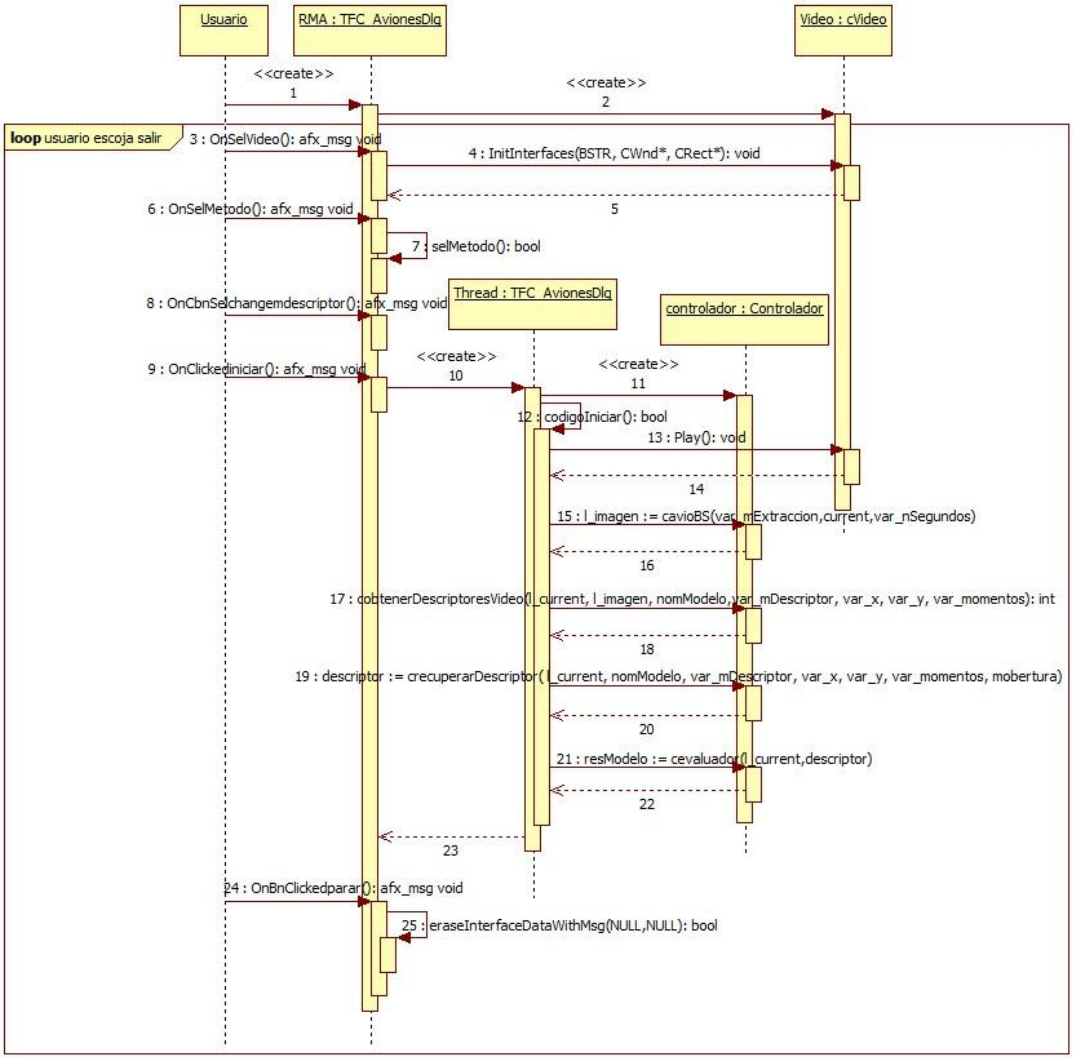


Ilustración 6 Diagrama de secuencia interfaz de usuario

En cuanto el usuario inicia el programa se ejecutan los pasos 1 y 2 donde se crean las clases *TFC_AvionesDlg* y ésta a su vez crea la clase *cVídeo*, en este momento el programa está listo para recibir las indicaciones técnicas disponibles para el usuario. Al tratarse de una interfaz grafica el usuario puede ir introduciendo los datos correspondientes a los pasos 3, 6 y 8 en distinto orden al propuesto en el diagrama.

Una vez que el usuario inicia la fase de evaluación se crea un thread de tipo *TFC_AvionesDlg* encargado de evaluar de forma independiente la imagen. Se pasa de forma secuencial por los pasos 15, 17, 19 y 21 correspondientes a obtención de la imagen, el proceso de descripción de ésta y posterior recuperación del descriptor del sistema de ficheros y la evaluación respectivamente.
La creación de un thread es necesaria para garantizar que todo el programa no quede bloqueado y permitir que el usuario interrumpa de forma manual la ejecución del programa.

En la ilustración 7 que se muestra a continuación se puede observar el diagrama de secuencia correspondiente a la parte donde el usuario quiere parar la ejecución del programa.

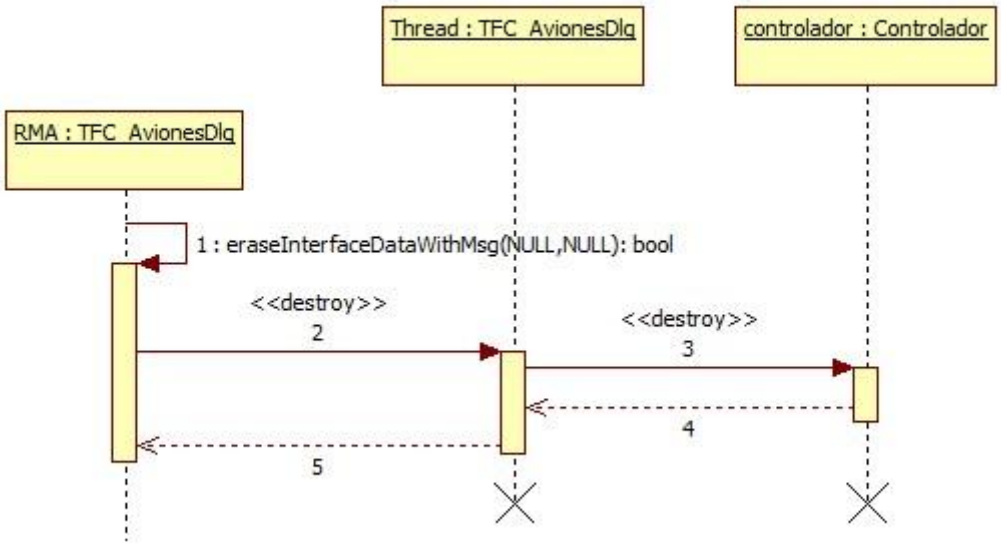


Ilustración 7 Diagrama de secuencia cuando el usuario para el proceso ya iniciado

Éste diagrama va vinculado al punto 25 de la Ilustración 6.

Diagramas de la librería estática

Dentro de este apartado podemos encontrar todos los diagramas y explicaciones referentes a la parte de la librería encargada de los cálculos, tanto de descripción como de evaluación.

A continuación se observa en la ilustración 8 el diagrama de secuencia de la parte de obtención de la imagen del avión segmentada, va vinculada al punto 15 de la Ilustración 6.

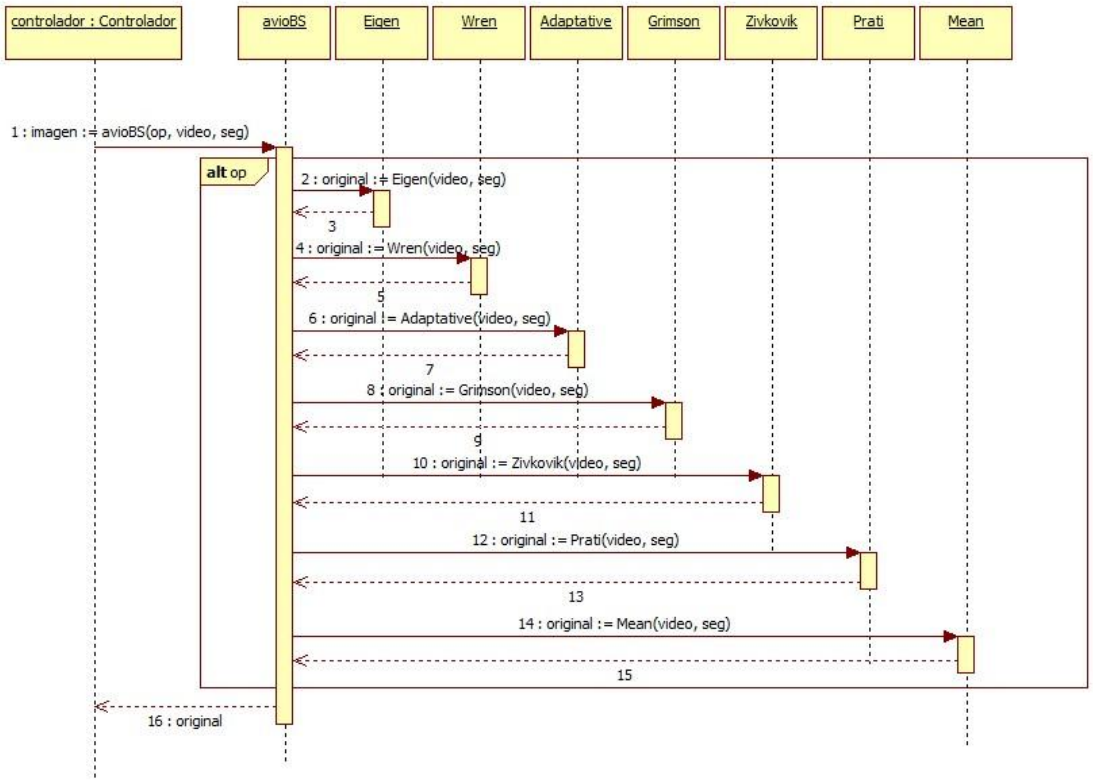


Ilustración 8 Diagrama de secuencia de la obtención de la imagen a partir de un vídeo

Como podemos observar en la ilustración 8, según la variable *op* se escoge que tipo de extracción se utilizará para extraer la imagen del vídeo. Se retorna la variable *original* que corresponde a la imagen la extraída y encuadrada.

En la ilustración 9 se pasará a analizar la parte donde se obtiene el descriptor de la imagen, este diagrama va vinculado al punto 17 de la Ilustración 6.

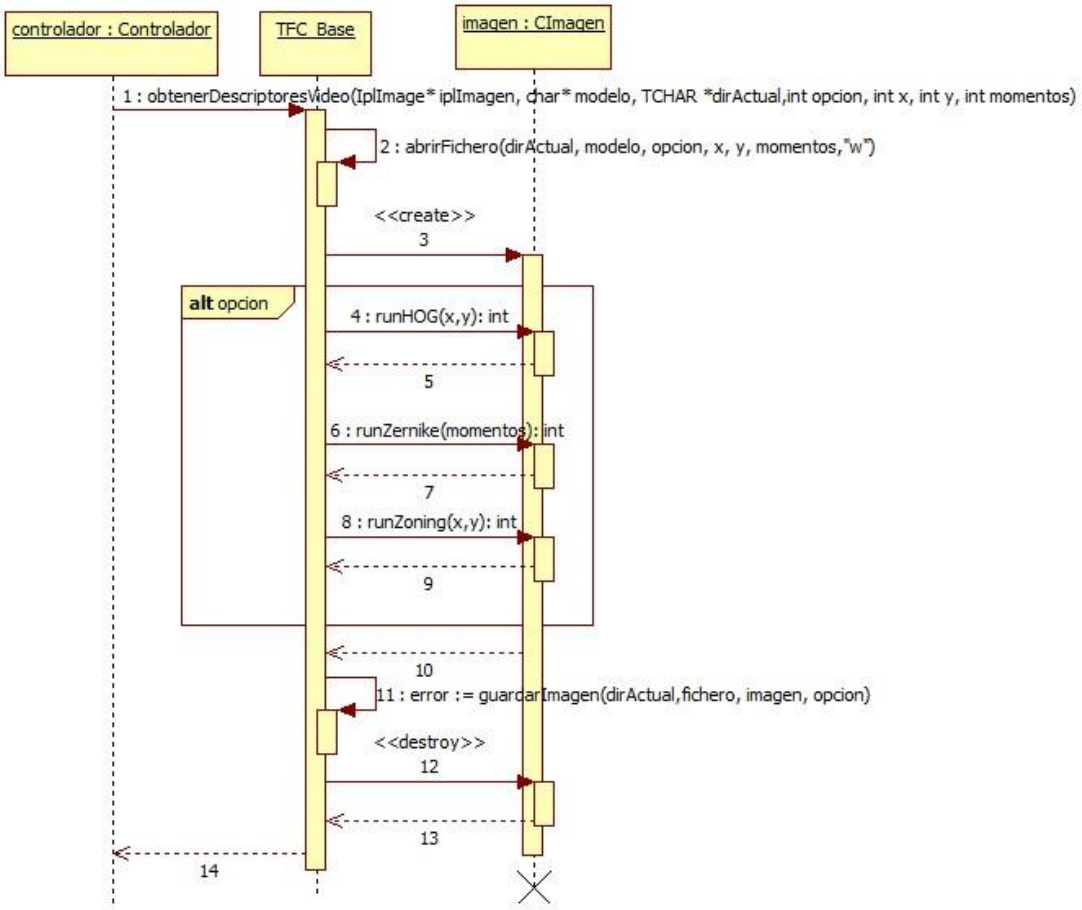


Ilustración 9 Diagrama de secuencia obtención del descriptor de la imagen

En el diagrama anterior se puede apreciar el proceso a seguir para la extracción del descriptor de la imagen. La variable *opcion* fija el método a utilizar.

La ilustración 10 está vinculada al punto 19 de la Ilustración 6. Se muestra el diagrama de secuencia de la fase de recuperación de la información del descriptor guardada anteriormente en el sistema de archivos.

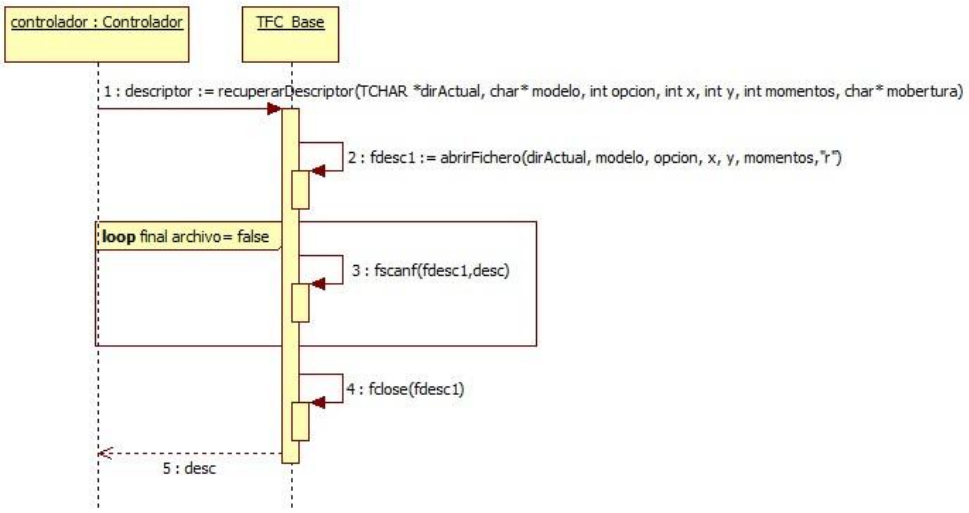


Ilustración 10 Diagrama de secuencia Recuperación datos descriptor

La ilustración siguiente, vinculada al punto 21 de la Ilustración 6, es un diagrama de secuencia donde se pueden observar los pasos a seguir para evaluar una imagen pasada anteriormente por parámetros.

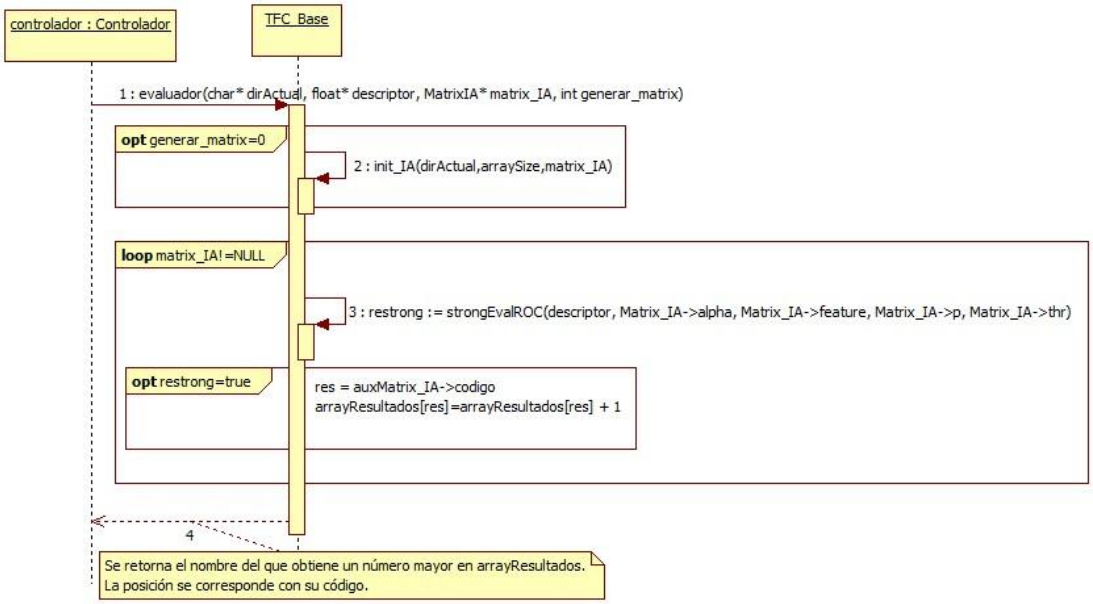


Ilustración 11 Diagrama de secuencia Evaluación de una imagen

3.-Resultados

En este apartado se expondrán los datos de los métodos, los experimentos a realizar para evaluar el software, y los resultados obtenidos.

3.1.- Datos

En esta sección se describen los datos que son necesarios para poder realizar los procesos de entrenamiento y evaluación (reconocimiento del modelo de avión).

3.1.1.- Descripción

En el apartado de descripción se utilizará como material de descripción las fotografías extraídas a pie de pista simulando la altura y posición de la cámara que actualmente está operativa. Éstas se encuentran dentro de la carpeta "fotos" en la carpeta raíz, clasificadas en diferentes carpetas según el modelo de avión a que se corresponda. A modo de ejemplo, en la Ilustración 12 se muestra el recorrido realizado por el avión durante la ejecución del vídeo.

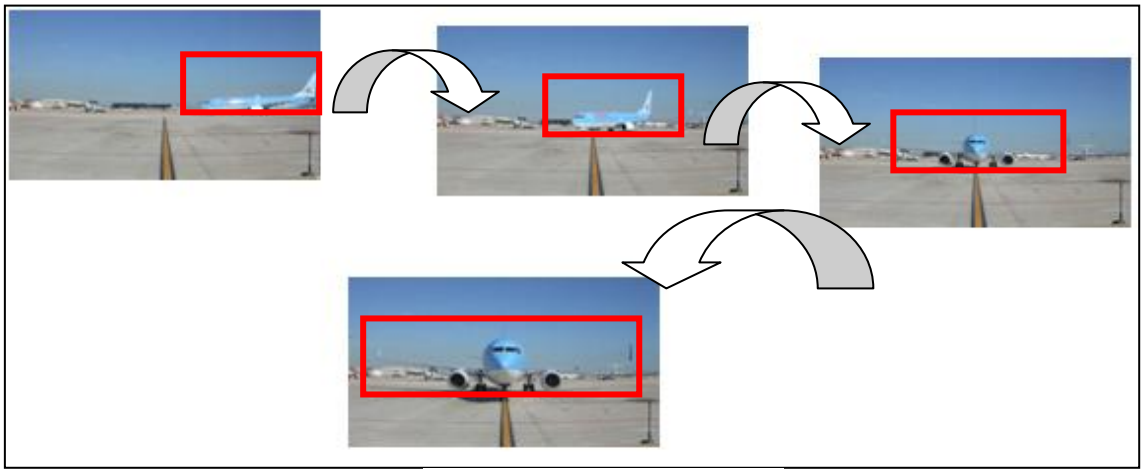


Ilustración 12 Recorrido del avión

3.1.2.- Entrenamiento

En el apartado de entrenamiento se utilizará como material de entrenamiento los archivos generados en la parte de descripción que se encuentran dentro de la carpeta "descriptores" en la carpeta raíz.

Cabe destacar que la calidad de los clasificadores [3] viene directamente relacionada con la cantidad de fotografías que se tengan disponibles para cada modelo en concreto, es decir, que un modelo con menos fotografías disponibles, por tanto menos información que lo detalle, le será a la máquina de aprendizaje más dificultosa la labor de reconocer si la imagen de entrada se corresponde con ése modelo en concreto.

3.1.3.- Evaluación

Para el apartado de evaluación se utilizarán fotografías y videos extraídos a pie de pista simulando la altura y posición de la cámara que actualmente está operativa. Es fundamental para este apartado, que tanto las fotografías como los videos, no hayan sido utilizados para entrenar a la máquina de aprendizaje ya que esto supondría enseñar una fotografía o parte de un vídeo que la maquina ya ha aprendido anteriormente e invalidaría cualquier resultado obtenido.

3.2.- Experimentos

En este apartado se muestran los experimentos realizados y los resultados obtenidos. Como valores de entrada del experimento se han tomado los siguientes:

- Fotografías: Tantas fotografías como modelos vamos a evaluar, en este experimento serán 4 modelos. Es fundamental que las imágenes de entrada no hayan servido para entrenar al programa.

- Vídeo: Se utilizará un vídeo por cada modelo del experimento para tratar de demostrar la eficacia del programa. Es fundamental como en el punto anterior que las imágenes extraídas del vídeo o el video en sí, no haya servido para entrenar al programa.

- Método de extracción: Se utilizarán todos los métodos de extracción disponibles.

- Segundos en el vídeo: Se capturará la imagen del video 2 segundos antes de su finalización con la intención de simular un posible funcionamiento real.

- Método descriptor: Se utilizarán los tres métodos descriptores disponibles para evaluar su fiabilidad.

- Número de subimagenes: En el caso de los métodos descriptores HOG y Zoning se escogerá como medida estándar 24 subimagenes por imagen, por lo tanto se partirá la imagen de entrada en 6 imágenes para el eje "y" y 4 imágenes para el eje "x".

- Número de momentos: En el caso del método descriptor Zernike se escogerá como medida estándar 9 momentos.

3.2.1.- Describiendo imágenes

Al acabar el procedimiento de descripción de una imagen se obtiene un archivo con formato *.dat* donde podremos encontrar los descriptores de la imagen. El número de descriptores vendrá dado por el numero de subimágenes que se hayan ordenado realizar, en este experimento se tratarán 24 subimágenes por imagen, por lo tanto podremos encontrar hasta 24 descriptores dentro de cada archivo *.dat*.

Para más información de cómo realizar este procedimiento puede leer en el apartado de anexos el punto *6.1.3.1.- Programa de creación de clasificadores para operario especializado.*

3.2.2.- Entrenando al programa

En el momento de terminar el entrenamiento del programa se obtiene como resultado unos archivos *.boost* donde podemos encontrar los valores devueltos por la máquina de aprendizaje, éstos son guardados para que posteriormente en la etapa de evaluación, o reconocimiento del modelo de avión, el programa pueda acceder a ellos y evaluar a partir de éstos valores, si el descriptor o descriptores que está tratando se aproximan más a los descriptores de un modelo u otro.

Para más información de cómo realizar este procedimiento puede leer en el apartado de anexos el punto *6.1.3.1.- Programa de creación de clasificadores para operario especializado.*

3.2.3.- Etapa de reconocimiento

Esta etapa del experimento corresponde al segundo programa propuesto en este proyecto que concierne al utilizado por el operario a pie de pista y es donde se comprueba que todo el trabajo realizado hasta el momento tiene alguna utilidad real.

Como se ha descrito al principio de este capítulo se utilizarán distintos valores fijos o estándar para acotar los resultados del experimento, y se dan como hechos y superados los pasos previos para poder realizar satisfactoriamente esta etapa del experimento.

Es importante destacar que tanto el vídeo utilizado como las fotografías no se han utilizado para entrenar la máquina de aprendizaje.

A continuación se procede a mostrar los pasos seguidos y los resultados obtenidos del experimento, aunque si se busca una explicación más detallada del funcionamiento del programa acudan al manual de usuario en el apartado de anexos en el punto *6.1.3.2.- Programa de evaluación para operario a pie de pista.*

3.2.3.1.- Reconocimiento utilizando fotografías

En este apartado se analizaran mediante todos los métodos descriptores todas las fotografías para cada uno de los modelos de avión.

3.2.3.1.1.- Caso modelo A320, método descriptor HOG

En la ilustración 13 se puede observar la pantalla principal del programa con los valores propuestos:

- Fotografía: modelo A320
- Método descriptor: HOG

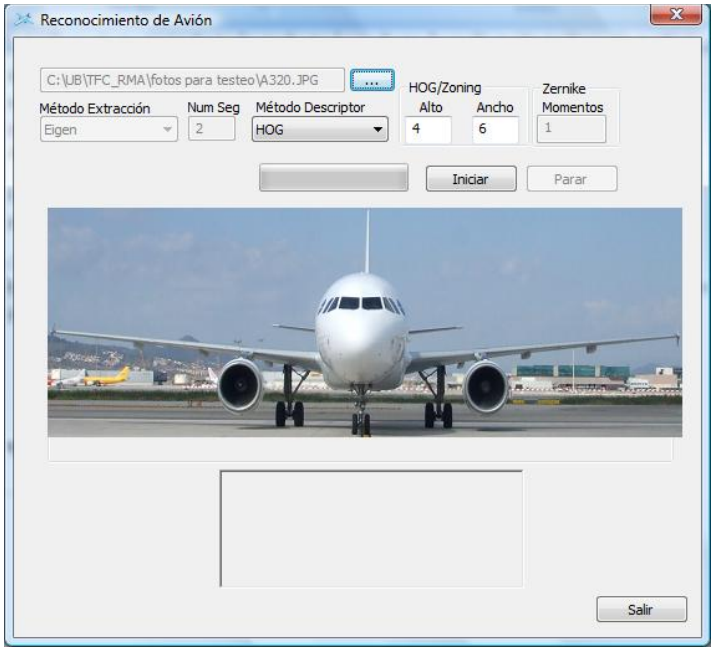


Ilustración 13 Pantalla principal con los valores propuestos para el modelo A320 utilizando como método descriptor HOG

El resultado obtenido para estos valores ha sido satisfactorio, dando como resultado el modelo A320 como se muestra en la ilustración 14:

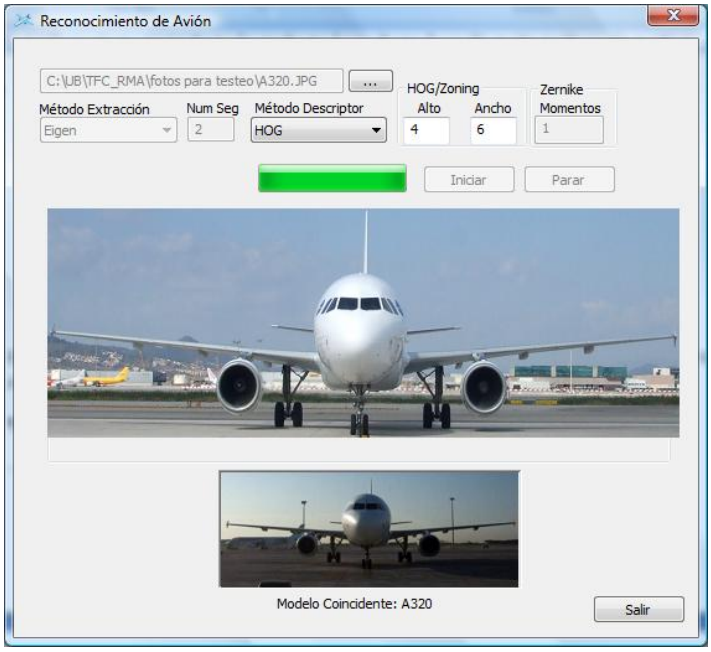


Ilustración 14 Resultado para los valores propuestos para el modelo A320 utilizando como método descriptor HOG

3.2.3.1.2.- Caso modelo A320, método descriptor Zernike

En la ilustración 15 se puede observar la pantalla principal del programa con los valores propuestos:

- Fotografía: modelo A320
- Método descriptor: Zernike

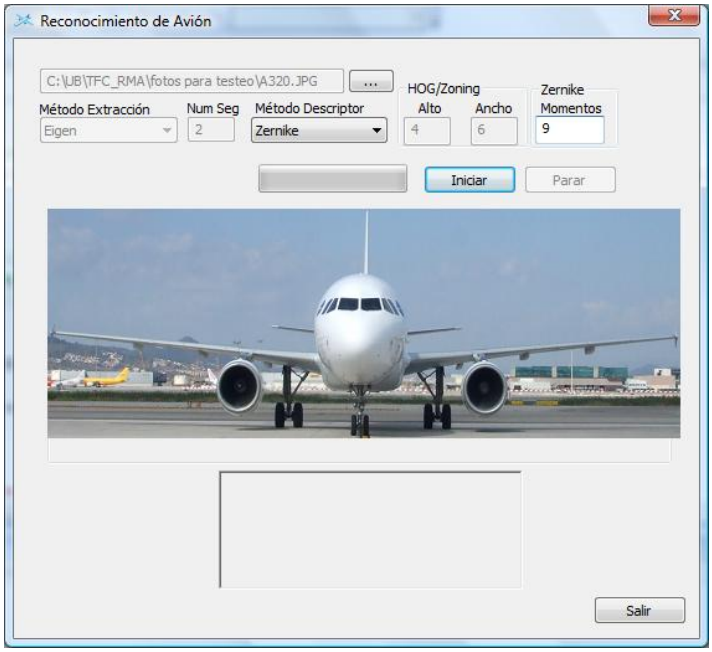


Ilustración 15 Pantalla principal con los valores propuestos para el modelo A320 utilizando como método descriptor Zernike

El resultado obtenido para estos valores ha sido satisfactorio, dando como resultado el modelo A320 como se muestra en la ilustración 16:

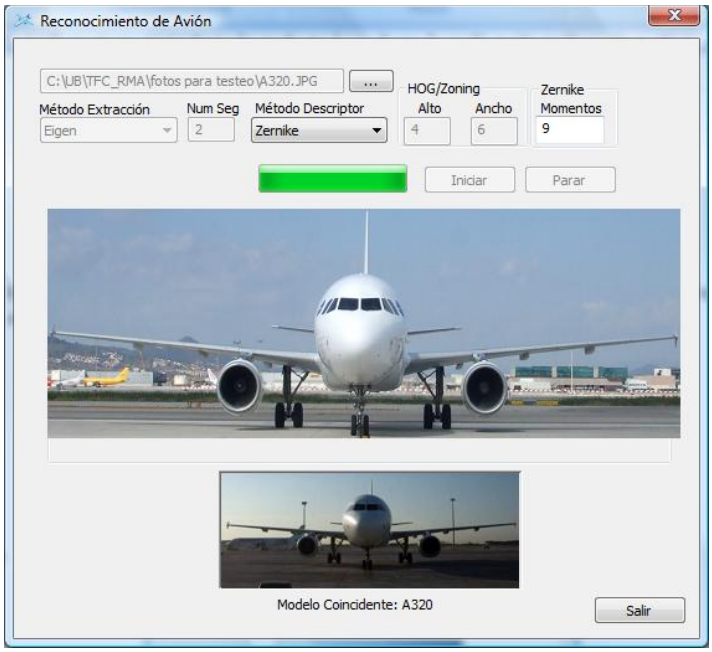


Ilustración 16 Resultado para los valores propuestos para el modelo A320 utilizando como método descriptor Zernike

3.2.3.1.3.- Caso modelo A320, método descriptor Zoning

En la ilustración 17 se puede observar la pantalla principal del programa con los valores propuestos:

- Fotografía: modelo A320
- Método descriptor: Zoning

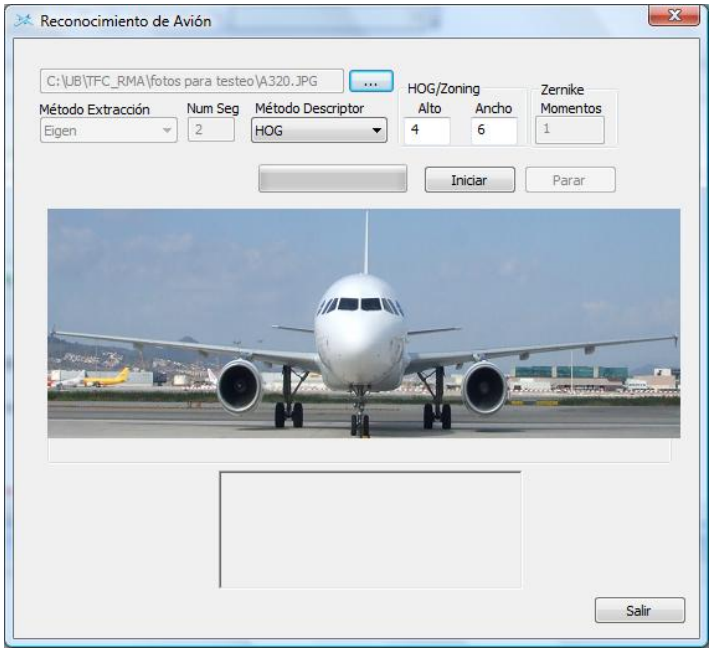


Ilustración 17 Pantalla principal con los valores propuestos para el modelo A320 utilizando como método descriptor Zoning

El resultado obtenido para estos valores ha sido satisfactorio, dando como resultado el modelo A320 como se muestra en la ilustración 18:

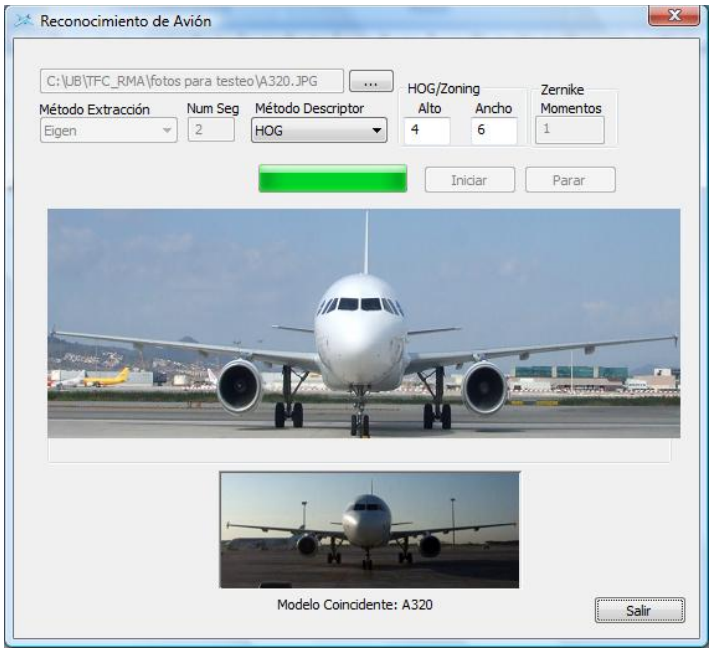


Ilustración 18 Resultado para los valores propuestos para el modelo A320 utilizando como método descriptor Zoning

3.2.3.1.4.- Caso modelo B717, método descriptor HOG

En la ilustración 19 se puede observar la pantalla principal del programa con los valores propuestos:

- Fotografía: modelo B717
- Método descriptor: HOG

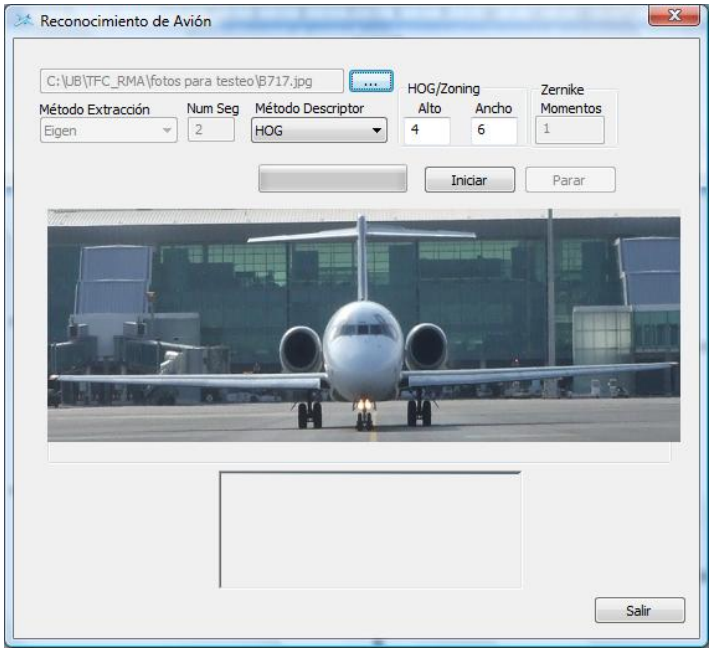


Ilustración 19 Pantalla principal con los valores propuestos para el modelo B717 utilizando como método descriptor HOG

El resultado obtenido para estos valores ha sido satisfactorio, dando como resultado el modelo B717 como se muestra en la ilustración 20:

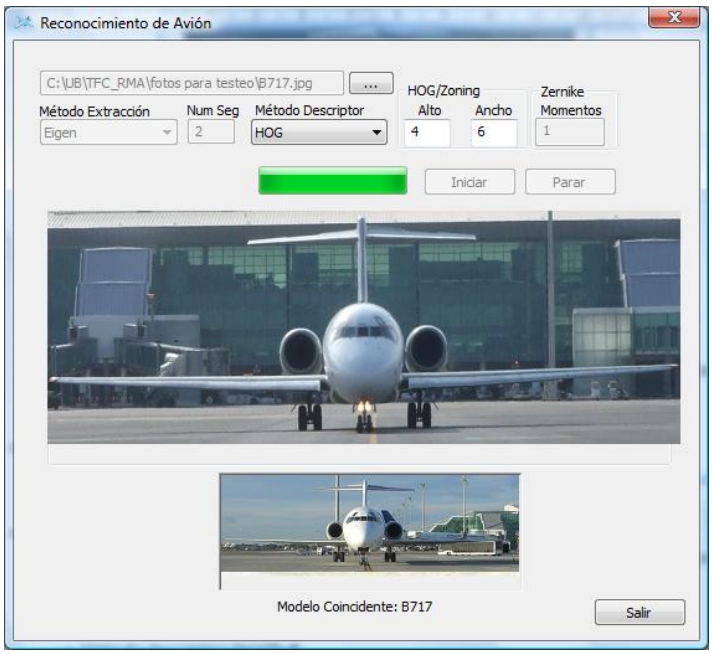


Ilustración 20 Resultado para los valores propuestos para el modelo B717 utilizando como método descriptor HOG

3.2.3.1.5.- Caso modelo B717, método descriptor Zernike

En la ilustración 21 se puede observar la pantalla principal del programa con los valores propuestos:

- Fotografía: modelo B717
- Método descriptor: Zernike

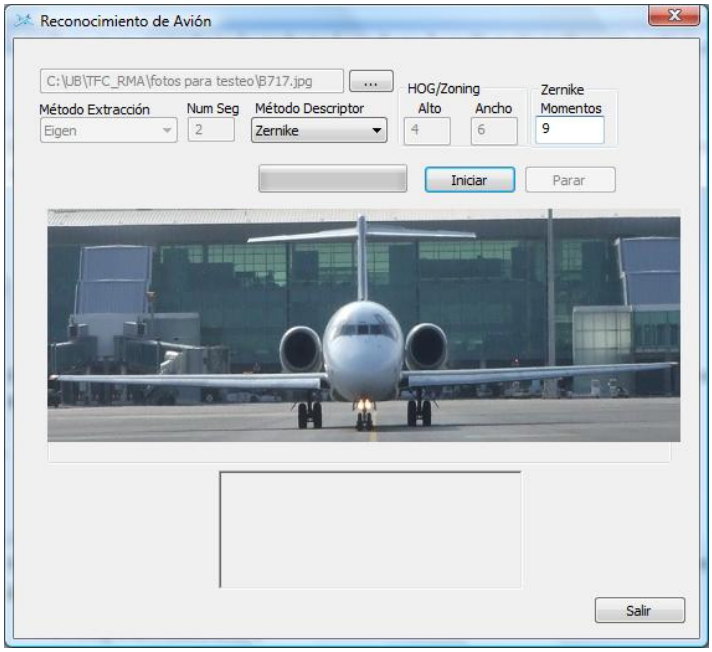


Ilustración 21 Pantalla principal con los valores propuestos para el modelo B717 utilizando como método descriptor Zernike

El resultado obtenido para estos valores ha sido satisfactorio, dando como resultado el modelo B717 como se muestra en la ilustración 22:

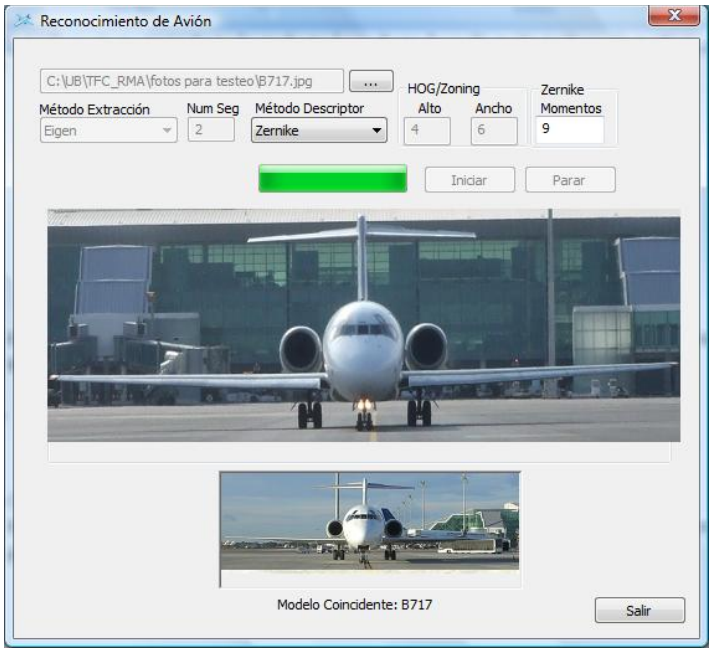


Ilustración 22 Resultado para los valores propuestos para el modelo B717 utilizando como método descriptor Zernike

3.2.3.1.6.- Caso modelo B717, método descriptor Zoning

En la ilustración 23 se puede observar la pantalla principal del programa con los valores propuestos:

- Fotografía: modelo B717
- Método descriptor: Zoning

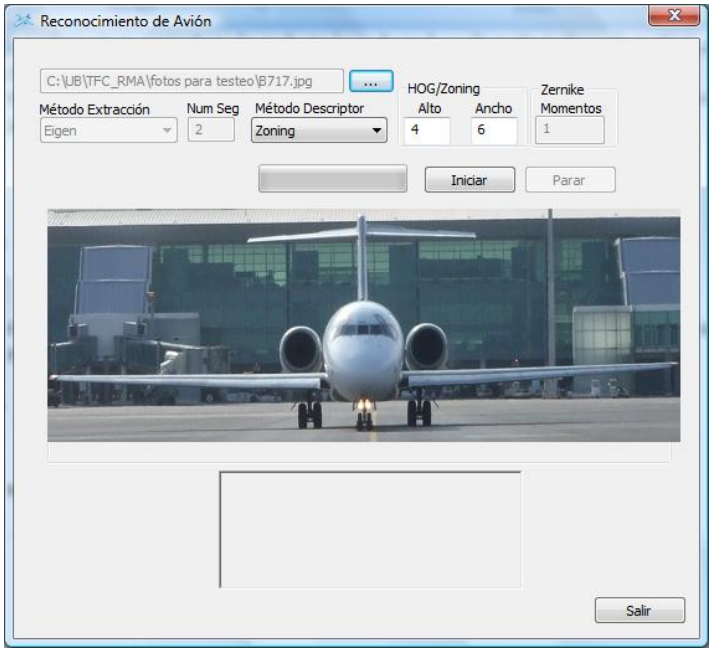


Ilustración 23 Pantalla principal con los valores propuestos para el modelo B717 utilizando como método descriptor Zoning

El resultado obtenido para estos valores ha sido satisfactorio, dando como resultado el modelo B717 como se muestra en la ilustración 24:

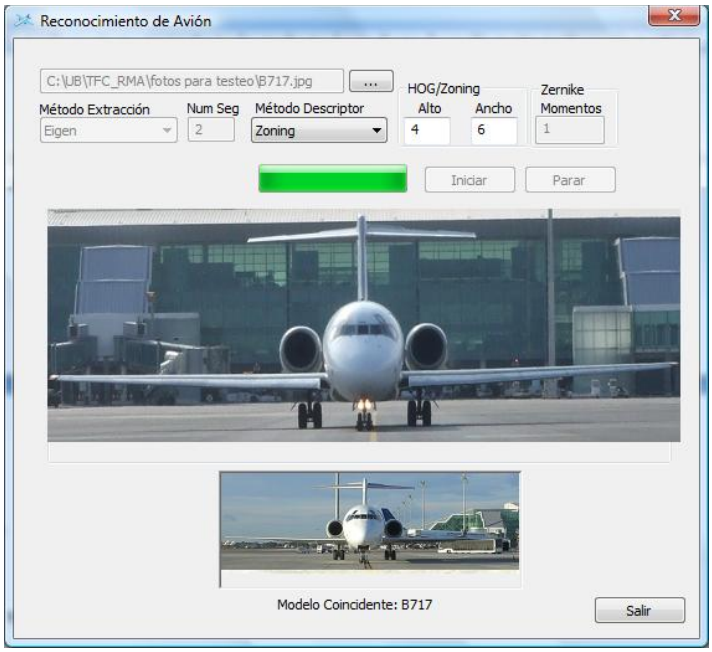


Ilustración 24 Resultado para los valores propuestos para el modelo B717 utilizando como método descriptor Zoning

3.2.3.1.7.- Caso modelo B737, método descriptor HOG

En la ilustración 25 se puede observar la pantalla principal del programa con los valores propuestos:

- Fotografía: modelo B737
- Método descriptor: HOG

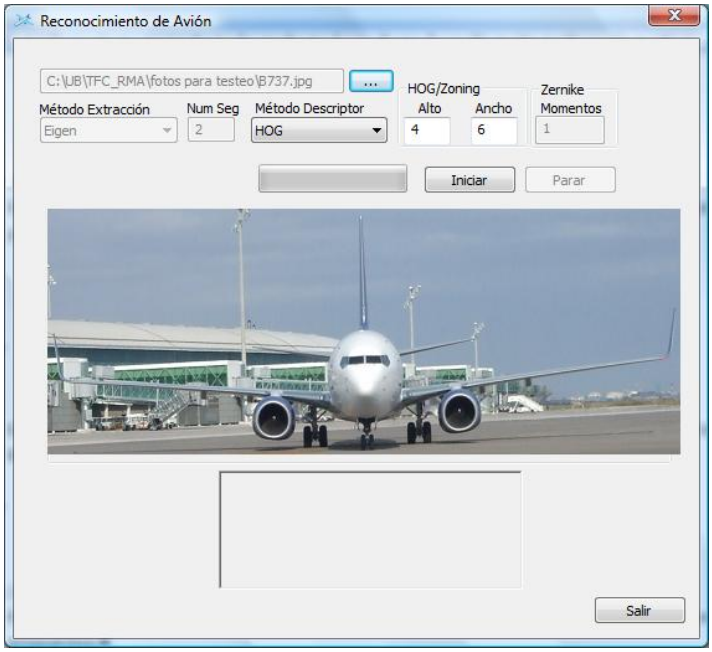


Ilustración 25 Pantalla principal con los valores propuestos para el modelo B737 utilizando como método descriptor HOG

El resultado obtenido para estos valores ha sido satisfactorio, dando como resultado el modelo B737 como se muestra en la ilustración 26:

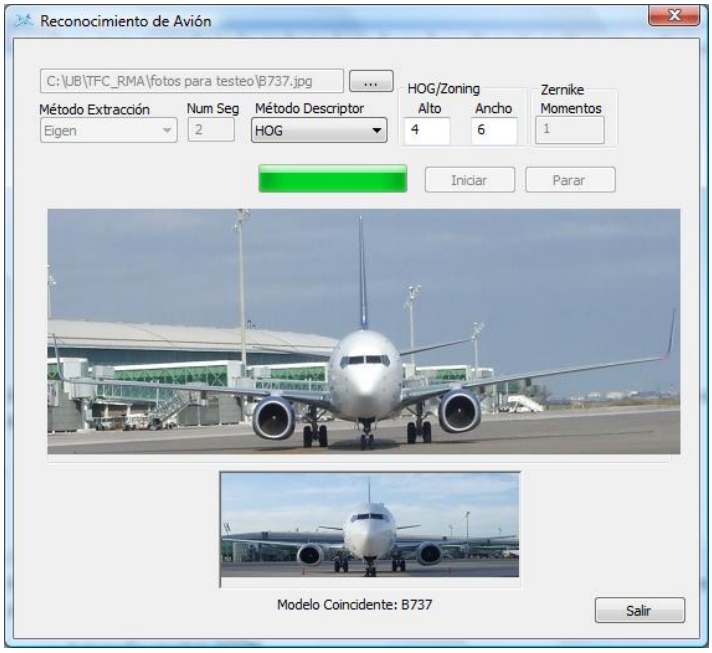


Ilustración 26 Resultado para los valores propuestos para el modelo B737 utilizando como método descriptor HOG

3.2.3.1.8.- Caso modelo B737, método descriptor Zernike

En la ilustración 27 se puede observar la pantalla principal del programa con los valores propuestos:

- Fotografía: modelo B737
- Método descriptor: Zernike

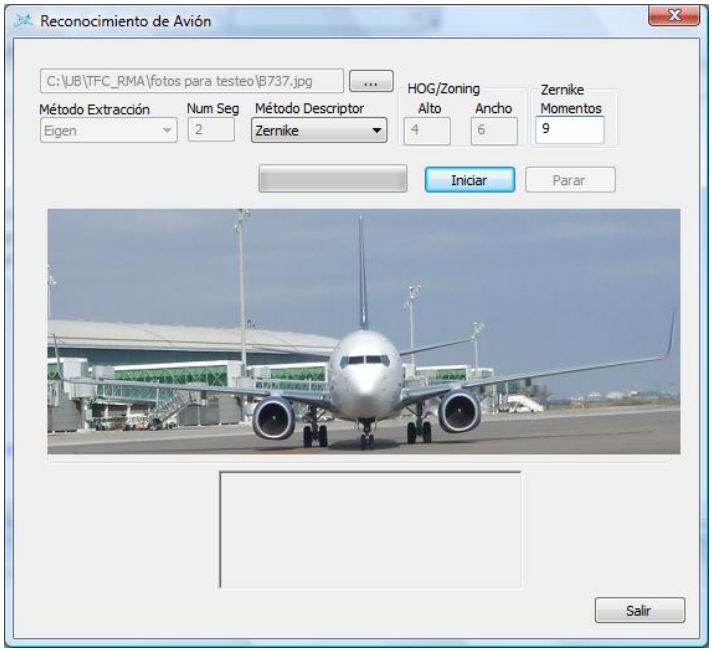


Ilustración 27 Pantalla principal con los valores propuestos para el modelo B737 utilizando como método descriptor Zernike

El resultado obtenido para estos valores ha sido satisfactorio, dando como resultado el modelo B737 como se muestra en la ilustración 28:

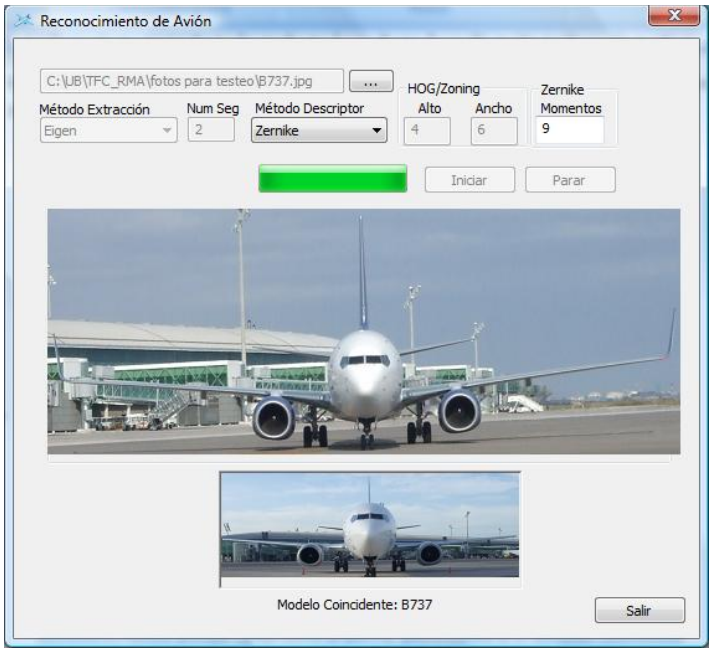


Ilustración 28 Resultado para los valores propuestos para el modelo B737 utilizando como método descriptor Zernike

3.2.3.1.9.- Caso modelo B737, método descriptor Zoning

En la ilustración 29 se puede observar la pantalla principal del programa con los valores propuestos:

- Fotografía: modelo B737
- Método descriptor: Zoning

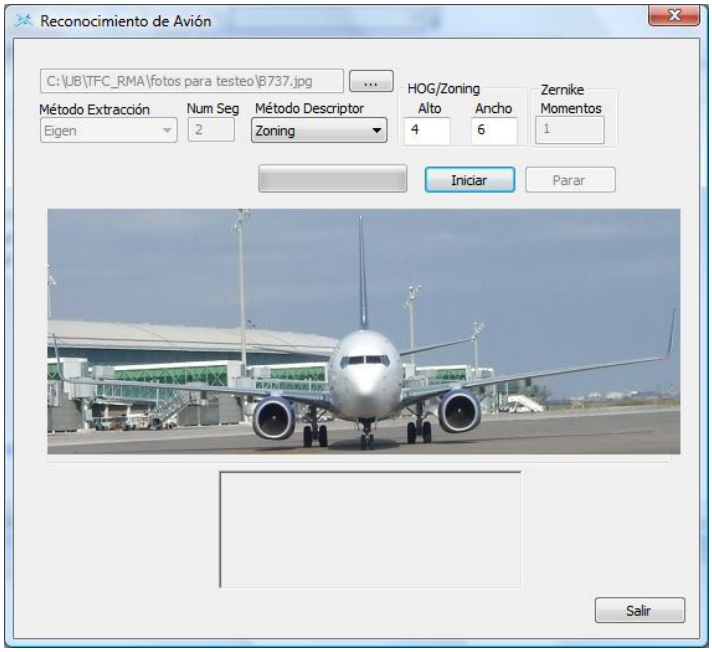


Ilustración 29 Pantalla principal con los valores propuestos para el modelo B737 utilizando como método descriptor Zoning

El resultado obtenido para estos valores ha sido no satisfactorio, dando como resultado el modelo A320 como se muestra en la ilustración 30:

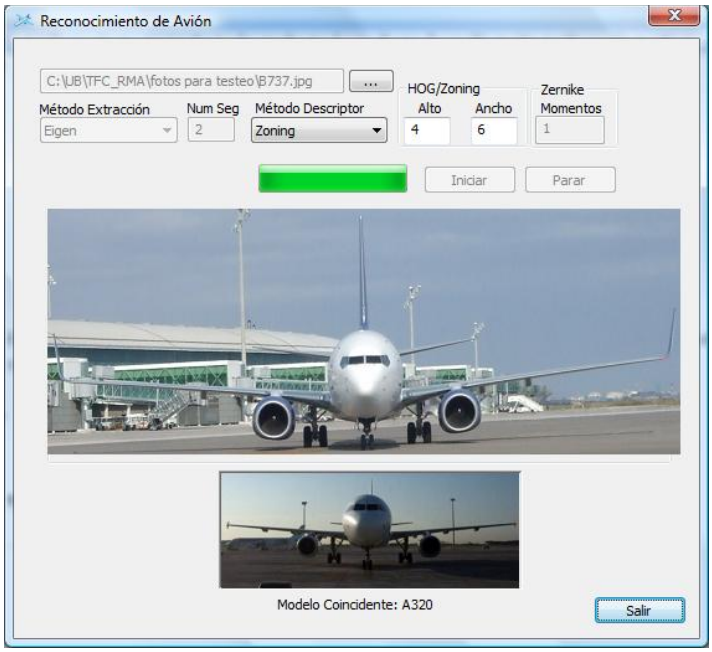


Ilustración 30 Resultado para los valores propuestos para el modelo B737 utilizando como método descriptor Zoning

3.2.3.1.10.- Caso modelo BA146, método descriptor HOG

En la ilustración 31 se puede observar la pantalla principal del programa con los valores propuestos:

- Fotografía: modelo BA146
- Método descriptor: HOG

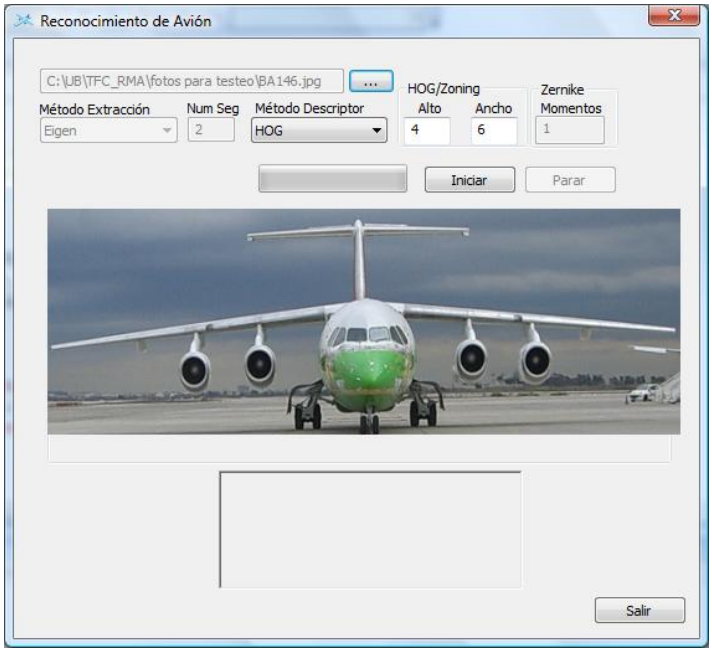


Ilustración 31 Pantalla principal con los valores propuestos para el modelo BA146 utilizando como método descriptor HOG

El resultado obtenido para estos valores ha sido satisfactorio, dando como resultado el modelo BA146 como se muestra en la ilustración 32:

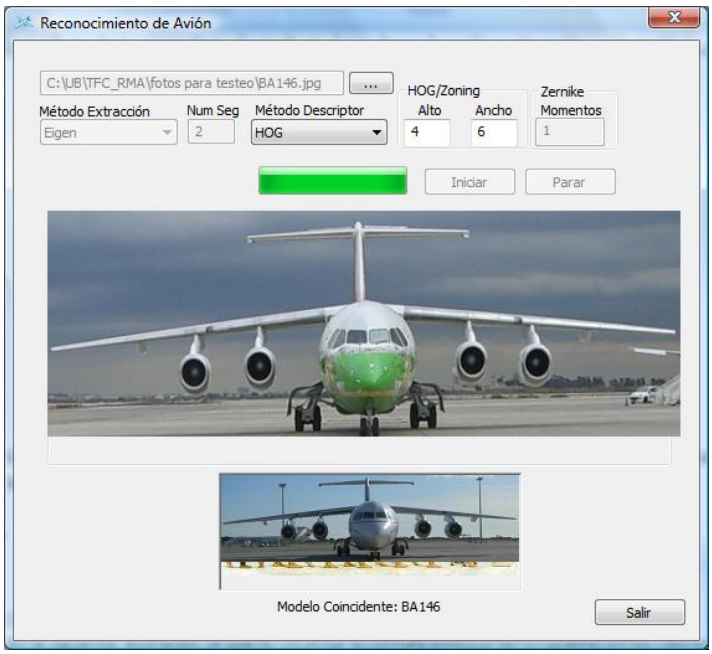


Ilustración 32 Resultado para los valores propuestos para el modelo BA146 utilizando como método descriptor HOG

3.2.3.1.11.- Caso modelo BA146, método descriptor Zernike

En la ilustración 33 se puede observar la pantalla principal del programa con los valores propuestos:

- Fotografía: modelo BA146
- Método descriptor: Zernike

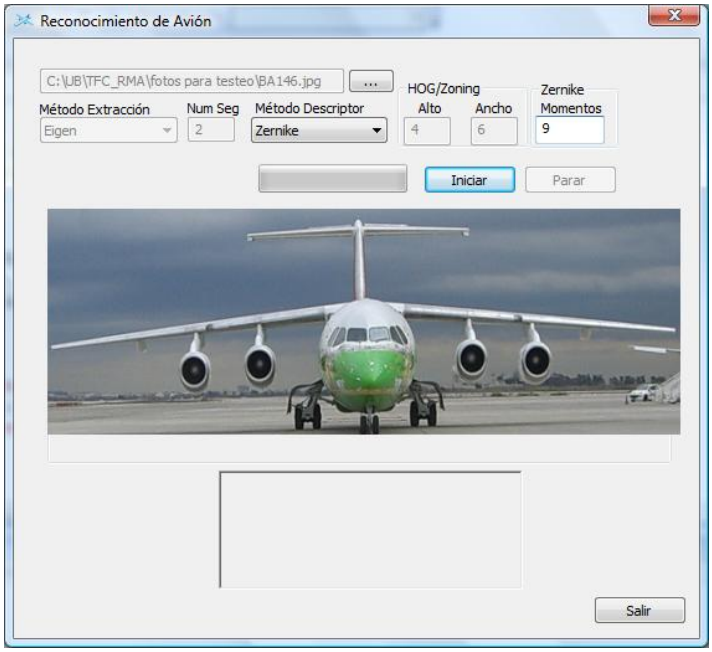


Ilustración 33 Pantalla principal con los valores propuestos para el modelo BA146 utilizando como método descriptor Zernike

El resultado obtenido para estos valores ha sido satisfactorio, dando como resultado el modelo BA146 como se muestra en la ilustración 34:

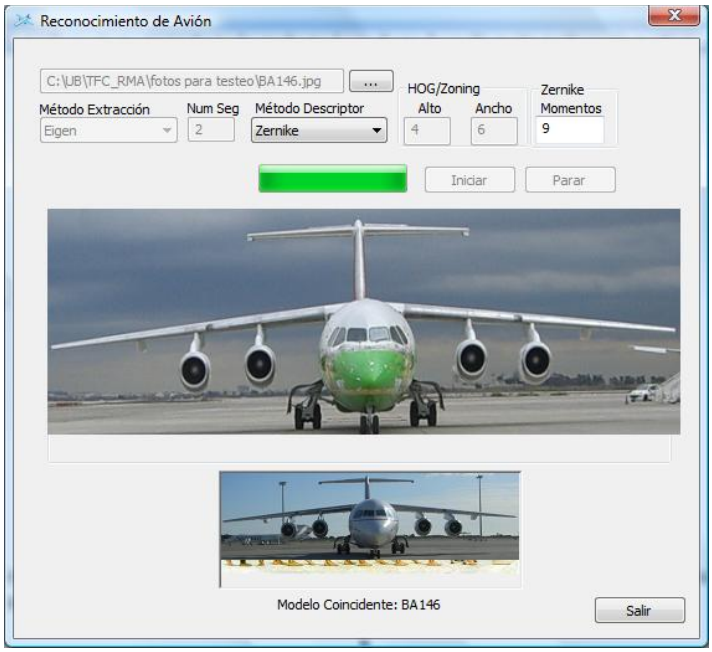


Ilustración 34 Resultado para los valores propuestos para el modelo BA146 utilizando como método descriptor Zernike

3.2.3.1.12.- Caso modelo BA146, método descriptor Zoning

En la ilustración 35 se puede observar la pantalla principal del programa con los valores propuestos:

- Fotografía: modelo BA146
- Método descriptor: Zoning

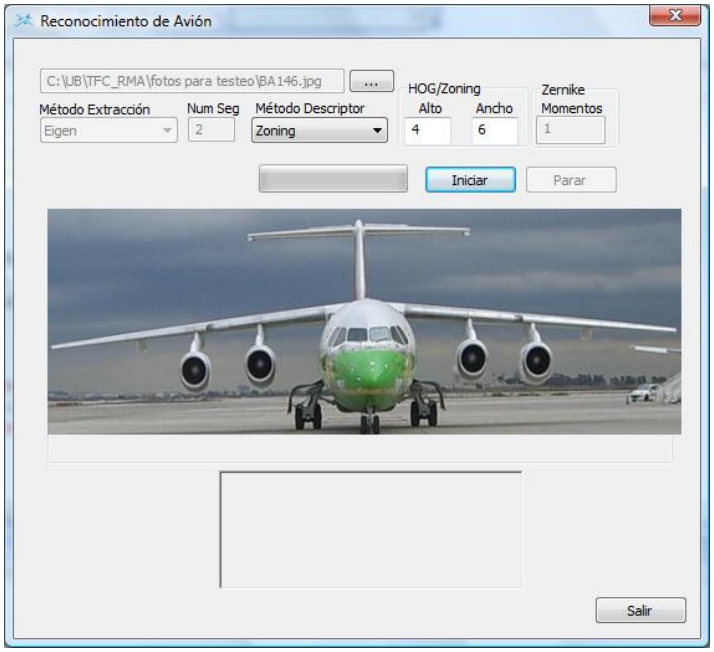


Ilustración 35 Pantalla principal con los valores propuestos para el modelo BA146 utilizando como método descriptor Zoning

El resultado obtenido para estos valores ha sido satisfactorio, dando como resultado el modelo BA146 como se muestra en la ilustración 36:

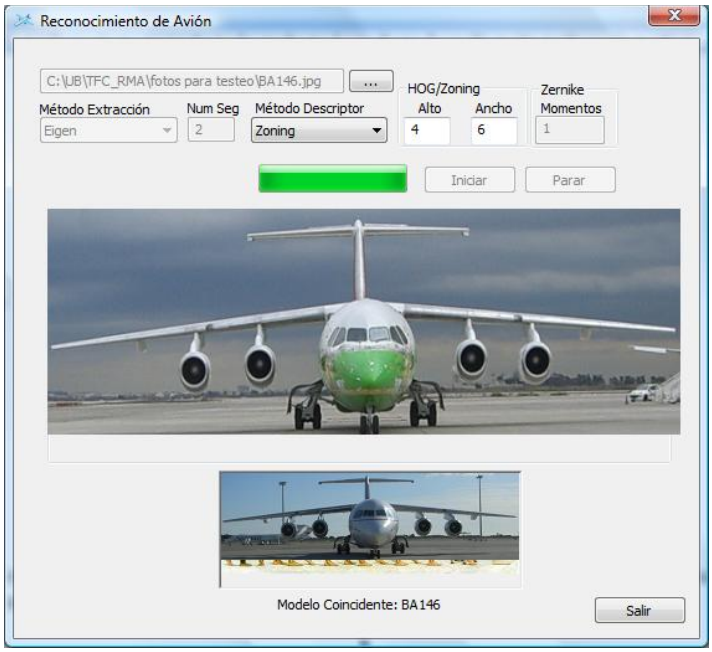


Ilustración 36 Resultado para los valores propuestos para el modelo B737 utilizando como método descriptor Zoning

3.2.3.2.- Reconocimiento utilizando vídeo

En este apartado del experimento se observarán los resultados obtenidos utilizando como datos de entrada las imágenes de vídeo.

3.2.3.2.1.- Caso modelo A320, Adaptative + HOG

En la ilustración 37 se puede observar la pantalla principal del programa con los valores propuestos:

- Vídeo: modelo A320
- Método de extracción: Adaptative
- Número de segundos: 2
- Método descriptor: HOG

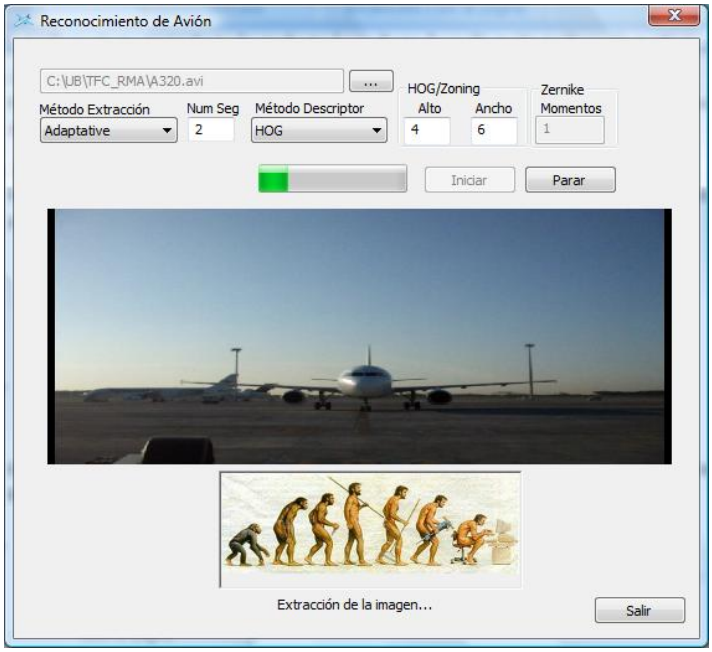


Ilustración 37 Pantalla principal con los valores propuestos para el modelo A320: Adaptative2 + HOG

El resultado obtenido para estos valores ha sido satisfactorio, dando como resultado el modelo A320 como se muestra en la ilustración 38:

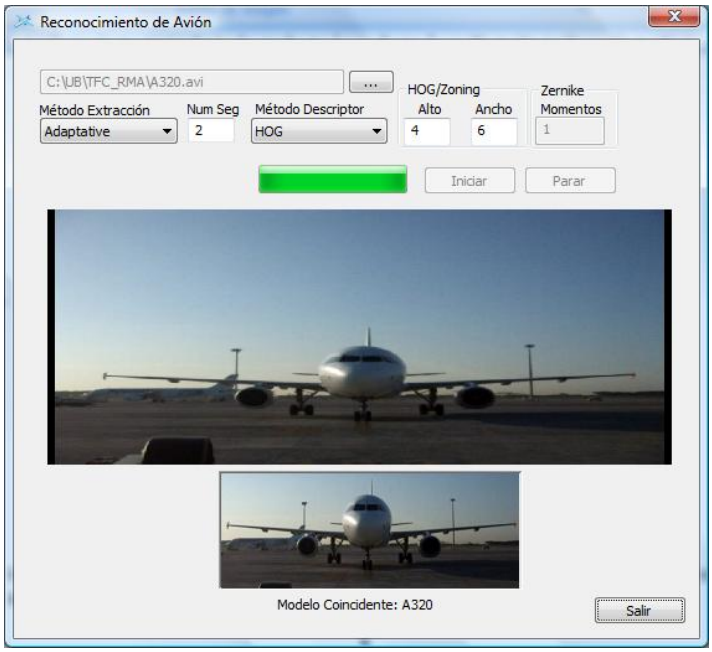


Ilustración 38 Resultado para los valores propuestos para el modelo A320: Adaptative2 + HOG

En la ilustración 39 se puede observar la pantalla principal del programa con los valores propuestos:

- Vídeo: modelo A320
- Método de extracción: Adaptative
- Número de segundos: 8
- Método descriptor: HOG

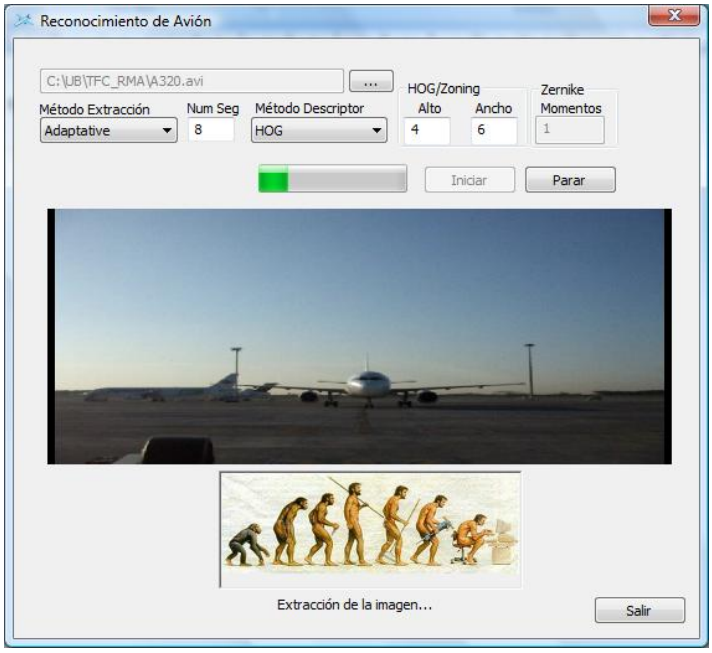


Ilustración 39 Pantalla principal con los valores propuestos para el modelo A320: Adaptative8 + HOG

El resultado obtenido para estos valores ha sido no satisfactorio, dando como resultado el modelo B717 como se muestra en la ilustración 40:

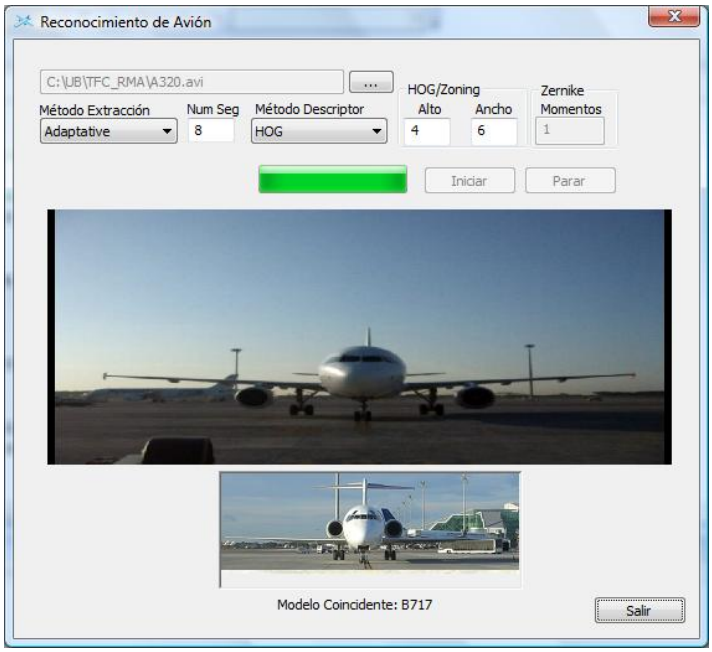


Ilustración 40 Resultado para los valores propuestos para el modelo A320: Adaptative8 + HOG

3.2.3.2.2.- Caso modelo A320,Mean + HOG

En la ilustración 41 se puede observar la pantalla principal del programa con los valores propuestos:

- Vídeo: modelo A320
- Método de extracción: Mean
- Número de segundos: 2
- Método descriptor: HOG

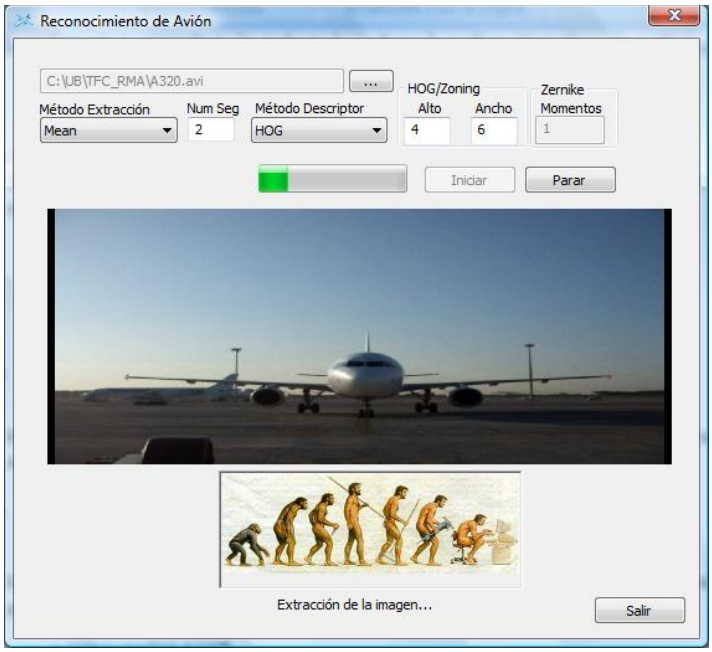


Ilustración 41 Pantalla principal con los valores propuestos para el modelo A320: Mean2 + HOG

El resultado obtenido para estos valores ha sido no satisfactorio, dando como resultado el modelo B737 como se muestra en la ilustración 42:

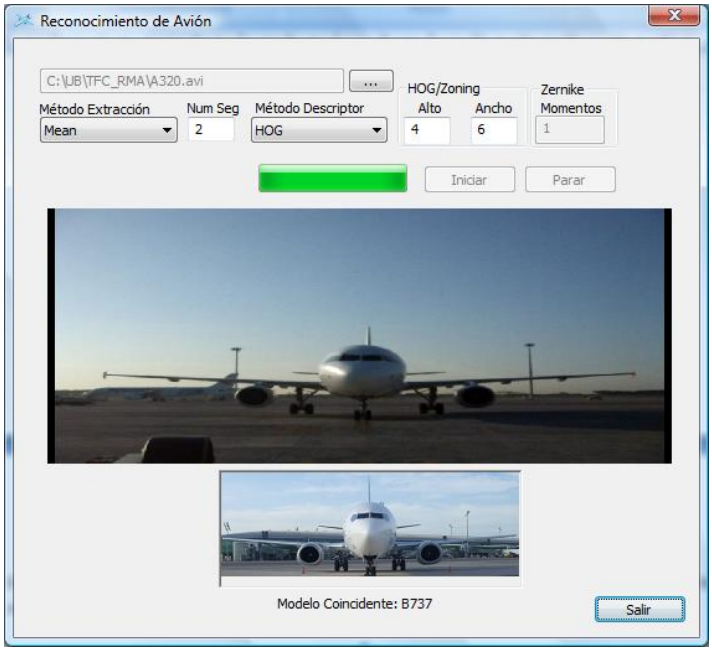


Ilustración 42 Resultado para los valores propuestos para el modelo A320: Mean2 + HOG

En la ilustración 43 se puede observar la pantalla principal del programa con los valores propuestos:

- Vídeo: modelo A320
- Método de extracción: Mean
- Número de segundos: 8
- Método descriptor: HOG

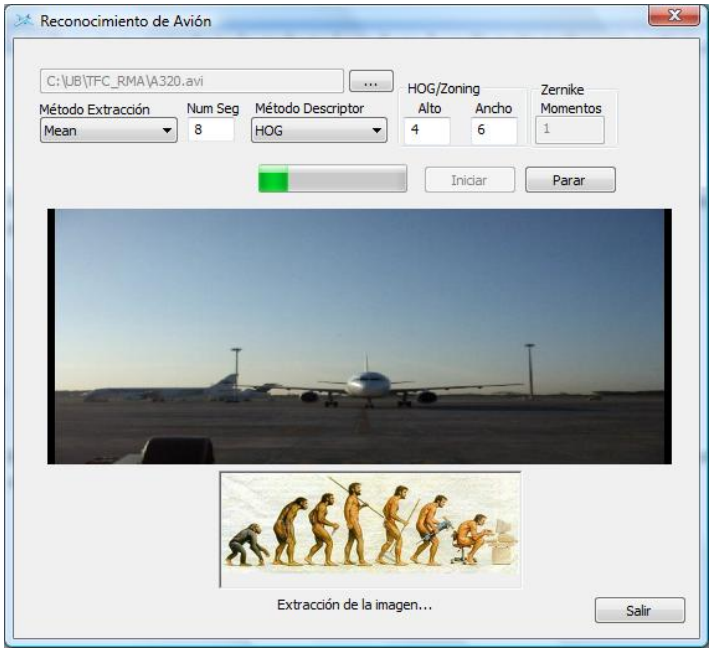


Ilustración 43 Pantalla principal con los valores propuestos para el modelo A320: Mean8 + HOG

El resultado obtenido para estos valores ha sido no satisfactorio, dando como resultado el modelo B737 como se muestra en la ilustración 44:

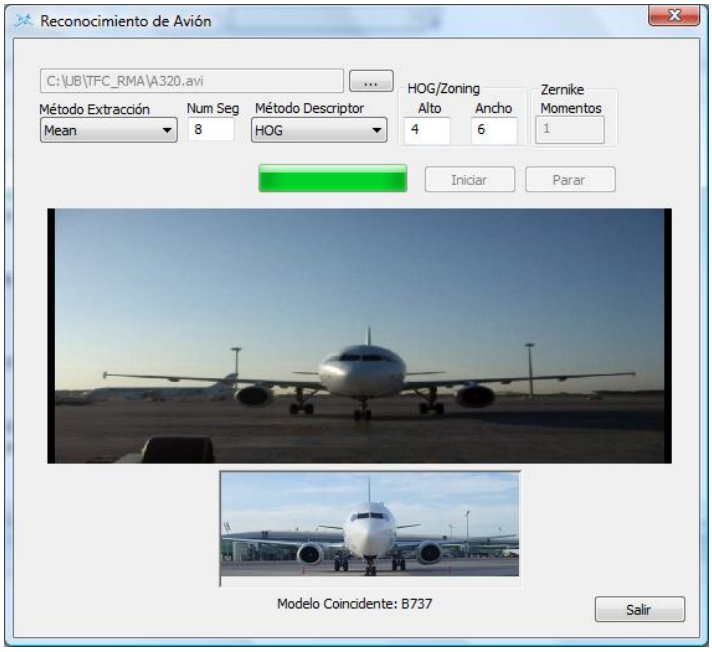


Ilustración 44 Resultado para los valores propuestos para el modelo A320: Mean8 + HOG

3.2.3.3.- Interpretación de resultados

En este apartado se presenta de manera más detallada los resultados obtenidos en los apartados 3.2.3.1 y 3.2.3.2 con el fin de aglutinar toda la información obtenida en todas las pruebas realizadas y no detalladas anteriormente.

En la tabla 3 se puede observar una comparativa entre métodos descriptores utilizando fotografías como datos de entrada:

	A320	B717	B737	BA146
HOG	✓	✓	✓	✓
Zernike	✓	✓	✓	✓
Zoning	✓	✓	✗	✓

Tabla 3 Comparativa entre métodos descriptores

Mirando estos resultados se puede observar a simple vista que, de los tres métodos descriptores, el único que ha fallado y por tanto el de menor precisión es Zoning. Con este resultado se podría intuir cual de los tres métodos podría ser de los peores.

En la tabla 4 se puede observar una comparativa combinando los métodos de extracción con los métodos descriptores entre todos los modelos, utilizando 2 segundos antes de finalizar el video para aproximar lo máximo posible el avión al objetivo de la cámara. Los casos no acertados se muestran con una X.

2 segundos	A320			B717			B737			BA146		
	HOG	Zernike	Zoning	HOG	Zernike	Zoning	HOG	Zernike	Zoning	HOG	Zernike	Zoning
Eigen	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Wren	1,03m	✗	✗	✗	✗	1,03m	✗	1,02m	✗	1,14m	1,07m	✗
Adaptative	34s	33,7s	30,6s	36,7s	✗	32,3s	✗	✗	✗	42,5s	✗	✗
Grimson	3,18m	✗	✗	✗	✗	2,59m	✗	2,05m	✗	2,37m	2,24m	✗
Zivkovik	43,6s	✗	✗	✗	✗	41,97s	✗	39,22s	✗	40,4s	✗	✗
Prati	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Mean	✗	✗	✗	✗	41,6s	✗	38,6s	✗	✗	✗	42,4s	41,9s

Tabla 4 Comparativa cualitativa entre métodos

A continuación se mostrará, desde el Gráfico 1 hasta el Gráfico 4, los tiempos obtenidos correspondientes a los vídeos de los distintos modelos de avión combinando los diferentes métodos 2 segundos antes de terminar el vídeo. El tiempo está expresado en segundos:

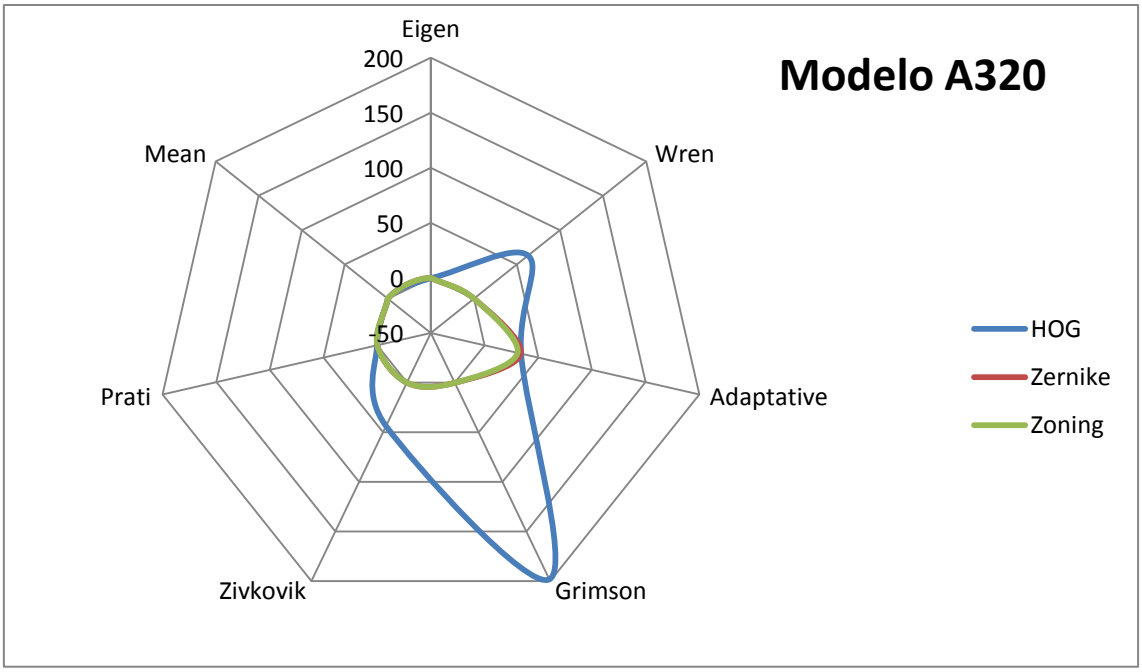


Gráfico 1 Tiempo obtenido por el modelo A320

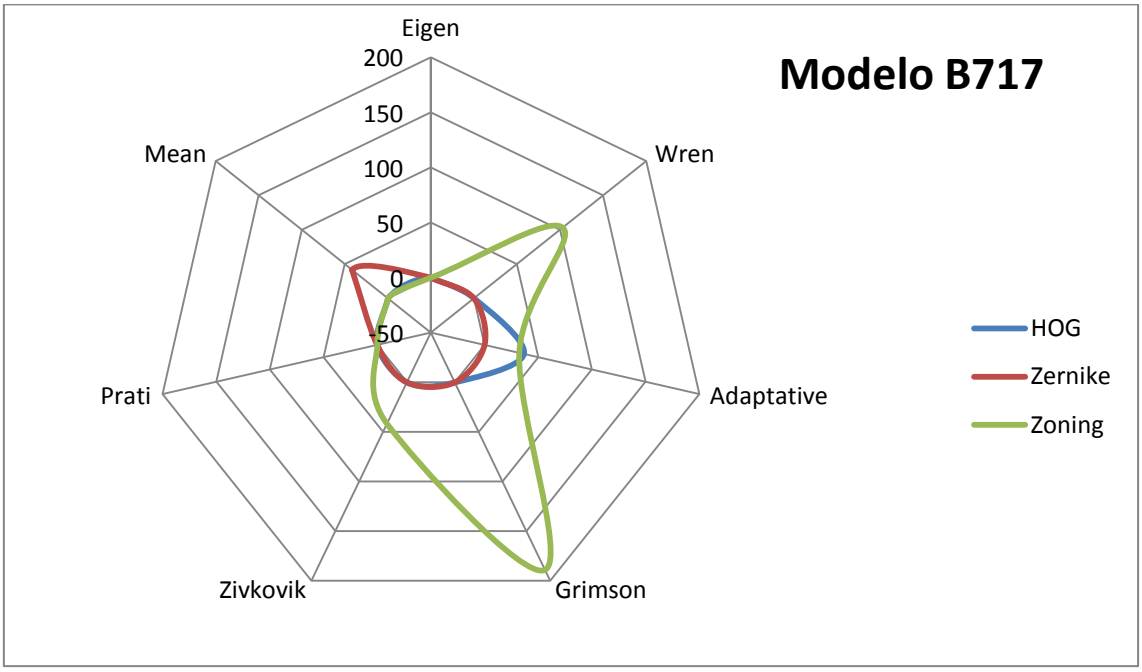


Gráfico 2 Tiempo obtenido por el modelo B717

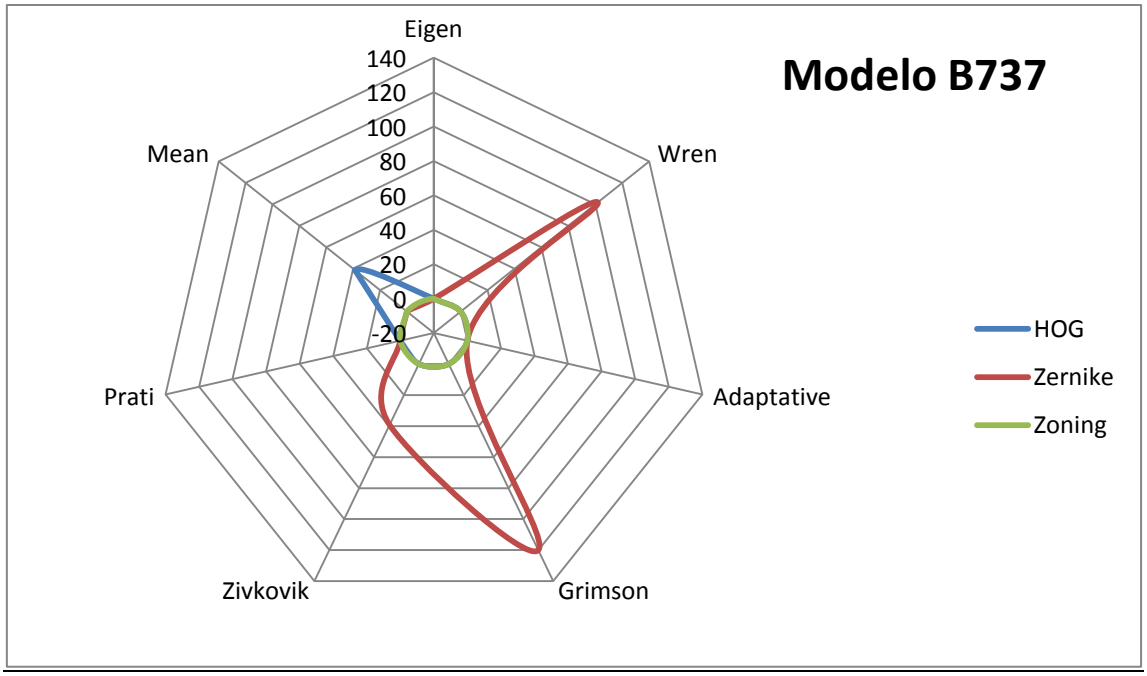


Gráfico 3 Tiempo obtenido por el modelo B737

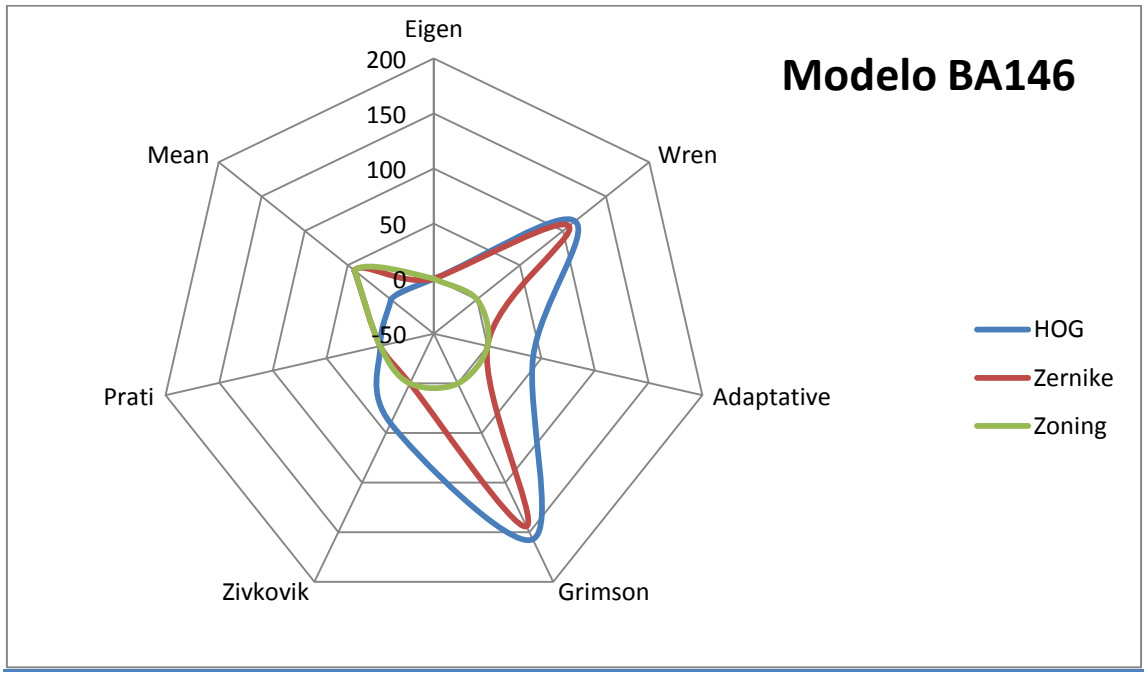


Gráfico 4 Tiempo obtenido por el modelo BA146

La combinación ideal entre método de extracción y método descriptor deberá ser aquella que más cerca se mantenga de 0 sin llegar a tener tal valor. De esta manera podemos concluir que para todos los modelos menos para el B737, la mejor combinación es *Adaptative + HOG,* mientras que para el modelo B737 la mejor combinación es *Mean + HOG*.

Finalmente en líneas generales, se puede concluir que el mejor método de extracción según su tiempo de ejecución es el método Adaptative y el mejor método descriptor se corresponde a HOG por su mayor número de aciertos aunque Zoning sea el método descriptor más rápido.

4.- Conclusiones

En este proyecto se ha desarrollado un software en lenguaje visual C++ capaz de resolver de forma eficaz y en un porcentaje elevado el problema de distinguir entre diferentes modelos de avión mediante capturas de imágenes a partir de un vídeo de entrada.

Éste ha sido un proyecto de ingeniería realizado en un entorno real, cercano a un problema existente en estos momentos en el Aeropuerto de Barcelona, por este motivo se ha querido dotar al programa de la documentación más rigurosa posible, con la finalidad de desarrollar un software robusto y especializado. Durante todo el proyecto se ha seguido una metodología en cascada compuesta por diferentes etapas que se han ido desarrollando de forma secuencial. Éstos se ha podido ver en la sucesión de los apartados de esta memoria.

4.1.- Conclusiones de las diferentes etapas del proyecto

Este apartado se divide en 6 subapartados, correspondientes a cada etapa del programa, donde se comentarán las conclusiones extraídas durante el transcurso del proyecto.

4.1.1.- Métodos de Segmentación

Esta etapa del desarrollo correspondería a los ojos del programa, por este motivo, para realizar la extracción de las imágenes a partir de un vídeo se han utilizado funciones pertenecientes a las librerías de openCV y blobs, y se han desarrollado 7 métodos diferentes encargados de segmentar la imagen hasta conseguir como resultado una nueva imagen donde el avión estuviera acotado. Este trabajo ha sido realizado por Marc Garcia y Ramis y posteriormente integrado en el proyecto.

Después de la etapa de Test se ha podido concluir que de los 7 métodos disponibles los que mejor se adaptan tanto en tiempo de ejecución como en precisión a la hora de acotar la imagen del avión han sido los métodos Adaptative, Zivkovik y Wren. Éstos han tenido el tiempo de respuesta más bajo (tiempo valorado en segundos) y han sido los que mejor precisión han ofrecido respectivamente. Esto se puede observar en el apartado de Resultados, donde se muestran tablas y gráficos explicativos.

4.1.2.- Métodos descriptores

Esta etapa del desarrollo correspondería a la parte del cerebro capaz de discernir que está viendo. Para que el programa sea capaz de realizar la misma función se han desarrollado 3 métodos descriptores diferentes encargados de describir la imagen utilizando desde el gradiente de orientación de la silueta del avión hasta los puntos que lo componen. Estos métodos se han ayudado de la librería openCV para tratar las imágenes.

HOG ha resultado ser un método complejo de gestionar dada la gran cantidad de información que genera conforme se va seccionando más la imagen de entrada en subimagenes, cosa que el método Zoning dada su simplicidad no ha sido tan dificultosa.

Después de la etapa de testeado se puede concluir que de los 3 métodos utilizados el más preciso en cuanto a número de aciertos es el método HOG con un porcentaje de aciertos del 100%, en el caso de fotografías como elemento de entrada, y el método más rápido es Zoning con un tiempo de respuesta más bajo, valorado en segundos. Zernike se trata de el método

con el rendimiento más bajo con un porcentaje de aciertos del 45% y un tiempo de respuesta intermedio.

4.1.3.- Entrenamiento de la máquina de aprendizaje

Esta etapa del desarrollo correspondería a las partes del cerebro encargadas del aprendizaje. Para poder emular este proceso se ha utilizado el algoritmo de aprendizaje Adaboost que utiliza los descriptores generados anteriormente para generar como datos de salida los clasificadores para cada modelo de avión, algo parecido a la memoria en los humanos.

La calidad de los clasificadores es directamente proporcional a la calidad de los descriptores. Adaboost ha resultado ser un algoritmo elegante, rápido y fácil de utilizar. Su implementación en Matlab le dota de rapidez y sencillez a la hora de escribir el código.

4.1.4.- Reconocimiento de los modelos de avión

Esta etapa del desarrollo es el corazón del proyecto y dota de inteligencia al programa. Se ha utilizado el método Voting para decidir si un descriptor procedente de una imagen de entrada corresponde a un modelo de avión u a otro. Esta metodología es la manera más rápida y fácil de extender los clasificadores binarios creados en la etapa anterior a un sistema multi-clase. Esto permite al programa no solo poder distinguir entre 2 modelos de avión sino entre varios. El método consiste en comparar el descriptor de entrada con los clasificadores obtenidos, el modelo de avión que más aciertos haya conseguido se da como el modelo correcto. El resultado obtenido después de utilizar esta metodología ha sido muy bueno cumpliendo al 100% con su objetivo.

4.1.5.- Desarrollo de la interfaz

El desarrollo de la interfaz ha sido pensada para ser intuitiva y fácil de utilizar por el usuario. Se ha priorizado la usabilidad en el manejo de los controles, haciendo más robusta la lógica del programa, blindando aquellas partes en las que por error del usuario podría llegar a ocurrir algún fallo inesperado.

4.1.6.- Creación del instalador

Con el fin de conseguir un programa más fácil de utilizar y amigable para el usuario se ha desarrollado un asistente de instalación que acompaña al usuario en la instalación de todos los programas ejecutables, librerías dll, clasificadores y material multimedia de ejemplo, carpetas necesarias para la correcta ejecución del programa y manuales de usuario.

4.2.- Continuidad del proyecto

Como continuidad del proyecto se propone:

- Una optimización de los métodos de extracción utilizados en cuanto a tiempo de ejecución para conseguir una respuesta más rápida del programa y optimizar todo el proceso al operador de pista.
- La integración final con el actual entorno de trabajo del operador de pista.
- Algoritmo capaz de determinar la posición más idónea para capturar la imagen del avión utilizando el Sistema de Guía de Atraque (SGA) existente en el Aeropuerto. Para más información acerca del SGA puede leer el manual que se encuentra en el punto *6.2.- Manual Sistema de Guía de Atraque*.

5.-Bibliografía

[1]. Descriptor. [En línea] http://es.wikipedia.org/wiki/Descriptores_visuales.

[2]. *Zernike.* [En línea] http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/SHUTLER3/node11.html.

[3]. Clasificador de imagenes. [En línea] http://www.gdf-hannover.de/lit_html/nutshell_v10_es/node23.html.

[4]. Adaboost. [En línea] Adaboost: http://en.wikipedia.org/wiki/AdaBoost.

[5]. **Schapier, Yoav Freund & Robert.** *ADDITIVE LOGISTIC REGRESSION: AN STATISTICAL POINT OF VIEW OF ADABOOST.*

[6]. OpenCV Wiki. [En línea] http://opencv.willowgarage.com/wiki/.

[7]. wikipedia. [En línea] http://es.wikipedia.org/wiki/Imagen_digital.

[8]. **Wolf, A. B. Bhatia and E.** *On the circle polynomials of Zernike and related orthogonal sets.* s.l. : Proc. Cambridge Philosophical Society, 1954. 50.

[9]. Métodos de Extracción. [En línea] http://dparks.wikidot.com/source-code

6.-Anexos

6.1.-Manual de Usuario

Antes de empezar con la instalación del programa debe asegurarse de tener instaladas tanto las librerías OpenCV como las librerías Blobs, con el fin de evita posible fallos inesperados del programa. Podrá encontrar manuales para su instalación en el CD adjuntado más adelante.

Una vez instaladas sendas librerías ya puede proceder a ejecutar la instalación del programa de Reconocimiento Automático de Modelos de Avión en Secuencias de Vídeo.

6.1.1.- Instalación del programa

En la carpeta programas del CD adjunto podrá encontrar el archivo instalable *TFC_RMA.msi* para plataforma Windows Vista hacia adelante o *setup.exe* para sistemas anteriores. De *doble clic sobre el icono* o *botón derecho sobre el icono->instalar* para instalar el programa. El sistema comenzara la instalación del programa, siga los pasos indicados a continuación hasta finalizar la instalación.

1) El sistema nos indicará que se va a instalar el programa en nuestro sistema, clicamos siguiente como muestra la ilustración 45.

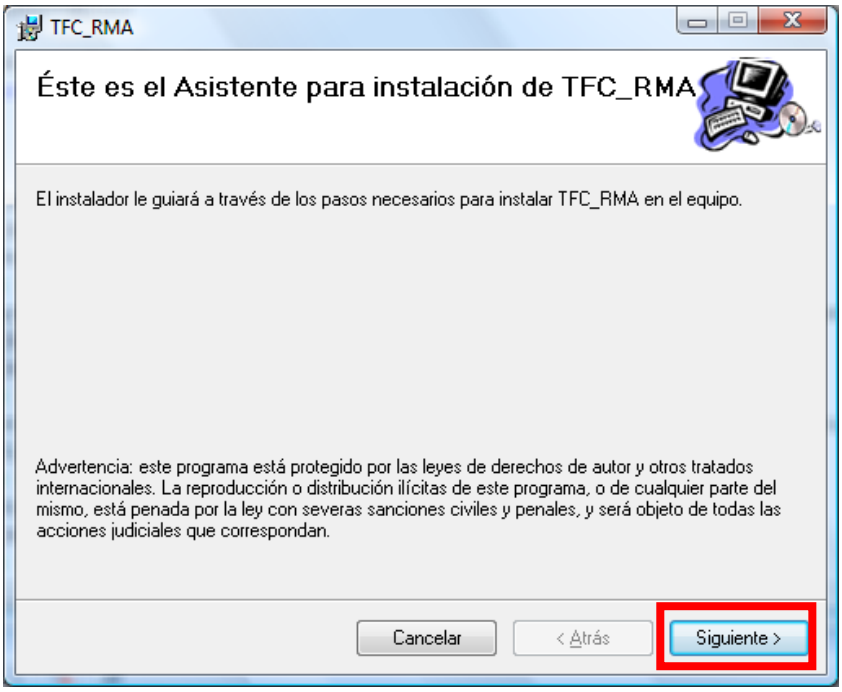


Ilustración 45 Paso 1 de la instalación

2) El programa de instalación nos dejará escoger donde queremos instalar el programa los valores que se muestran por defecto son válidos pero si el usuario ve conveniente puede cambiarlos. Clicamos siguiente como muestra la ilustración 46:

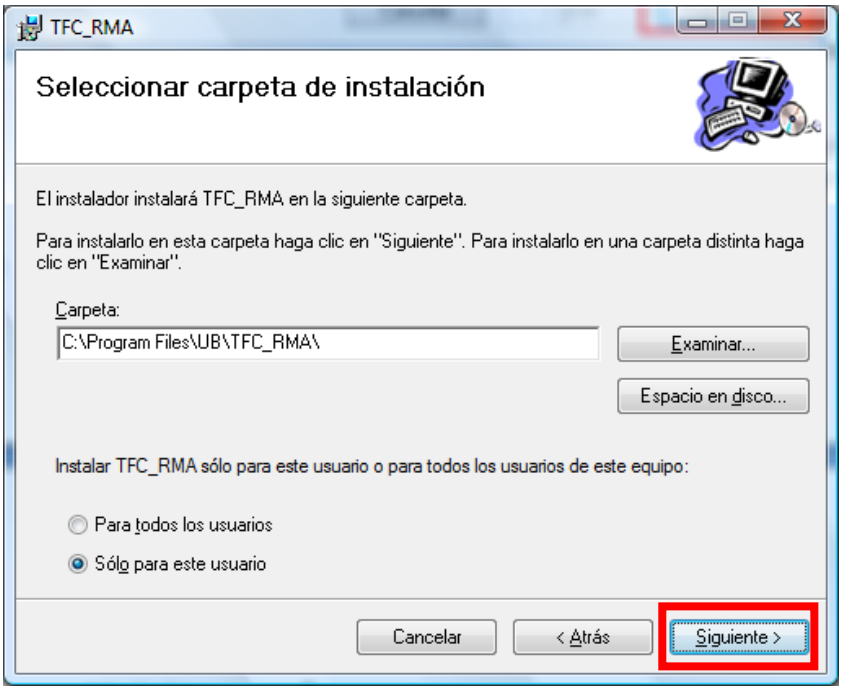


Ilustración 46 Paso 2 de la instalación

3) Si estamos conformes con todos los pasos ejecutados clicamos a siguiente para que el asistente empiece con la instalación de los archivos necesarios para la correcta ejecución del programa tal y como se muestra en la ilustración 47.

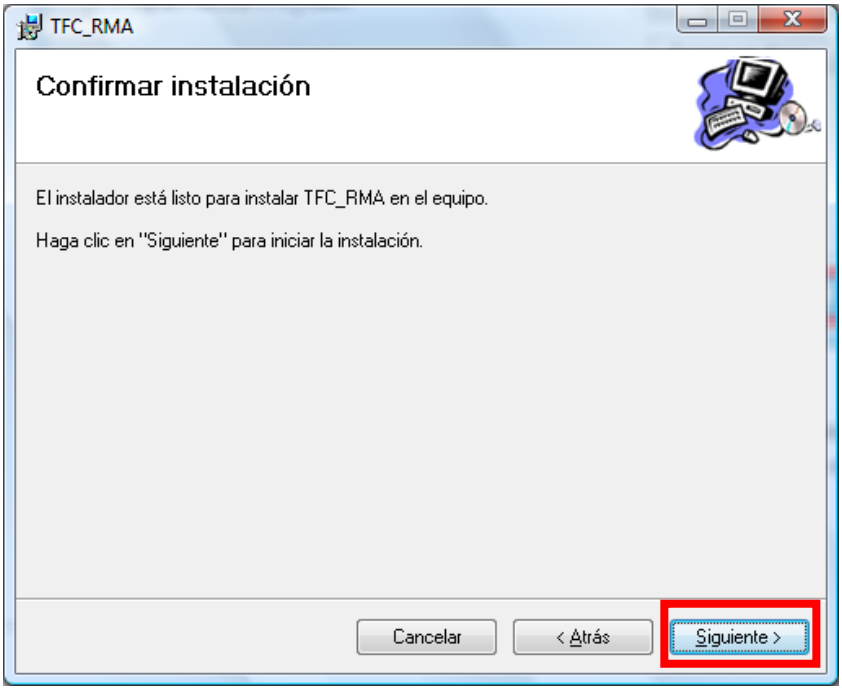


Ilustración 47 Paso 3 de la instalación

4) Dé los permisos necesarios para que el programa pueda seguir con la instalación.

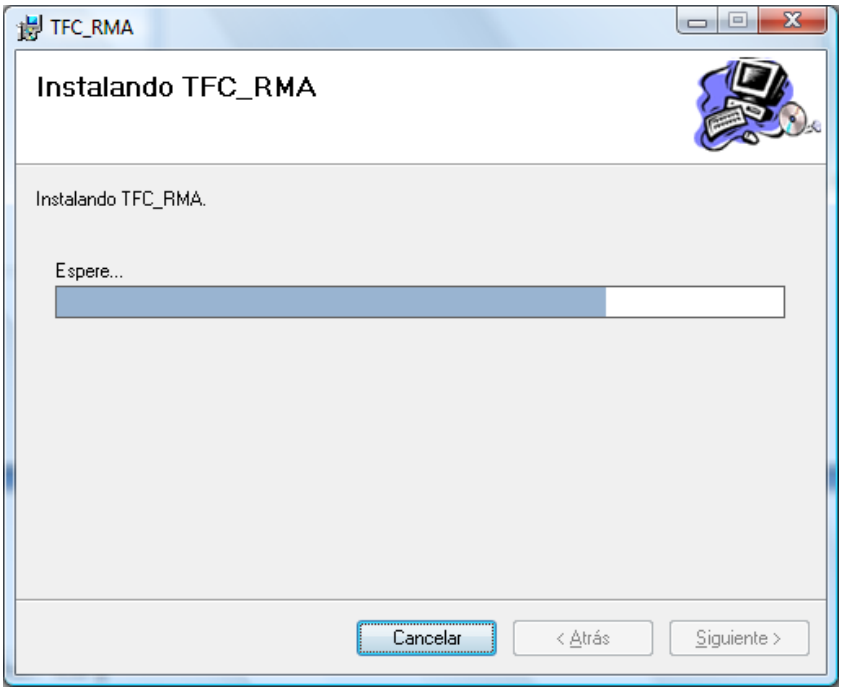


Ilustración 48 Paso 4 de la instalación

5) Si todo ha ido bien deberá poder ver una pantalla como se muestra en la ilustración 49, indicándole que la instalación ha finalizado correctamente. Ya puede pulsar cerrar como se muestra en la ilustración 49:

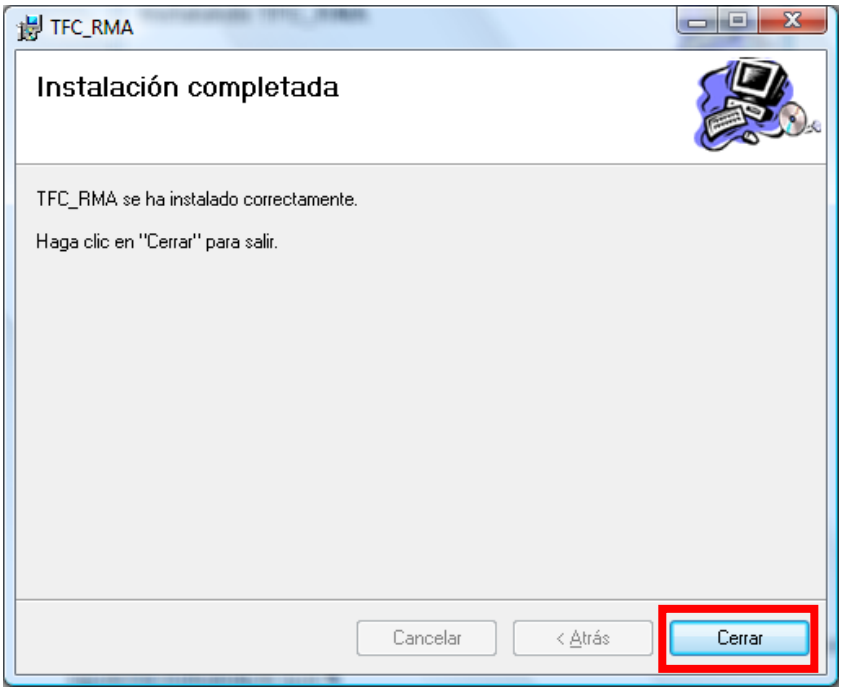


Ilustración 49 Paso 5 de la instalación

En la carpeta C:\Program Files\UB\TFC_RMA podrá encontrar todos los archivos de la instalación y en el escritorio un acceso directo al programa de reconocimiento de modelos de avión.

6.1.2.- Ejecución y descripción de TFC_RMA

Si está leyendo este apartado sin haber instalado el programa por favor lea el apartado 6.1.1 de este manual de usuario para la instalación del programa. Si ha instalado correctamente el programa usted está listo para ejecutarlo. Diríjase a su escritorio donde podrá encontrar un acceso directo bajo el nombre *TFC_RMA.exe*, dé doble clic sobre el icono y espere a que aparezca una pantalla como la que se muestra en la ilustración 50:

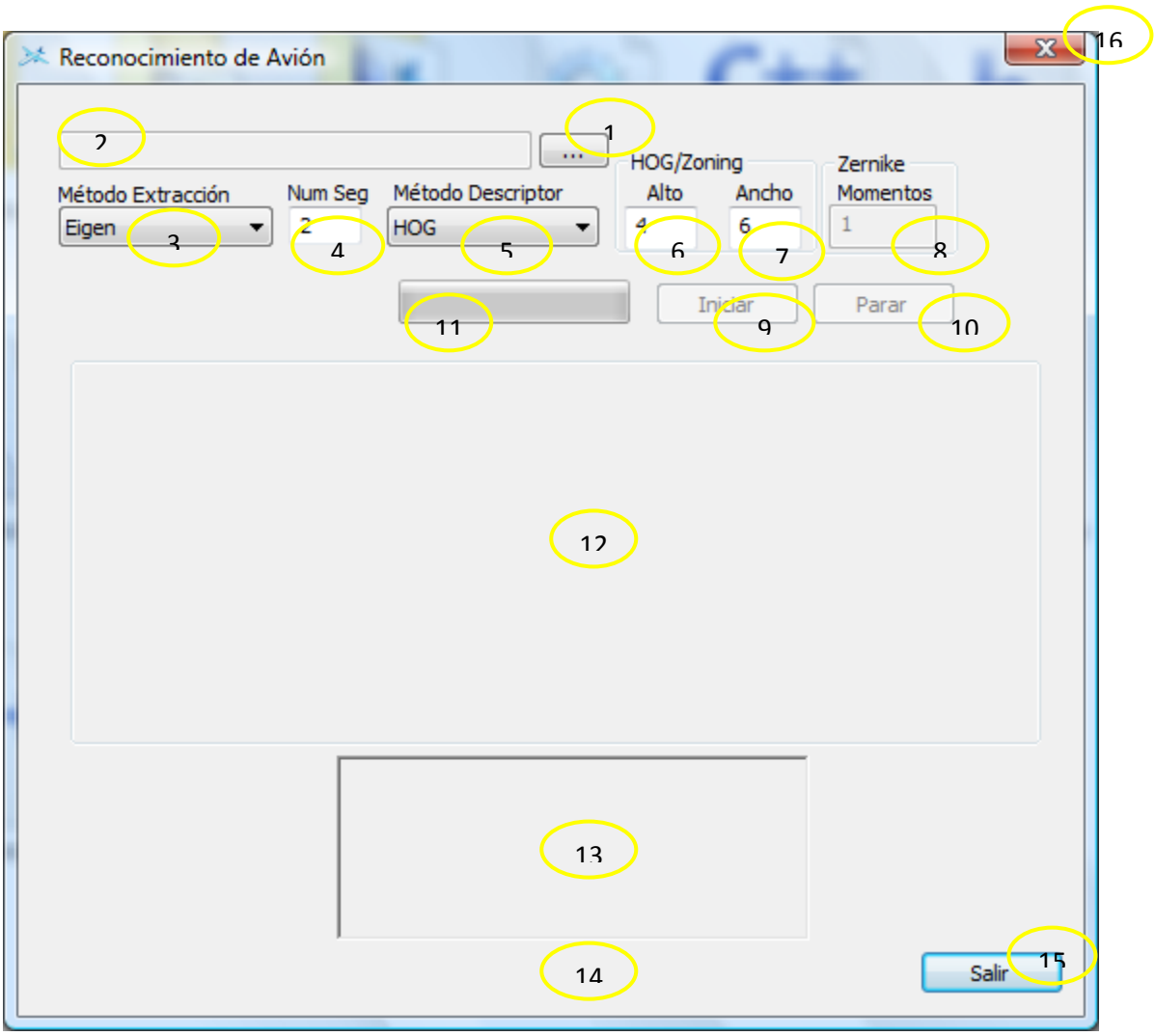


Ilustración 50 Pantalla principal del programa

A continuación se pasa a describir las diferentes opciones que ofrece el programa, la numeración que se detalla a continuación se corresponde con la numeración que aparece en la ilustración anterior.

(1) Éste es el botón de navegación que le permite desplazarse por el sistema de archivos de su sistema para seleccionar un video o una foto.
(2) En éste ventana de texto se muestra la ruta escogida con el botón (1).
(3) En ésta lista desplegable puede escoger entre los distintos métodos para la extracción de la imagen a partir de un video seleccionado con el botón (1).
(4) En éste cuadro de texto usted puede indicar en qué segundo desea que el programa extraiga la imagen. Los segundos se descuentan desde el final del video hacia el principio.
(5) En ésta lista desplegable usted puede escoger entre los distintos métodos descriptores de la imagen a evaluar.
(6) (7) Éstos dos cuadros de texto aparecerán editables únicamente cuando se haya escogido como método descriptor HOG o Zoning, en caso contrato aparecerán como no editables. Cada uno indica el número de cuadros en los que usted desea que la imagen se trocee tanto de alto como de ancho respectivamente.
(8) Éste cuadro de texto aparecerá editable únicamente cuando se haya escogido como método descriptor Zernike, en caso contrario aparecerá como no editable. Con éste valor usted puede indicar cuantos momentos desea que el programa aplique a la imagen.
(9) Con éste botón puede iniciar el proceso de evaluación. Éste botón aparecerá deshabilitado hasta que no introduzca una ruta de video o foto válida.
(10) Con éste botón usted puede parar la ejecución del programa manualmente. Éste botón aparecerá deshabilitado hasta que no presione el botón iniciar (9).
(11) Se trata de la barra de estado y le indica el progreso del programa y en qué punto hasta el final se encuentra.
(12) En la pantalla de visualización podrá ver el video seleccionado a partir del botón de navegación (1).
(13) Desde esta ventana se muestra la imagen en estado de evaluación y la imagen del modelo de avión resultante de la evaluación.
(14) Desde ésta ventana de texto usted podrá observar el dialogo informativo de lo que va ocurriendo durante la ejecución del programa.
(15) (16) Con éstos botones usted puede salir de la aplicación.

6.1.3.- Funcionamiento de TFC_RMA

6.1.3.1.- Programa de creación de clasificadores para operario especializado

A continuación se detalla los pasos a seguir para crear los clasificadores necesarios para poder iniciar el proceso de reconocimiento. Cabe decir que este proceso se deberá realizar en todos los equipos donde se instale por primera vez el programa, pudiéndose cambiar los clasificadores por otros más especializados o realizados con más muestras, siempre que se quiera, y antes de ejecutar la evaluación.

Prerrequisitos antes de empezar a crear los clasificadores:

- Se deberá tener instaladas correctamente las librerías OpenCV y Blobs véase punto 6.1.1 para información de cómo instalarlas.
- Se deberá tener en la carpeta raíz donde se encentre el ejecutable, una carpeta llamada "fotos" donde dentro se encuentren las fotos a clasificar. El asistente de instalación crea las carpeta pero sin contenido.
- Se deberá tener en la carpeta raíz donde se encuentre el ejecutable, una carpeta llamada "boost" donde se guardaran los clasificadores. El asistente de instalación crea esta carpeta y otra carpeta llamada "boost de ejemplo", con clasificadores de ejemplo para 4 modelos de avión.
- Se deberá tener instalado en el equipo Matlab para su ejecución, ya que parte del proceso se realiza con este programa.

A continuación se detalla paso a paso como generar los clasificadores:

1) Diríjase al directorio raíz de la instalación, por defecto "C:\Program Files\UB\TFC_RMA" y ejecute el programa *TFC_Clasificadore.exe*.
2) Una vez ejecutado deberá ver una pantalla como se muestra en la siguiente ilustración, seleccione el método descriptor introduciendo el número correspondiente y presione "Entrar" de su teclado. Para éste ejemplo se va a utilizar como método descriptor HOG que se corresponde con el 1 como se muestra en la ilustración 51.

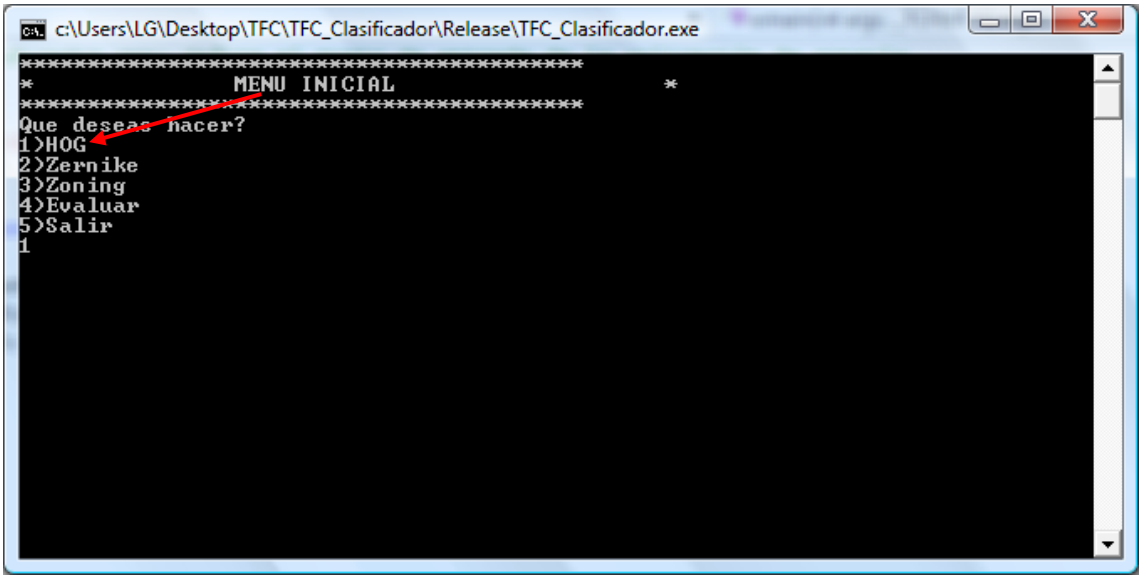


Ilustración 51 Paso 2 ejemplo clasificador

El programa le pedirá en cuantos trozos desea recortar la imagen para analizar cada trozo por separado. En este ejemplo elegiremos 4 y 6 para la altura y el ancho respectivamente.

3) Cuando haya introducido los datos y presionado “Entrar” el programa empezará a generar los descriptores dentro de la carpeta “descriptores” dentro de la carpeta raíz como se muestra en la ilustración 52.

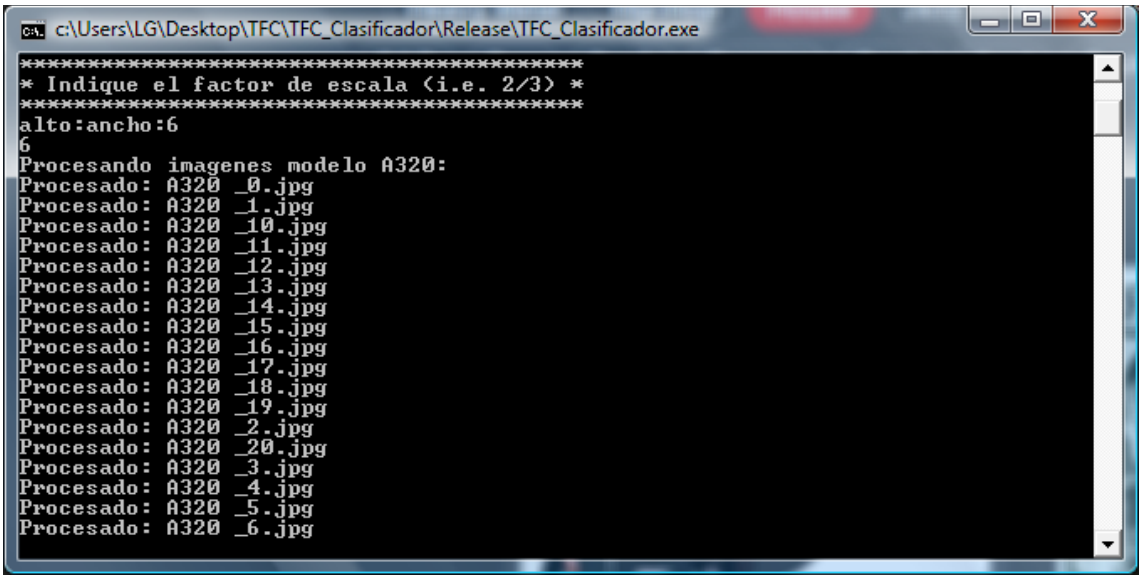


Ilustración 52 Paso 3 ejemplo clasificador

4) Cuando el proceso haya finalizado presione “Entrar” y salga de la aplicación introduciendo un 5 más “Entrar” como se muestra a continuación en la ilustración 53.

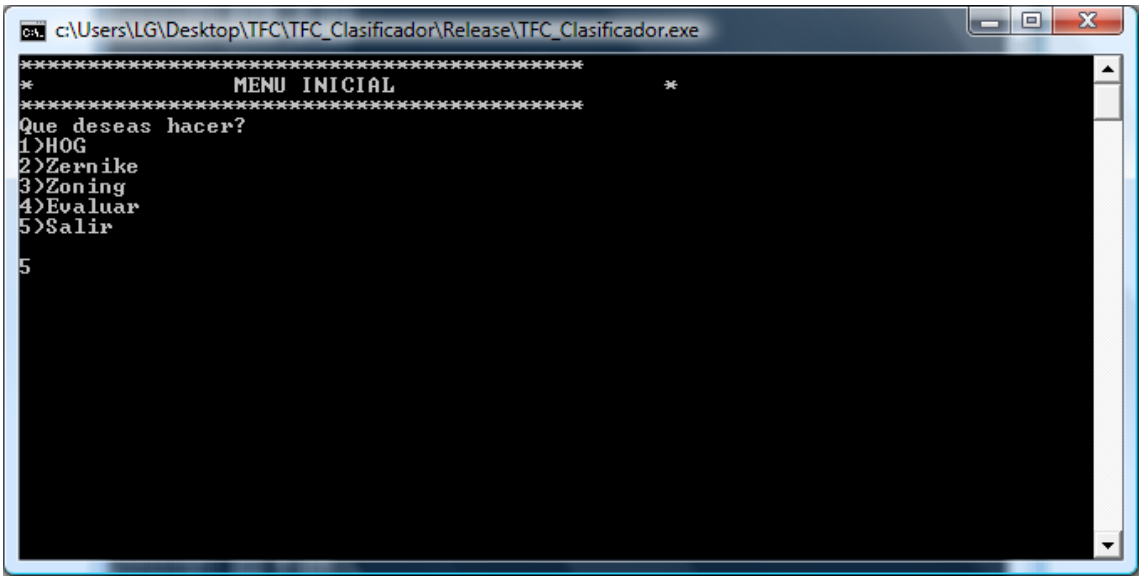


Ilustración 53 Paso 4 ejemplo clasificador

5) Ejecute el programa Matlab desde donde lo tenga instalado y cambie el directorio de trabajo al directorio raíz donde tenga instalado el programa *TFC_RMA* de forma estándar la ruta sería "C:\Program Files\UB\TFC_RMA", como se indica en la ilustración 54.

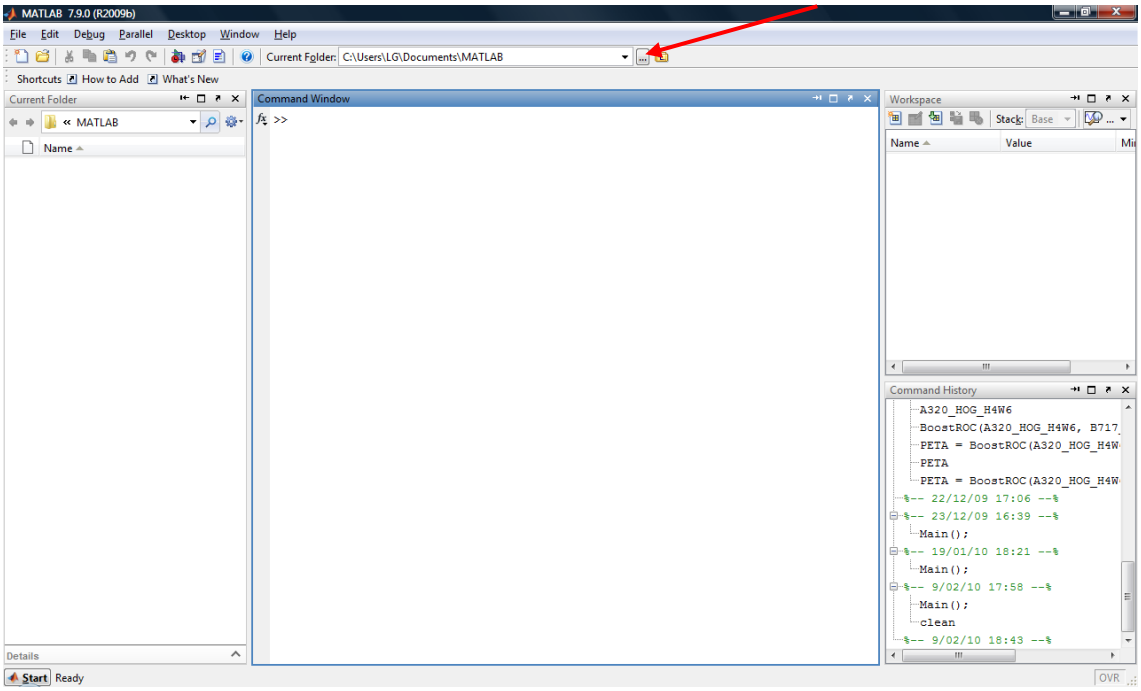


Ilustración 54 Paso 5 ejemplo clasificador

6) Si ha cambiado correctamente el directorio actual en la columna de la izquierda deberán aparecer todos los archivos contenidos en la carpeta donde está contenido el programa y entre ellos el archivo Main.m, como se muestra en la ilustración 55.

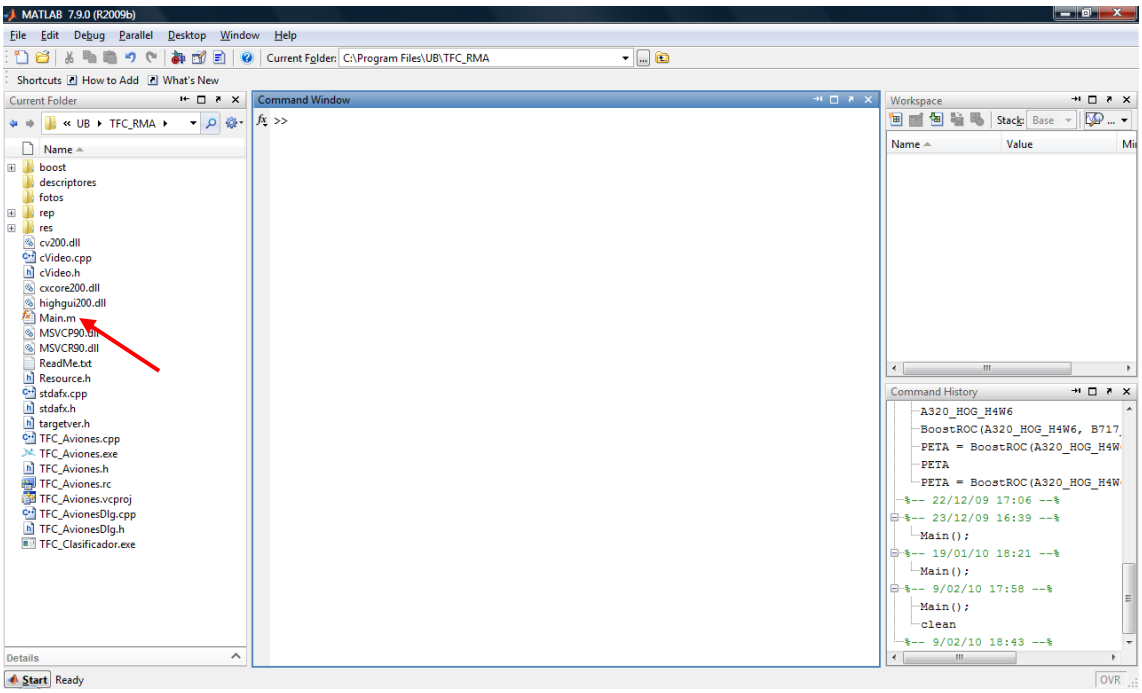


Ilustración 55 Paso 6 ejemplo clasificador

7) Ahora únicamente escriba *Main();* en la ventana de comandos y se generaran automáticamente los clasificadores en la carpeta *boost, como se puede observar en la ilustración 56*.

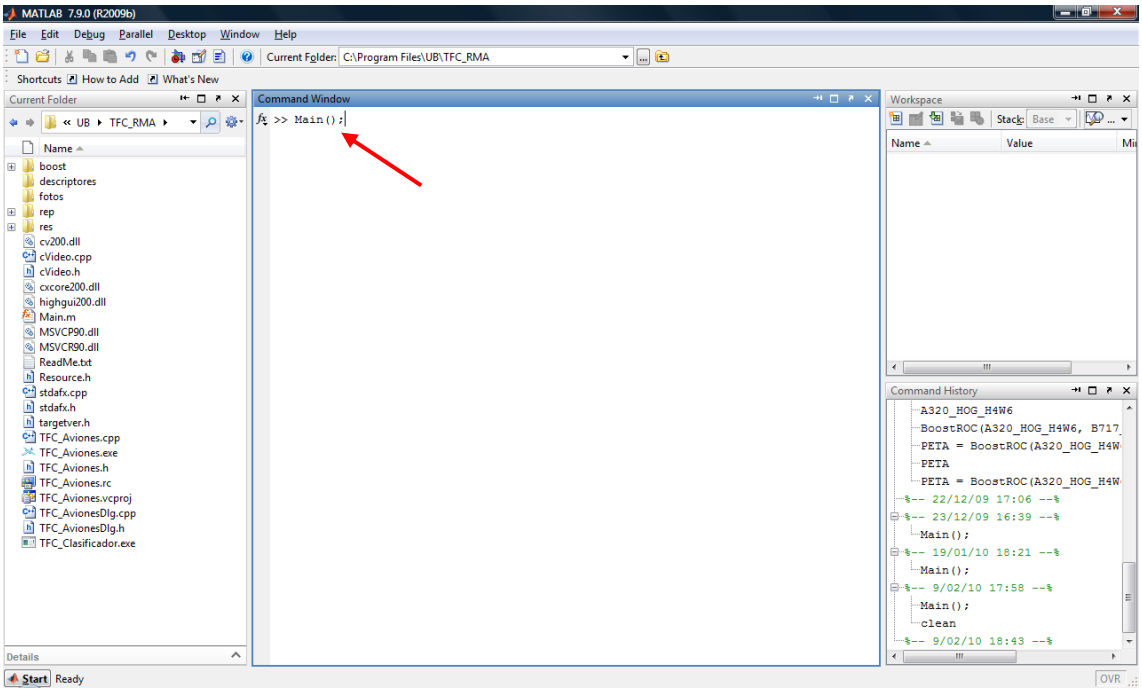


Ilustración 56 Paso 6 ejemplo clasificador

6.1.3.2.-Programa de evaluación para operario a pie de pista

Para poder mostrar el funcionamiento del programa se ejecutara a modo de ejemplo la detección del modelo A320.

Prerrequisitos para poder seguir el ejemplo:

- Tener instaladas correctamente las librerías OpenCV y Blobs véase punto 6.1.1 para información de cómo instalarlas.
- Tener instalado el programa *TFC_RMA*, véase punto 6.1.1 para información de cómo instalarlo.
- Tener un video en formato ".avi" del modelo de avión A320.

A continuación se detalla paso a paso como realizar la detección del modelo A320 con el programa de reconocimiento de modelos de avión.

1) Ejecutar desde el escritorio del sistema el archivo *TFC_RMA.exe* dando doble clic encima del icono.
2) Introduzca la ruta donde se encuentra el video utilizando el botón de navegación, tal y como se indica en la ilustración 57.

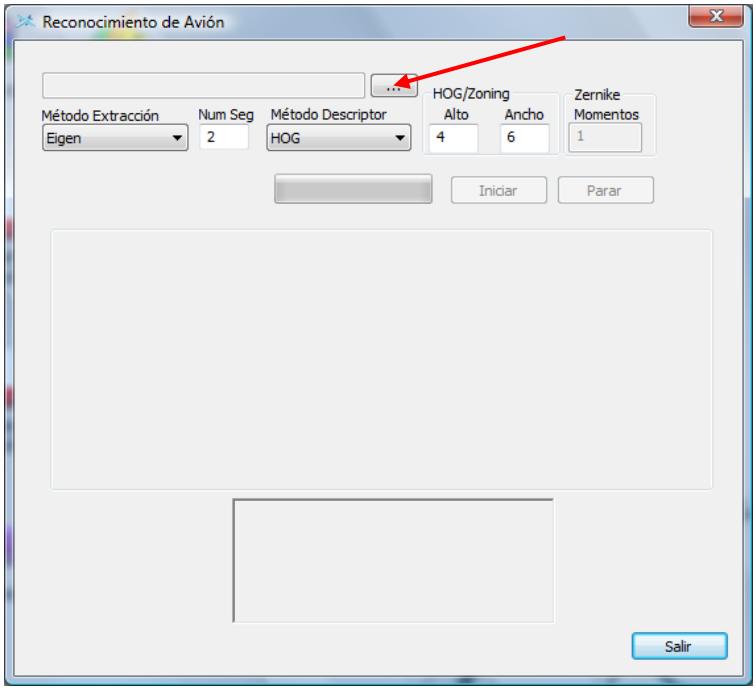


Ilustración 57 Paso 2 ejemplo evaluación

3) Seleccione el método de extracción que prefiera, en éste ejemplo se seleccionará el método *Adaptative*, y 2 segundos como instante donde queremos extraer la imagen del vídeo antes de que termine. En la ilustración 58 se puede observar lo explicado anteriormente

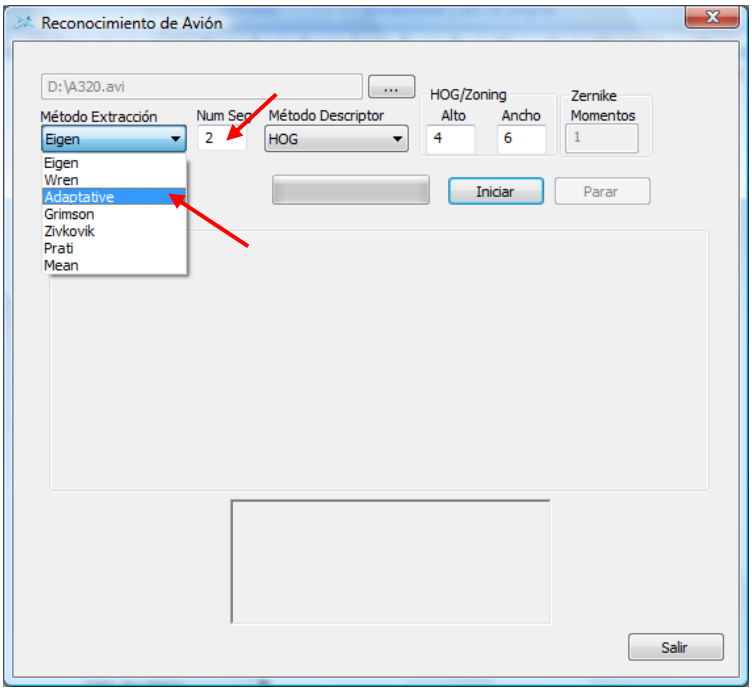


Ilustración 58 Paso 3 ejemplo evaluación

4) Seleccione el método descriptor que prefiera para la imagen, en este ejemplo se seleccionará el método HOG con una altura de 4 y un ancho de 6. Como se muestra en la ilustración 59

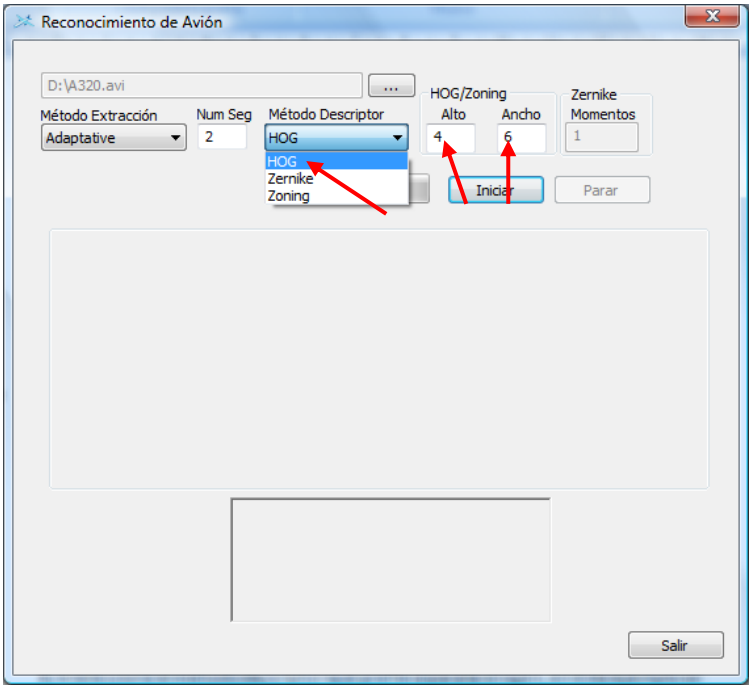


Ilustración 59 Paso 4 ejemplo evaluación

5) Presione el botón *Inicar*, tal y como se indica en la ilustración 60, para iniciar el proceso de evaluación.

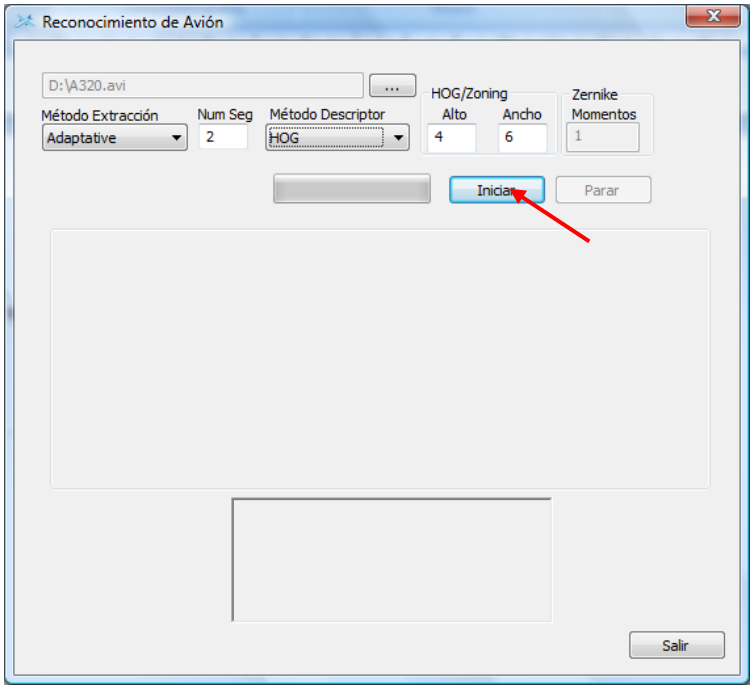


Ilustración 60 Paso 5 ejemplo evaluación

6) Al acabar el proceso de reconocimiento se le mostrará una fotografía del modelo coincidente y su nombre, como se puede observar en la ilustración 61.

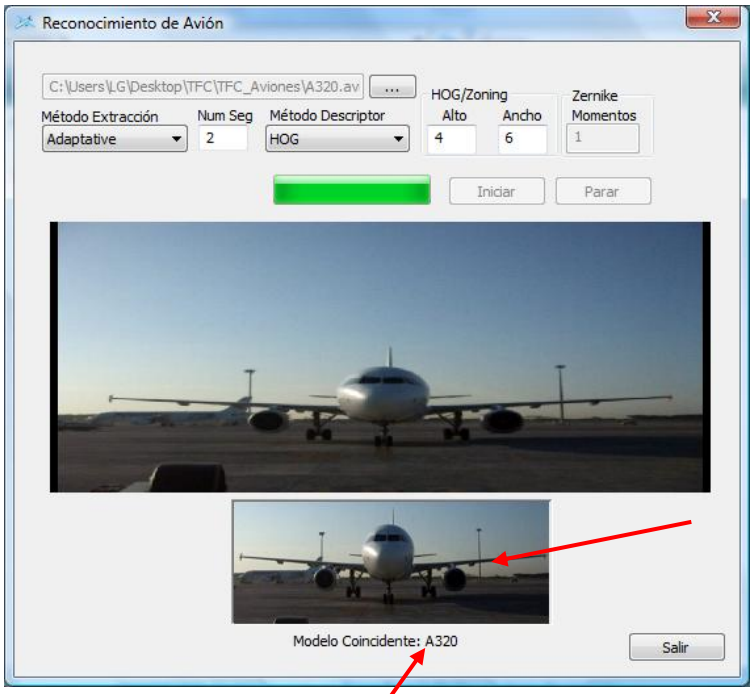


Ilustración 61 Paso 6 ejemplo evaluación

Si quiere volver a iniciar el proceso de reconocimiento aunque se trate del mismo vídeo en un segundo diferente deberá empezar desde el paso 2 de esta guía de funcionamiento.

6.1.4.- Como incluir nuevos métodos

Este aparatado es de interés para todos aquellos que decidan ampliar este proyecto con nuevos métodos de extracción y de descripción de imágenes.

Este apartado del manual está compuesto por 2 partes correspondientes a las interfaces del programa de reconocimiento y el programa de creación de clasificadores y a la librería que utilizan tanto la parte de creación de clasificadores como el programa de reconocimiento.

6.1.4.1.- Programa de reconocimiento

Para empezar se describirá la parte de interfaz del programa de reconocimiento, en ésta se encuentran 2 listas desplegables correspondientes al método de extracción y la otra a los métodos descriptores como muestra la ilustración 62.

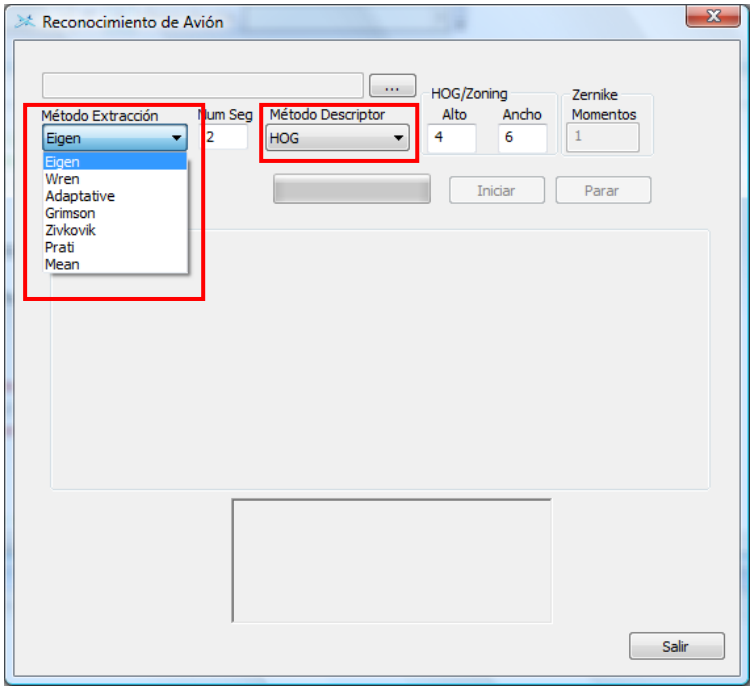


Ilustración 62 Listas desplegables de la interfaz

Para que en las listas desplegables aparezcan los nuevos métodos se deben los siguientes pasos:

Desde el proyecto *TFC_Aviones* diríjase a la Vista de recursos como se indica en la ilustración 63 y de doble clic sobre *IDD_TFC_AVIONES_DIALOG* desde aquí podrá ver la interfaz tal y como se mostrará por pantalla. Ahora póngase con el cursor encima de la lista desplegable que desee y de clic derecho + Propiedades. En la nueva ventana que aparecerá en el apartado *Data* puede introducir los nuevos métodos. Estos pasos son iguales para las dos listas desplegables.

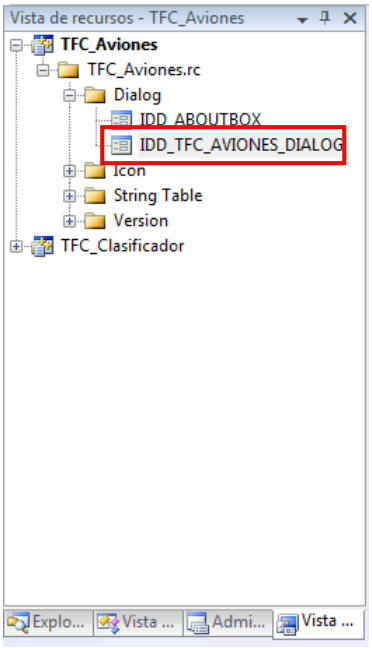


Ilustración 63 Vista de recursos

En los siguientes subapartados se explicará en que funciones del programa de reconocimiento es necesario introducir el nuevo código para codificar los nuevos métodos.

- Método de Extracción

Desde el proyecto *TFC_Aviones* diríjase a la clase *TFC_AvionesDlg -> selMetodo()*, dentro de esta función se encuentra la codificación de los métodos de extracción. Es importante no cambiar el orden si no se es consecuente del cambio. Se puede empezar a añadir métodos a partir del número 8. Sería adecuado conservar la misma numeración para todas las partes del programa como en la parte de la librería.

- Método Descriptor

Desde el proyecto *TFC_Aviones* diríjase a la clase *TFC_AvionesDlg -> OnCbnSelchangemdescriptor()*, dentro de esta función se encuentra la codificación de los métodos descriptores. Es importante no cambiar el orden si no se es consecuente del cambio. Se puede empezar a añadir métodos a partir del 4. Sería adecuado conservar la misma numeración para todas las partes del programa como en la parte de la librería.

6.1.4.1.- Programa de creación de clasificadores

Para este programa se deberán modificar los menús indicando que existe un método descriptor más y parte del código correspondiente a la librería compartida *TFC_Base*.

Desde el proyecto *TFC_Base* diríjase a la función *TFC_Base.cpp -> mainProgress()*, aquí se deberá modificar la función *menuInicial()* y el *switch(opcion)* agregando el nuevo método.

6.1.4.1.- Librería compartida

Desde el proyecto *TFC_Base* diríjase a la función *TFC_Base.cpp -> obtenerDescriptores()* se deberá modificar la función abrirFichero(), el *switch(opcion),* agregar en la clase imagen el nuevo método, y la función guardarImagen().

6.2.- Manual Sistema de Guía de Atraque

En este punto del manual se describe de una forma más detallada la labor a realizar por el operador a pie de pista, parte de la nomenclatura utilizada y su entorno de trabajo. Toda la información ha sido extraída del Manual de Operaciones y Mantenimiento del Sistema de Guía de Atraque Safedock® que ha sido compilado con el fin de permitir al usuario el operar y mantener el sistema de un modo efectivo y seguro.

6.2.1.- Glosario de Términos

Estado activo	El SGA está escaneando el área de atraque para la aeronave que se aproxima.
Unidad de Presentación alfanumérica/Unidad de Presentación de texto	La parte superior de la unidad de presentación LED. Se usa como unidad de presentación de texto capaz de mostrar dos líneas de caracteres alfanuméricos o un línea de caracteres de doble altura
Programa de Mantenimiento del SGA	Es el software que se ejecuta en el SGA y para realizar la calibración y configuración del mismo. Es lo mismo que 'Stand Configuration Utility'
Captura/ Modo de captura	El SGA está escaneando la zona de aparcamiento de una aeronave que se aproxima
Unidad de Presentación de Índice de Aproximación	Se trata de la parte inferior de la unidad de presentación LED. Se utiliza para indicar la distancia que le queda a la aeronave que realiza la secuencia de atraque para llegar al punto de parada
Unidad de Control	Centro de proceso del sistema SGA. Está montada en el mismo armario de la Unidad de Presentación
Sistema de Guía de Atraque	Se trata de un sistema que proporciona información visual al piloto y/o copiloto referente a la posición de la aeronave con respecto a la línea central y al punto de parada. Actúa como una ayuda para el piloto a la hora de maniobrar la aeronave para alcanzar el punto de atraque correcto.
Secuencia de atraque	Es el procedimiento mediante el cual un SGA dirige a una aeronave entrante hasta un punto deparada predeterminado. Consiste en una selección de aeronave por el operador, un auto-chequeo del sistema, un control de calibración, captura, seguimiento e identificación.
Determinador de Distancia de Láser/ Unidad Láser	Es un instrumento compuesto de un determinador de distancia de láser y dos espejos montados sobre motores de pasos. Utiliza los dos espejos para dirigir los pulsos del láser dese el determinador de distancia de láser en dos dimensiones, resultando un escaneado tridimensional del área del punto de estacionamiento.
Unidad de presentación LED	El SGA utiliza una unidad de presentación consistente en varios módulos de 8x8 puntos LED (diodos emisores de luz) para mostrar a los pilotos texto, información de acimut e información del índice de aproximación.

6.2.2.-Descripción del sistema

Incluye una descripción general y una descripción técnica detallada de las partes que constituyen el sistema. Este apartado proporciona al usuario una buena descripción de la arquitectura del sistema y de los componentes que la forman.

6.2.2.1.- Breve arquitectura del sistema

La ilustración 64 es una vista general del sistema SAFEDOCK o SGA, Sistema de Guía de Atraque. La siguiente sección es una breve descripción del sistema SGA y sus sub-unidades.

(a) La unidad principal del SGA consiste en una unidad de presentación en tiempo real formada por LEDS, una Unidad de Control y una Unidad de Escaneado por Láser todo ello alojado en el interior del mismo armario.
(b) El sistema también incluye un Panel de Operado, que tiene una pantalla LCD y un pulsador de parada de emergencia. El Panel de Operador se instala en el Puesto de Control de la Pasarela y/o a nivel del estacionamiento.
(c) La unidad de escaneado por láser está basada en tecnología 3-D para efectuar la secuencia de atraque de modo seguro cuando una aeronave se aproxima a un punto de estacionamiento del terminal. El Determinador de Distancia por Láser transmite a la unidad de control para su procesamiento datos de la distancia desde la aeronave que se aproxima o se aleja.
(d) La unidad de control transmite los resultados de los datos procesados para ser mostrados en la Unidad Presentación y en el Panel de Operador, y, opcionalmente, al Ordenador Central del Sistema de de Atraque, GOS.

6.2.2.2.- Sub-unidades del SGA

A continuación se detallan las sub-unidades que componen el Sistema de Guía de Atraque.

<u>- Unidad de Presentación</u>

- La unidad de presentación incorpora tres indicadores distintos para información alfanumérica, acimut y de índice de aproximación, claramente visible para ambos pilotos en la cabina de la aeronave. La unidad de Presentación comprende, según modelo, de 9 a 15 módulos de 8x8 puntos indicadores LED, contando cada uno con su propia unidad procesadora conectada en serie a una Unidad Central por medio de un cable de cinta. Se utiliza un protocolo de comunicación en serie para la comunicación entre unidad de control y los módulos LED. Las dos filas superiores se utilizan para información alfanumérica, la tercera fila para información acimut y la columna vertical para información de índice de aproximación.

<u>- Unidad de Control</u>

- La unidad de Control, situada con, o dentro de la Unidad de Presentación, comprende una tarjeta madre con una unidad de microprocesador PC 104/38 y dos tarjetas de control de los motores por pasos. La Unidad de Control continuamente monitoriza y controla la operación del Sistema de Guía de Atraque.

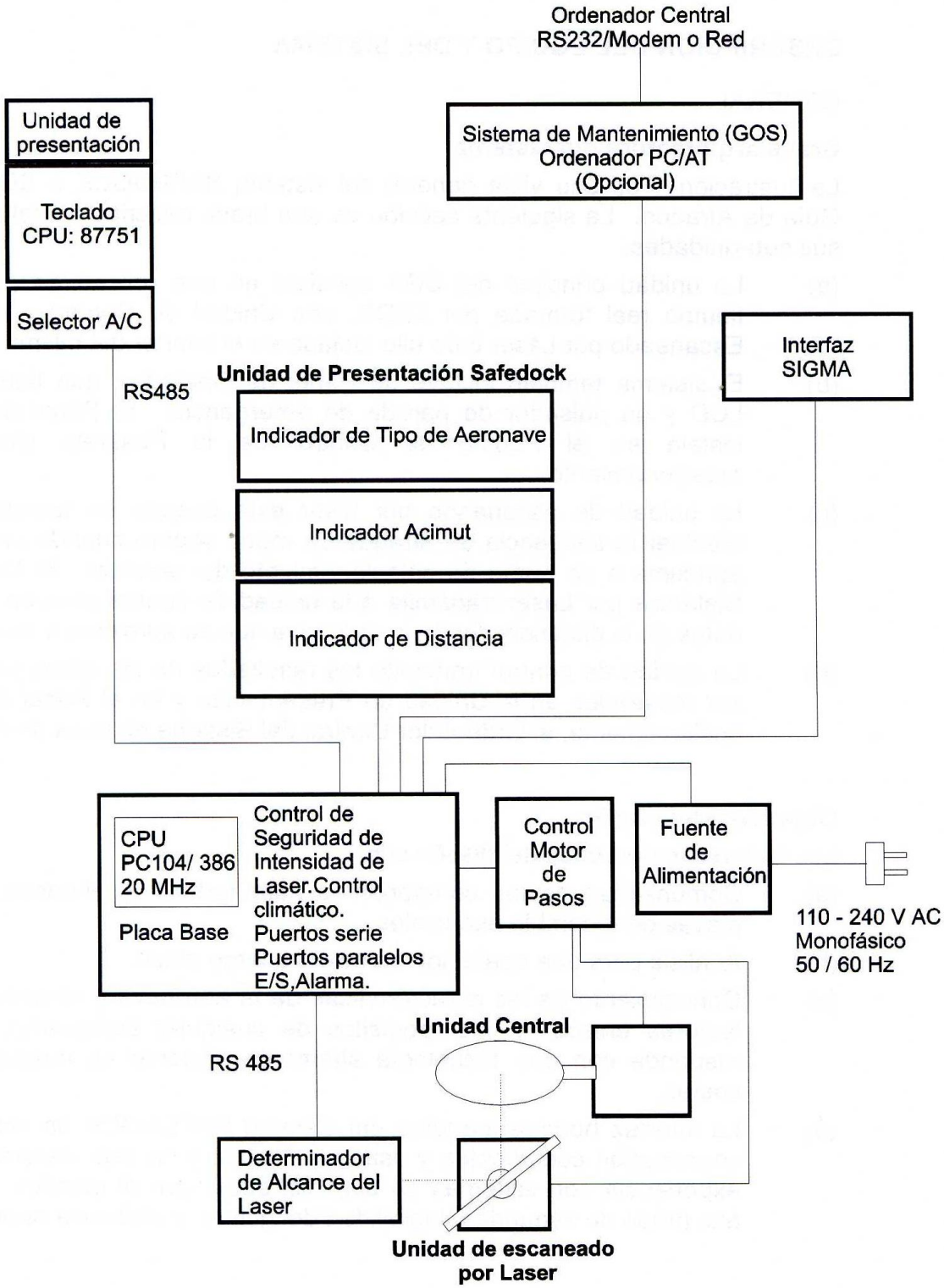


Ilustración 64 Vista general del SGA

- Unidad de Escaneado por Laser

- La Unidad de Escaneado por Láser, ilustración 65, comprende las siguientes sub-unidades:

(a) Determinador de Distancia por Láser.

(b) Espejo de escaneado vertical y motor por pasos.

(c) Espejo de escaneado horizontal y motor por pasos.

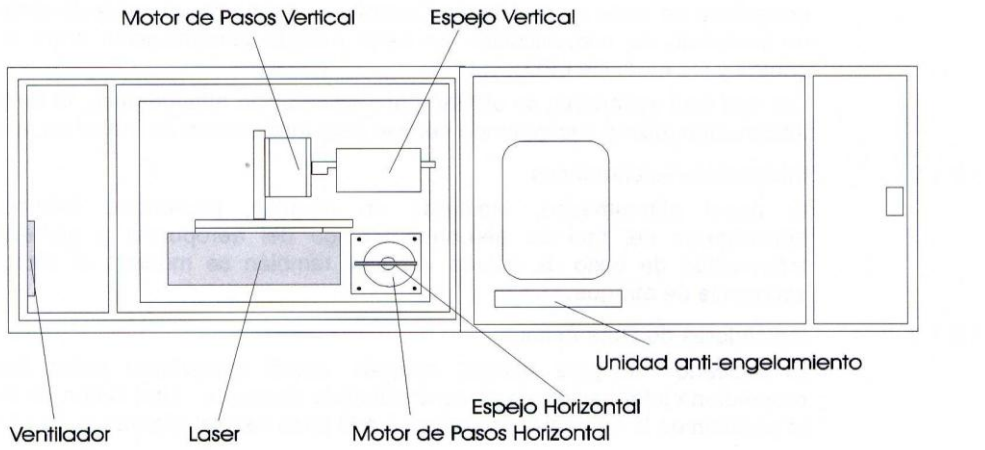


Ilustración 65 Unidad de Escaneado por Láser SAFEDOCK

Cuando la secuencia de atraque ha sido iniciada, el Determinador de Distancia por Láser transmite pulsos infrarrojos, escaneando tanto horizontal como verticalmente mediante un sistema de espejos controlados por n motor por pasos, para detectar una aeronave que se aproxima.

La distancia a la aeronave se determina en función del tiempo requerido por el pulso infrarrojo para ser transmitido y recibido por el Determinador de Distancia por Láser. La Unidad de Control, que controla la posición de los espejos de escaneado, utiliza los valores de distancia obtenidos desde el Láser para determinar la localización de la aeronave con respecto a su punto de parada.

- Panel de Operador

- El Panel de Operador, ilustración 66, es utilizado para controlar el Sistema de Guía de Atraque desde un lugar cercano o en la pasarela de pasajeros.

El Panel de Operador consiste en un teclado con 30 botones pulsadores, y una pantalla de cristal líquido (LCD). La pantalla LCD indica los modos de operación del sistema y proporciona información de diagnostico y error. El panel de control del operador esta retroiluminado.

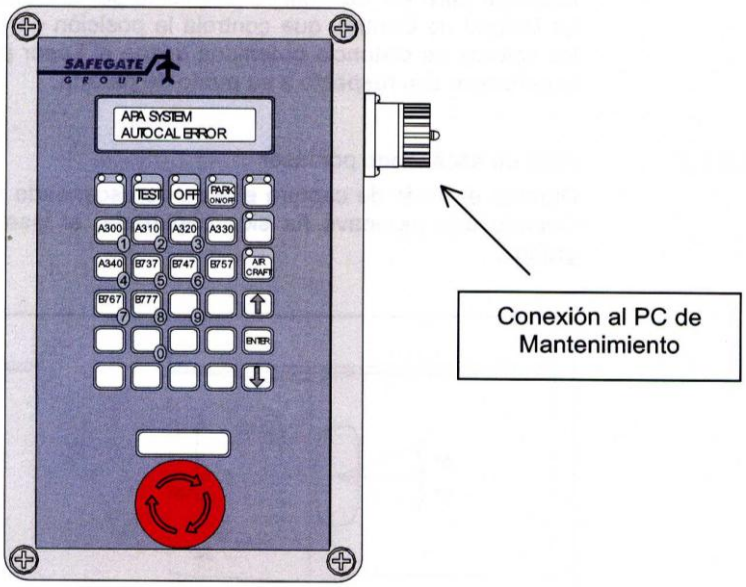


Ilustración 66 Panel de Operador SGA

6.2.2.3.- Operación de Atraque

- La secuencia de Atraque

Esta es una breve descripción de la operación del Safedock.

(a) El sistema individual SGA se opera desde el Panel de Operador situado en la pasarela o a nivel del estacionamiento o desde el GOS. Desde el GOS todos los sistemas conectados pueden ser controlados y monitorizados.

(b) En el sistema GOS un cierto número de tipos de aeronaves están definidos mediante un conjunto de parámetros como el perfil del morro, altura del morro y posiciones del motor. Al inicio de una secuencia de atraque el operador asigna un tipo de aeronave. Durante la secuencia de atraque los parámetros correspondientes a la aeronave actual son medidos por el equipo de láser. Los datos capturados son comparados con el perfil de seguridad de la aeronave asignada. Si los márgenes de seguridad no son correctos, el mensaje ID FAIL (identificación fallida) se mostrará al piloto, y la secuencia de atraque será abortada.

(c) El sistema en tres modos (ver abajo):
   a. Comprobación de calibración
   b. Captura
   c. Seguimiento

(d) La secuencia de atraque completo se muestra en la ilustración 67 en el diagrama de estado.

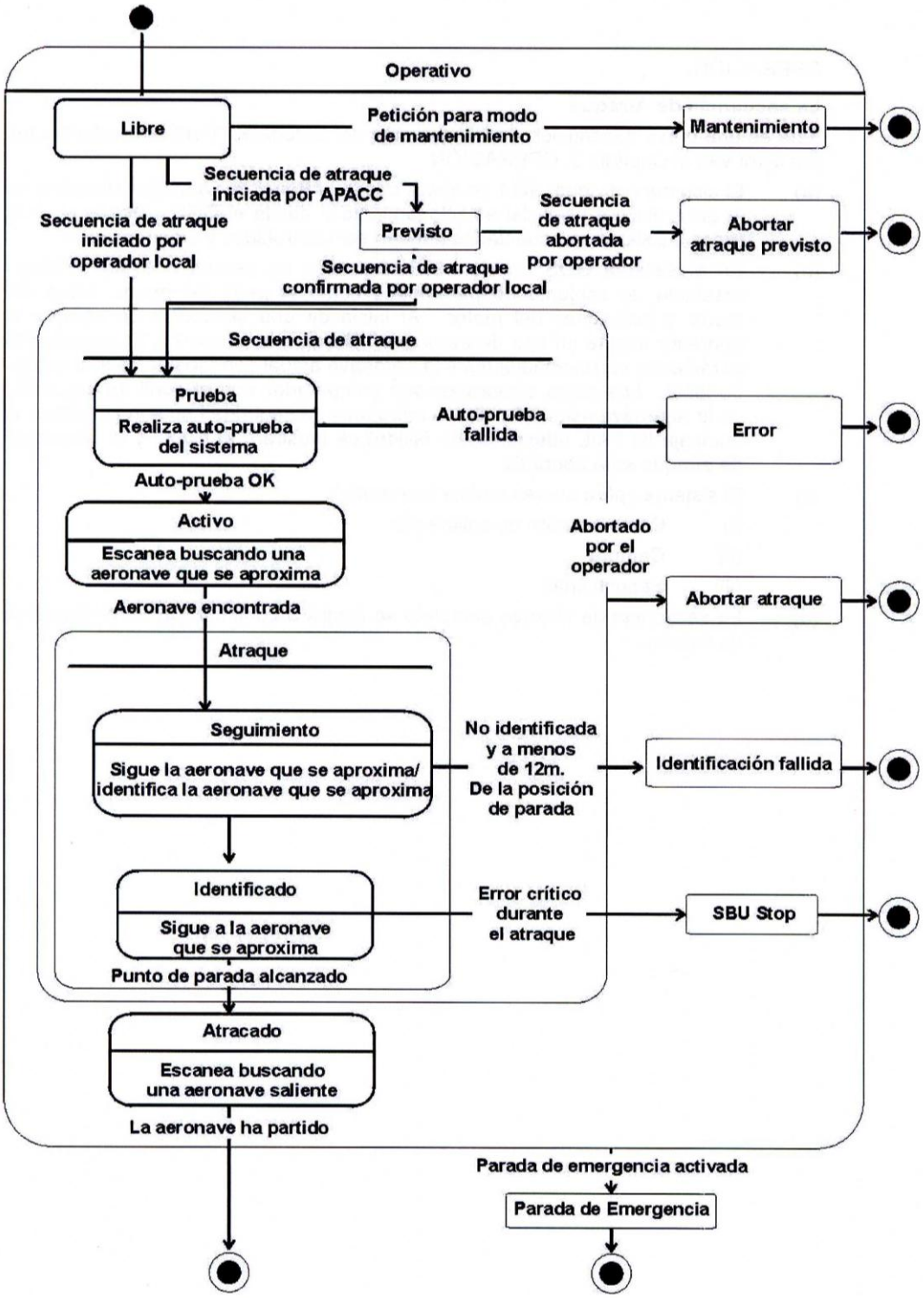
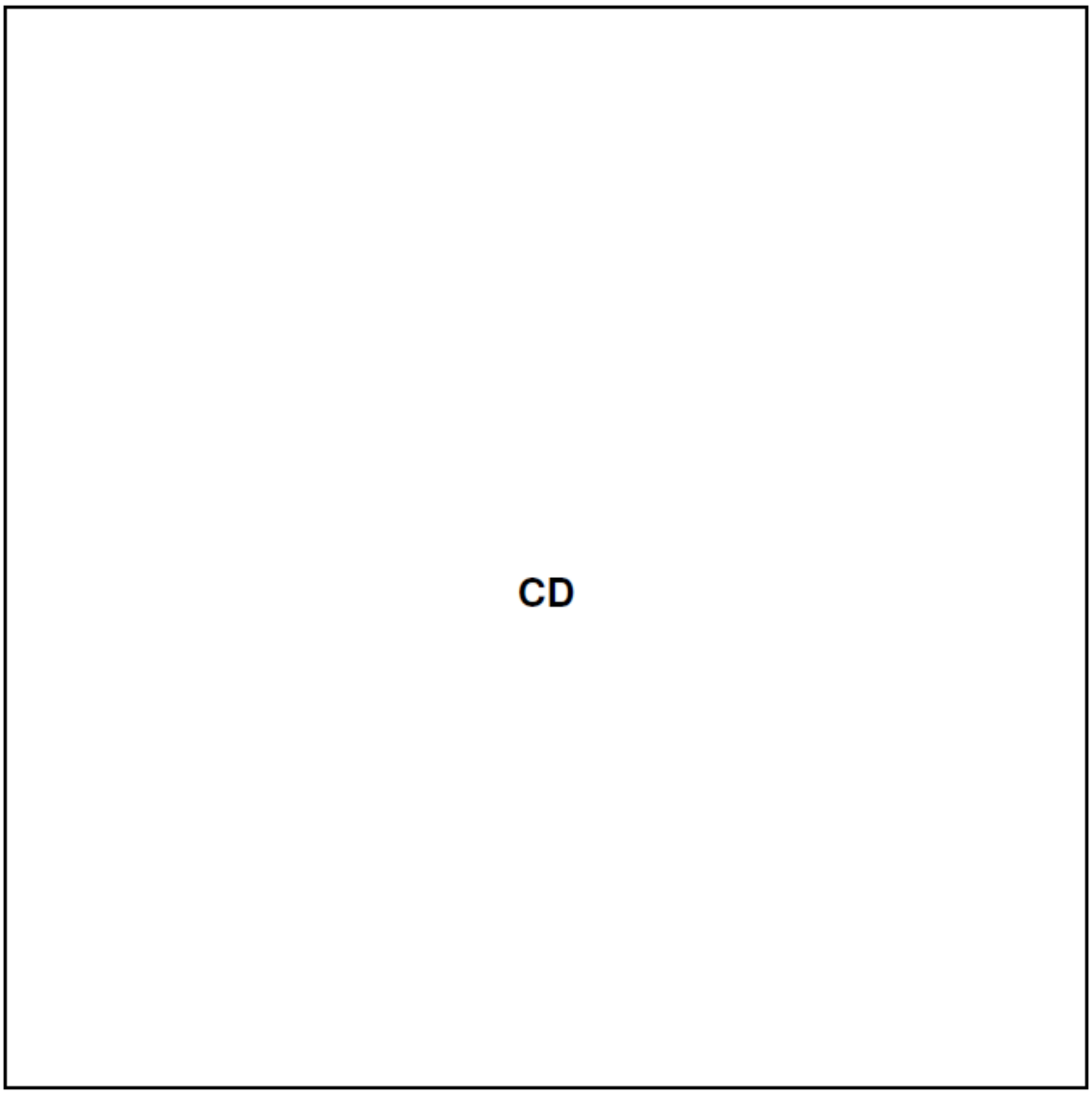


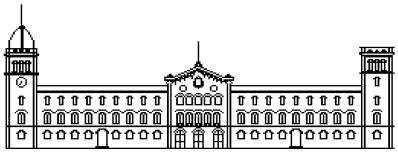
Ilustración 67 Secuencia de Atraque

6.3.- CD

El CD adjunto contiene las siguientes carpetas:

- Código fuente
    - Interfaz. Código de las 2 interfaces.
    - Librería. Código que se ha usado para generar la librería usada por la interfaz.
    - Setup. Código que se ha usado para generar el instalador.
- Setup. Archivos autoinstalables que contiene todos los ejecutables y datos de ejemplo para probar la aplicación.
- Documentación. Manuales de usuario y archivos de interés.
- Memoria del proyecto. Contiene la memoria del proyecto en distintos formatos.





ENGINYERIA TÈCNICA EN INFORMÀTICA DE SISTEMES

UNIVERSITAT DE BARCELONA

Treball fi de carrera presentat el dia de de 200
a la Facultat de Matemàtiques de la Universitat de Barcelona,
amb el següent tribunal:

Dr. President

Dr. Vocal 1

Dr. Secretari

Amb la qualificació de: