Trabajo Fin de Máster
Máster en Ingeniería Industrial

Entorno interactivo Matlab para análisis de texturas en imágenes II

Autor: Jesús Medrán Castro
Tutor: Manuel Ruíz Arahal

Dep. Ingeniería de Sistemas y Automatización
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Máster
Máster Universitario en Ingeniería Industrial

Entorno interactivo Matlab para análisis de texturas en imágenes II

Autor:
Jesús Medrán Castro

Tutor:
Manuel Ruíz Arahal
Profesor titular

Dep. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Trabajo final de Máster: Entorno interactivo Matlab para análisis de texturas en imágenes II

Autor: Jesús Medrán Castro

Tutor: Manuel Ruíz Arahal

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

Dedico el presente trabajo a todas aquellas personas que siempre me han ayudado, mi familia y amigos. Por estar siempre a mi lado y por su constante apoyo y consejos que he recibido. También agradecer a todos aquellos profesores por guiarme y enseñarme.

Jesús Medrán Castro

Alumno del Máster Universitario en Ingeniería Industrial

Sevilla, 2020

Resumen

El objetivo principal de este documento es el estudio del análisis de imágenes mediante la metodología propuesta por Laws. Para ello se efectuará un estudio teórico y matemático de los métodos de análisis de imágenes, así como las distintas características útiles para elaborar procesos de identificación y clasificación de objetos en imágenes. Se pasará a explicar en detalle el método de Laws para el análisis de las texturas existentes en las imágenes. Posteriormente se implantará la metodología completa en un programa informático desarrollado en Matlab. El programa se ha desarrollado con la idea de facilitar el uso al usuario y la posibilidad de implementar mejoras futuras progresivas. Esto servirá como herramienta para estudios avanzados de del campo del análisis de imágenes empleando este programa como asistente de resolución para comprobar los resultados obtenidos del estudio teórico.

Índice

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN: OBJETIVO Y ALCANCE

Las máquinas no existirían sin nosotros, pero nuestra existencia ya no es posible sin ellas.

- Munari, Bruno -

El presente trabajo se presenta como un estudio de la implantación de un método de análisis de imágenes concreto: *Método de Law*s. La implementación se realizará en un programa de cálculo comercial como es MATLAB. El objetivo principal es la creación de una interfaz funcional que sirva para poder analizar distintas imágenes mediante este modelo y que sea accesible a cualquier usuario facilitando su uso sin requerir conocimientos específicos en este campo.

Primeramente, en el capítulo 2 se explicará la teoría, casuística y metodologías relacionadas con el ámbito del análisis de imágenes mediante programas informáticos. Se partirá de aspectos más generales como son el color, contraste, etc. de una imagen, hasta llegar a entender los diferentes conceptos como los descriptores, clasificadores, etc.

El método de estudio y aplicación de este documento es el Método de Laws. Método que se analizará más en profundidad en el capítulo 3 y se centrará en la definición de "descriptores" basados en las distintas texturas existentes en las imágenes. Estas texturas nos servirán para poder identificar y clasificar distintos objetos en las imágenes empleando para ello clasificadores que se nutrirán de los datos resultantes de la interfaz desarrollada en este documento.

Explicada la teoría que sustenta el método, en el capítulo 4 se explicará cómo implantar todos los procesos pertenecientes a la metodología de Laws en el programa MATLAB. El código del programa se escribirá íntegramente en Matlab. Todas las funciones, cálculos y operaciones se realizarán con Matlab. El problema completo posee un número considerable de variables de todo tipo, por lo que se hace necesario la creación de un método de organización de estos datos y cómo analizarlos. Se creará para ello una metodología de programación y codificación que permita el desarrollo de una interfaz funcional la cual sirva como analizador de imágenes usando el método de Laws. Esta interfaz de desarrollará de forma que facilite el uso al usuario y pueda aportar resultados eficientes y eficaces para el proceso que ha sido creada. Se expondrá en este capítulo 4 todo el proceso seguido para la creación y programación del programa denominado *ATI* resultado del trabajo del presente documento.

ATI se plantea como un programa creado con la ayuda de Matlab que es capaz de recoger y analizar los datos de la imagen al igual que de plantear las ecuaciones de la metodología de Laws y resolverlas. Pero para mejorar el uso de este programa por parte del usuario se ha incluido funciones adicionales muy útiles que permiten mejorar la experiencia de uso. Se han incluido funciones tales como poder guardar y cargar los datos de un análisis realizado a una imagen, la creación automática de informes en pdf sobre los resultados y un manual de usuario para guiar a usuarios inexpertos entre otros.

1.1 Estado del arte

El análisis de imágenes mediante reconocimiento de texturas es un campo que empezó su trayectoria de manera teórica a mediados del siglo XX. Por aquel entonces las cámaras y los métodos de procesamiento visual no eran comúnmente utilizados bien por la falta de tecnología necesaria (cámaras, sistemas de visión, etc.) o bien por la escasez de procedimientos computacionales capaces de procesar esa información. A lo descrito se unía que, de existir los medios anteriormente descritos, el análisis computacional era lento y poco eficiente, siendo preferible buscar otras alternativas para resolver el problema.

Autores como Haralick se centrarían en definir descriptores basados en la intensidad de los píxeles de la imagen. Ojala, por su parte, definiría lo que hoy se entiende como descriptor LBP. En nuestro caso merece mención a destacar Laws, que intento cuantificar la variación de energía de cada textura de la imagen para poder ser esta información utilizada como un descriptor. Posteriormente surgirían nuevas ideas de descriptores basados en el gradiente de la imagen (HOG) que también sería de relevancia.

En cambio, muchos son los distintos métodos con los que contamos hoy día para hacer frente a este problema. Unido a todo ello está la implicación cada vez más grande de cámaras en casi todos los procesos conocidos siendo de la índole que sean. Esto, a pesar de suponer un gran avance requiere de la implementación de métodos de análisis de imágenes para poder procesar la información necesaria. Los avances teóricos alcanzados en este campo de la ciencia junto con los avances computacionales hacen que actualmente se puedan implementar estos métodos en casi cualquier aparato electrónico de uso cotidiano. Con el gran avance en el campo de la informática, ya no supone un gasto computacional grande para los procesadores actuales. Todo ello ha desembocado en una gran aplicación de estos métodos para distintos fines: Control de procesos, Implementación en robot industriales, Detección de objetos, Ocio, etc. Las posibilidades de aplicación son muy elevadas y cada día surgen nuevas aplicaciones que se nutren de estos métodos.

Como ejemplos concretos de utilización de esta tecnología podemos encontrar: el monitoreo de piezas dentro de fábricas con el fin de detectar piezas erronas o con defectos; el reconocimiento de rostros para el control de personal en una empresa; la identificación del carril propio o contrario en un coche de conducción autónoma.

2 INTRODUCCIÓN A LOS MÉTODOS DE ANÁLISIS DE IMÁGENES

El verdadero progreso es el que pone la tecnología al alcance de todos.

- Ford, Henry -

El proceso de análisis y clasificación entendido de manera global puede definirse con las siguientes etapas básicas:

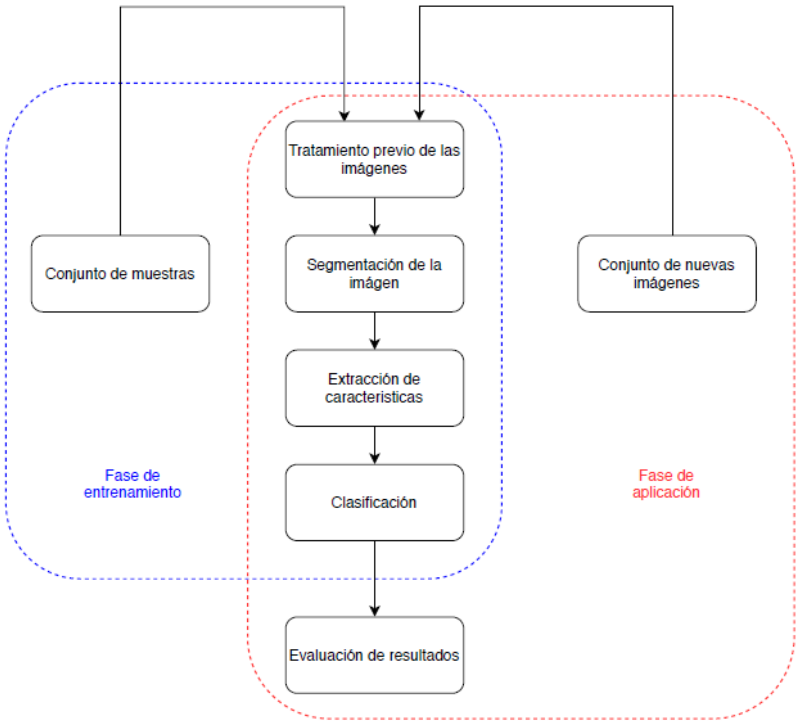


Figura 1

Primeramente, se adquiere un conjunto elevado de muestras de diferentes texturas ya clasificadas. Este conjunto de datos nos servirá como referencia de distintas texturas.

Como segundo paso opcional se debe realizar un tratamiento de la imagen con el objetivo de eliminar elementos que puedan afectar a la aplicación de los métodos, como son ruido, defectos de iluminación, de contraste, etc.

Al tratarse la imagen de una composición de elementos, habrá que realizar una etapa de segmentación que trate de dividir las distintas regiones que aparecen, con el objetivo de seleccionar aquellas de interés. Esto puede ser entendido como de un problema de detección de objetos, donde habría que localizar los posibles elementos de interés.

Como siguiente paso iría una etapa de entrenamiento para que el sistema se capaz de aprender a diferenciar las distintas texturas contenidas en la imagen. Finalizada esta etapa se asignará una etiqueta a cada muestra del conjunto de datos inicialmente conseguidos para probar la eficacia del sistema. Finalmente, los resultados obtenidos serán evaluados.

2.1 Conceptos generales

Una imagen digital puede tratarse como una matriz de tamaño $M \times N$ en la que cada elemento, denominado píxel, discretiza y describe una región del espacio. Usando los índices i y j para la asignación de las filas y columnas respectivamente de esa matriz se puede hacer referencia a cada punto de una imagen como Imagen(i,j).

Normalmente se trabaja con un rango de 8 bits (1 byte) de información por cada punto, lo que es equivale a un total de 28 valores distintos que podrán representar el nivel de intensidad de un píxel en el intervalo [0, 255]. El valor 0 representa la ausencia de intensidad (lo que se correspondería con un píxel negro en la imagen), a medida que la intensidad va aumentando, van apareciendo tonos de gris más claros hasta llegar a un valor de 255, la mayor intensidad posible (píxel blanco).

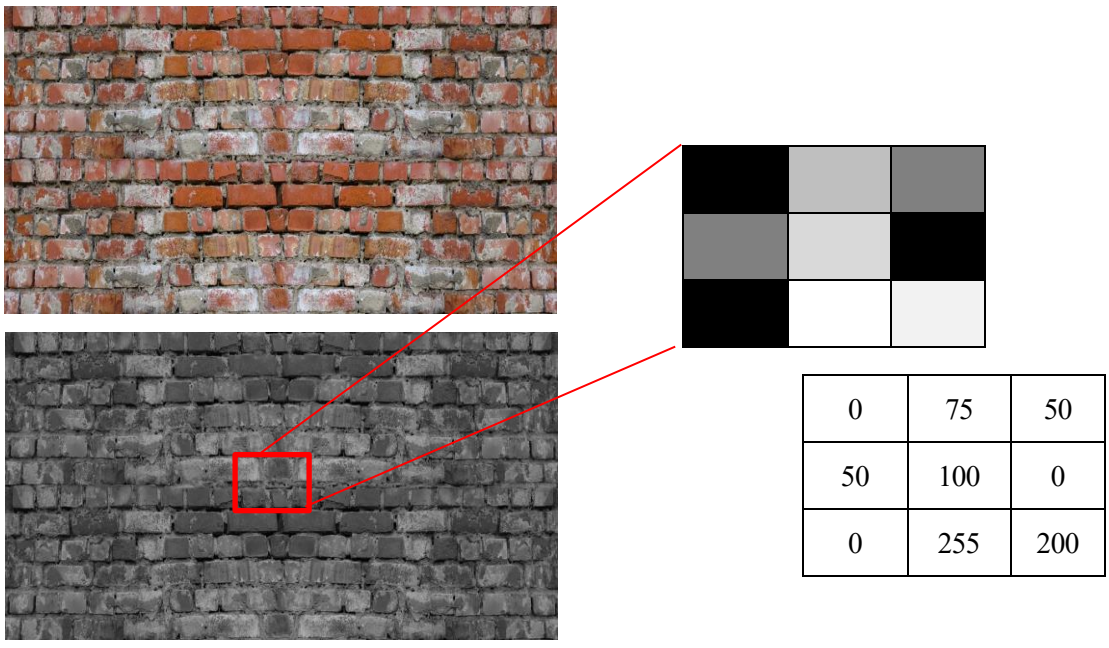


Figura 2

2.2 Muestreo y Cuantificación

Cuando capturamos una escena del mundo real nos encontramos con una señal continua, es decir, analógica, que debemos discretizar en dos sentidos a la hora de digitalizar la imagen: la cuantificación y el muestreo.

La cuantificación, como se ha comentado es habitual trabajar con 8 bits de información por píxel, consiguiendo así diferenciar entre 256 niveles de gris distintos. Es por esto que, en la práctica, a pesar de que se pueda realizar un proceso de cuantificación más preciso, 8 bits son más que suficientes para la mayor parte de las aplicaciones.

El muestreo hace referencia al número de píxeles en los que se dividirá la escena, de modo que, cuanto mayor sea este número, más detalles se podrán captar, y viceversa. Las cámaras actuales cuentan con la potencia necesaria para dividir la escena en millones de píxeles, por lo que la resolución alcanzada puede ser enorme. Sin embargo, desde el punto de vista del análisis de patrones y propiedades con una resolución tan elevada conllevaría un gasto computacional tremendo, además de que las variaciones observables disminuirían su frecuencia de aparición (no es lo mismo que aparezcan cambios interpretables a uno o dos píxeles de distancia que a varias decenas), pudiendo incluso perjudicar los resultados de caracterización. Por este motivo, para la mayoría de las aplicaciones que conlleven el análisis de un número muy elevado de muestras, se recurre a reducir la resolución de las muestras.

2.3 Color

De forma habitual el color es una propiedad que se suele representar con el modelo RGB (u otros modelos como el CMY de sus siglas en inglés).

Figura 3

De forma teórica la mezcla de estos colores primarios de luz (Rojo, verde y azul en este caso) dan lugar a nuevos colores. De esta forma los colores de una imagen pueden ser representados como una combinación de valores de estos tres colores. Las maneras normales de trabajar en estos casos son o empleando una matriz de NxMx3 de tres dimensiones (cada dimensión destinada a un color) o trabajar con tres matrices distintas de NxMx1 (cada matriz NxM destinada a un color).

Como una forma sencilla de identificación y clasificación en una imagen se podría pensar en emplear métodos que se basen en el color para poder realizar esta tarea. De esta forma identificar una nube blanca en un cielo de fondo azul puede parecer sencillo. Aun así, este método es poco robusto y depende de muchos más factores que afectan indirectamente al color como son la luminosidad, saturación, sombras, etc. Es por ello que emplear conceptos superiores como las texturas se plantea como una solución a los errores de esto métodos.

Cabe decir que estos valores de color además de no aportar una ayuda clara en el proceso de identificación y clasificación en imágenes pueden llegar a afectar negativamente a otros métodos. Los valores de color (RGB) puede aumentar la carga computacional o influir en errores por lo que usualmente se convierte la imagen a estala de grises para poder trabajar mejor con ella. De esta forma, los valores de color quedan englobados y simplificados a una escala de grises con valores de 0 para el color negro y de 255 para el color blanco.

2.4 Histograma

Si realizamos una representación del número de pixeles que aparece en la imagen con el mismo tono de gris podemos el llamado histograma de la imagen. Esta representación muestra el nivel de intensidad frente al número de pixeles que tienen dicha intensidad, proporcionando así una información general de la imagen. El análisis de su forma puede ayudar a la extracción de información de la imagen.

2.5 Intensidad

Como ya se ha descrito anteriormente, una vez pasada la imagen a escala de grises se consigue asignar a cada píxel un valor en la escala [0 - 255] según su color original. Es este valor lo que se conoce como intensidad de un pixel o nivel de gris medio. Como intensidad media se define:

$$Intensidad\ media\ (m)\ = \sum_{i=1}^{L} z_i\ p(z_i)$$

Siendo *p* el vector de probabilidades de niveles de gris.

2.6 Brillo

El brillo se puede entender como una estimación promedio de cuan cerca están la mayoría de pixeles del valor 255 (en escala de grises). Una imagen muy oscura seria aquella que posee muchos pixeles con valores cercanos a 0. Por el contrario, se dice que la imagen tiene mucho brillo si posee mucha cantidad de pixeles con valores cercanos a 255.

2.7 Contraste

Partiendo del concepto de intensidad se puede decir que si todas las intensidades de la imagen son de un nivel parecido la imagen tendrá poco contraste. En cambio, si la imagen posee distintos niveles de intensidad el contraste será más elevado. Una imagen con más contraste refleja mejor visibilidad de las distintas regiones.

2.8 Gradiente

Se define como gradiente de cada pixel un vector (con dos componentes, horizontal y vertical) cuyo valor es la dirección de la máxima variación de intensidad entre dicho pixel y sus vecinos. De esta forma, un pixel (i,j) con gradiente con valor [1 0] indica que en la dirección horizontal se encuentra la mayor variación de intensidad con respecto al pixel. De forma análoga un pixel (i,j) con gradiente con valor [0 25] indica que la mayor variación de intensidad es en dirección vertical. El módulo del vector también nos informa del contraste

del pixel en cuestión con respecto a sus vecinos. De esta forma el gradiente se emplea en la detección de contornos. La definición matemática del gradiente resulta:

$$\nabla g = \begin{pmatrix} Gx \\ Gy \end{pmatrix} = \begin{pmatrix} \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial y} \end{pmatrix}$$

El módulo del gradiente se puede expresar:

$$|\nabla g| = \sqrt{Gx^2 + G_y^2}$$

2.9 Características para la descripción de texturas

El conjunto de características o rasgos que puede n ser medidos en una imagen sobre un objeto o región concreta se puede denominar como: descriptor. Los descriptores deben de presentar el mismo comportamiento ante variaciones en la imagen y ser lo suficientemente discriminatorios para permitir a un clasificador poder distinguirlos con facilidad. Como concepto de descriptores que se han usado para la caracterización de texturas se presentan:

- Descriptores obtenidos por métodos estadísticos
- Descriptores obtenidos por el gradiente
- Descriptores LBP
- Descriptores de Laws

A pesar de esta división se puede discriminar entre descriptores de primer y de segundo orden.

- **Descriptores de primer orden:**

  Se obtienen de utilizar los momentos del histograma respecto a la intensidad de una imagen dada. Estos descriptores no repercuten mucho gasto computacional. El momento se calcula con la siguiente fórmula. Como este se pueden calcular más momentos de órdenes superiores. El momento n-ésimo se calcularía como:

$$\mu_n(z) = \sum_{i=1}^{L} (z_i - m)^n \; p(z\_i)$$

  Siendo *m* el valor de la intensidad media de la imagen y *p* el vector de probabilidades de niveles de gris.

  Se puede calcular el momento de segundo orden o también llamado varianza. Se usa como descriptor para cuantificar el contraste de intensidad de la imagen. El momento de tercer orden es capaz de cuantificar la desviación del histograma respecto al valor medio de la intensidad El momento de cuarto orden mide la monotonía relativa del histograma.

- **Descriptores de segundo orden:**

  Se define la matriz Co-ocurrente como aquella que establece la relación entre cada pixel con otro pixel de su entorno a una distancia determinada. Este cálculo conocido por sus siglas en inglés GLCM (Gray Level Co-occurrence Matrix) se emplea para mejorar los resultados obtenidos de los descriptores de primer orden utilizando para ello la información de la posición relativa de los pixeles de la imagen. El gasto computacional es un poco superior al necesario por los descriptores de primer orden

- **Descriptores de Laws**

  Cabe una mención especial para el conjunto de descriptores propuestos por Laws en su método. Laws con su método propone una serie de vectores, a los que llamará filtros, que combinados entre sí de una forma determinada darán lugar a unas matrices llamadas máscaras (también se conocen como plantillas, o ventanas). El objetivo fundamental de estas matrices es su utilización para el cálculo de la llamada "energía de una textura" usando este concepto como un descriptor.

  A rasgos generales Laws establece un método por el cual la imagen original es tratada con estos filtros y máscaras para obtener las características de cada pixel. Estas características son empleadas como descriptores que se emplearán para un proceso posterior de identificación y clasificación de imágenes. El método de Laws será descrito en profundidad en el capítulo 3 de este documento.

2.10 Clasificadores

Se pueden entender como clasificadores aquellos sistemas (métodos informáticos) capaces de realizar una división de unas muestras entregadas de acuerdo a una serie de características concretas. De esta forma, podemos estudiar a modo de ejemplo hipotético un clasificador que sea capaz de clasificar las imágenes dadas según en ellas aparezca naranjas o manzanas. Este clasificador deberá basarse en un conjunto de características las cuales le sirvan para hacer esa diferenciación. Así, el clasificador podrá fijarse en parámetros como la forma, el color o la textura para poder decidir si en la imagen aparece una naranja, una manzana o ninguna de estas frutas.

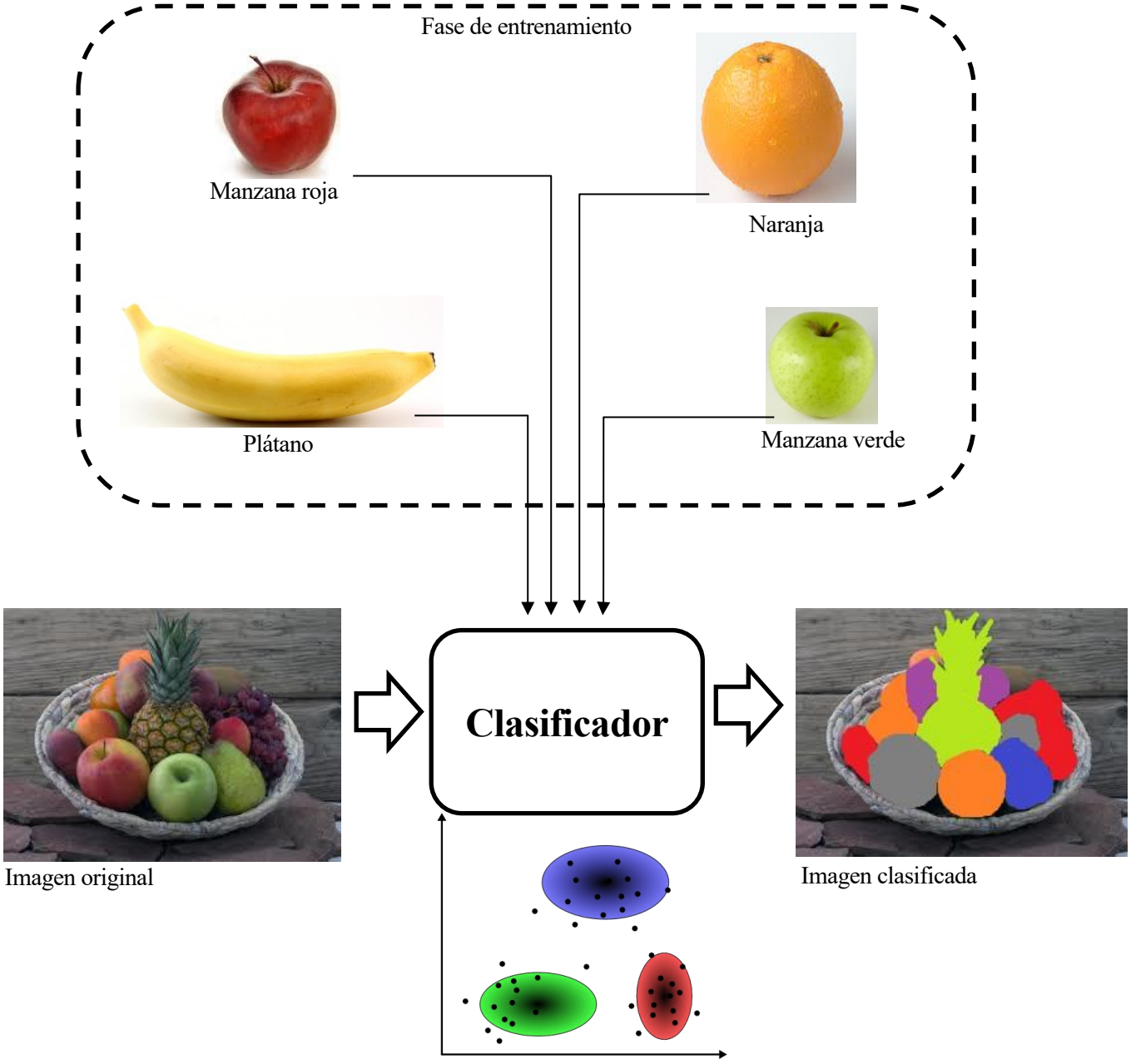


Figura 4

Esta serie de características que sirven de apoyo a los clasificadores son los descriptores. Estos descriptores surgen como resultado de la fase previa al clasificador conocida como análisis previo de la imagen. Estos descriptores dependerán del método de análisis elegido (descriptores LBP, descriptores basados en el gradiente, descriptores de Laws, etc.). En el caso de estudio de este documento nos centraremos en los descriptores propuestos por Laws en su metodología. Estos descriptores se basan en el análisis de las texturas existentes en la imagen para poder tener parámetros y características suficientes para poder diferenciar distintos objetos en imágenes.

Para esta primera fase de clasificación se vuelve fundamental disponer de un conjunto lo suficientemente representativo para poder mejorar el proceso discriminatorio posterior. Por otro lado, también resulta de interés disponer de bastante variabilidad en las muestras para permitir adaptar mejor al clasificador a los cambios leves en estos descriptores. Así, también se vuelve necesario utilizar en cada caso los descriptores que mejor se adapten y ofrezcan las características discriminatorias suficientes para realizar la tarea de discriminación. Puede darse el caso que en una imagen la discriminación efectuada solo teniendo en cuenta las texturas sea

insuficiente, haciéndose necesario utilizar otro método o combinación de métodos.

Como clasificadores se pueden utilizar clasificadores de mínima distancia, clasificadores lineales o clasificadores basados en redes neuronales muy desarrollados en la actualidad. Todos estos clasificadores independientemente de su tipología aportarán un resultado que deberá ser evaluado para conocer como de bien realizan su trabajo. Empleando parámetros como son el porcentaje de clasificaciones incorrectas (conocido con sus siglas como PCI) podremos determinar si el clasificador es capaz de identificar y discriminar con suficiente eficacia.

3 MÉTODO DE LAWS

Los avances tecnológicos no pueden suceder sin científicos o ingenieros. El desafío de la sociedad es equiparar a las suficientes personas, con las habilidades correctas y formas de pensar, que lleguen a trabajar en los problemas más importantes.

- Schmidt, Eric -

Laws en su estudio publicado en 1980 buscaba desarrollar métodos de análisis para extraer toda la información esencial de imágenes a través del estudio de las texturas que en ella aparecen. Mediante el análisis de la escena, Laws buscaba conocer qué propiedades, características y relaciones existen entre los distintos objetos que pueden aparecer en una imagen. Estos estudios resultarían útiles para los análisis superficiales del tratamiento de metales, el estudio de las fallas geográficas o el análisis meteorológico de la observación del cielo entre otros.

Laws propone como método de detección y clasificación de texturas en imágenes una aproximación estadística. De esta forma definió un concepto nuevo llamado Energía de textura (ET). Un único parámetro que se evalúa en cada punto de la imagen a través de una convolución de la imagen con una serie de vectores y matrices llamadas máscaras o ventanas.

Una vez convolucionada la imagen con las máscaras la suma de los valores cuadrados es llamada por Laws "medida de la ET", (a partir de ahora TEM por sus siglas en inglés) aunque también se puede emplear la suma de los valores absolutos, la suma de la desviación típica medida con la imagen original, etc.

Denominando a la matriz representativa de la imagen (resultado de un tratamiento previo) como *I* se debería realizar la convolución de esta matriz con las matrices *M* llamadas ventanas fruto de la combinación de 2 o más filtros denominados F_i (para *i* filtros distintos). Este resultado lo denominaremos como:

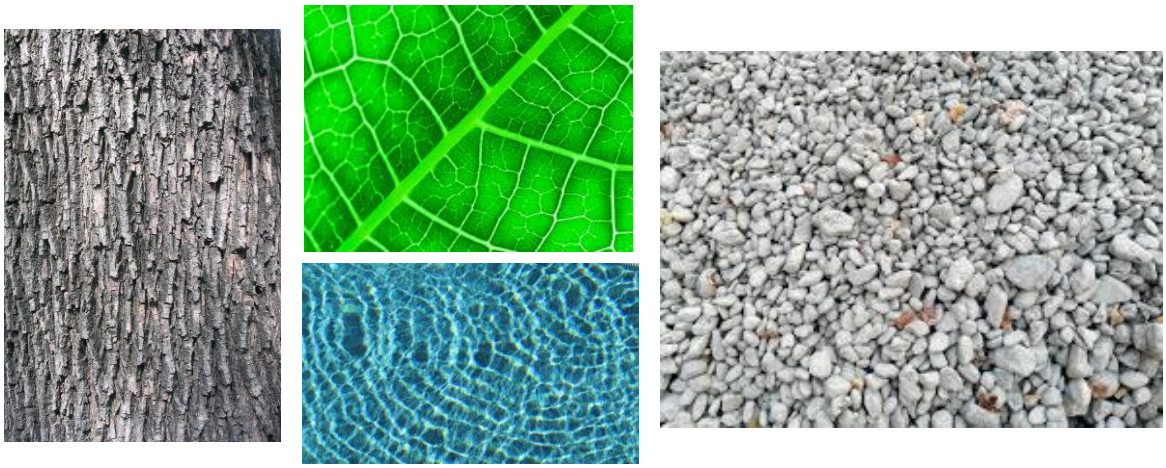
$$R_n = I * M_n$$

$$Siendo\ M_n\ una\ combinación\ de\ los\ filtros\ F_1, F_2, \dots F_i\ cualesquiera$$

A continuación, se explicarán ciertos conceptos como textura, filtros y máscaras y se pasará a la de descripción del método de Laws.

3.1 Textura como base del método

Son muchas los campos en los que las texturas suelen cobrar importancia en imágenes. Ejemplos como imágenes de las hojas de árboles, fachadas de ladrillos de edificios, grietas existentes en superficies de metales, etc.



Todos estos ejemplos guardan en común que están compuestos por elementos similares repetidos en forma de patrones para cada imagen. Estos patrones son llamados la textura del elemento y posee una relación íntima con la estructura de cada elemento. Identificado el patrón que conforma esta estructura este se extiende al resto del objeto.

El ojo humano es capaz, de forma natural, realizar una discriminación de estas texturas para poder identificar y clasificar los diferentes objetos existentes en una imagen. Identificadas las texturas se asume una cierta homogeneidad que simplifica la imagen y ayuda como método predictivo para definir el objeto (identificado una parte de la imagen de ladrillos es fácil entender visualmente la imagen completa).

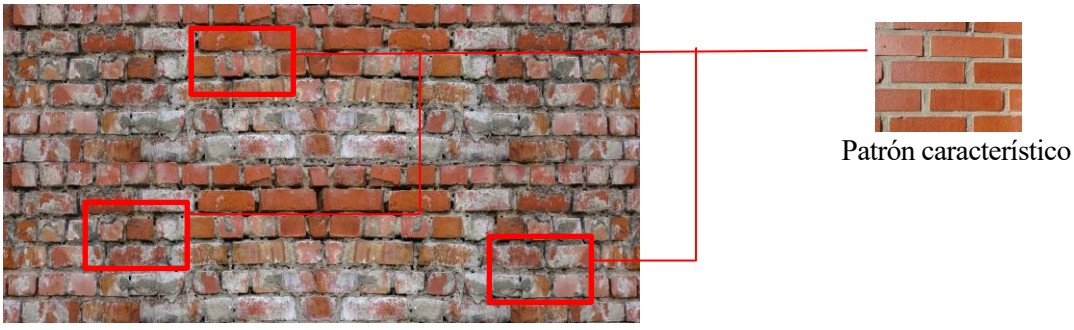


Figura 5

El sistema visual humano analiza estas propiedades texturales como son los grosores, dimensiones, aparición de puntos, ondas, etc. y realiza un proceso de agrupación en el cual unifica estas propiedades asignándoselas a una textura concreta ("textura de ladrillo", "textura de hoja", etc.). Este proceso biológico se basa en la identificación de las distintas propiedades (puntos, rallas, ondas, …). Una vez identificadas son agrupadas y contenidas bajo un único termino. Finalmente, el conjunto de términos existentes en la imagen es clasificado obteniendo una discretización razonable de cada objeto.

3.2 Definiciones previas del Método de Laws

A pesar de esta descripción general es necesario la definición de ciertos conceptos importantes que faciliten la explicación de este método:

- **Definición de filtros**

Bajo el método de Laws se entiende como filtros a un conjunto de vectores de cualquier dimensión impar siendo las más comunes tres, cinco, siete o nueve. La razón de ser impar es para mantener un píxel central en los futuros cálculos a realizar con estos filtros. Cada filtro se suele denominar con una letra mayúscula correspondiente a la medida que realiza (Level, Edge, Spot, Wave, Ripple, etc.) seguida de un número que indica el tamaño de ese vector.

Se muestra a continuación a modo de ejmplo, algunos filtros que Laws definió para regiones cuadradas de pixeles de 3x3, 5x5 y 7x7:

Región de 3x3	Región de 5x5	Región de 7x7
*L*3 = [1 2 1] *E*3 = [−1 0 1] *S*3 = [−1 2 −1]	*L*5 = [1 4 6 4 1] *E*5 = [−1 −2 0 2 1] *S*5 = [−1 0 2 0 −1] W5 = [−1 2 0 -2 −1] *R*5 = [1 −4 6 −4 1]	*L*7 = [1 6 15 20 15 6 1] *E*7 = [-1 -4 -5 0 5 4 1] *S*7 = [-1 -2 1 4 1 -2 -1] W7 = [-1 0 3 0 -3 0 1] R7 = [1 -2 -1 4 -1 -2 1] O7 = [-1 6 -15 20 -15 6 -1]

Figura 6

Estos filtros se pueden entender están especializados en características como: rugosidad (R), ondulación (W), media ponderada de intensidades (L), puntos (S), gradiente (E).

En cada filtro la suma de todos sus elementos es igual a cero, de esta forma en una región constante la respuesta obtenida será nula. Las excepciones de esta regla son los filtros L, el cual se verá en próximos apartados que se utilizará para procesos de normalización.

Cada filtro es capaz de resaltar una característica distinta de textura (ejem: rizos, ondulación, forma, puntos, rugosidad, etc.). Cuando la matriz representativa de la imagen es convolucionada con los distintos filtros se obtienen nuevas imágenes en las que estas características están resaltadas.

- **Definición de máscara o ventana**

Lo que Laws denominó como máscaras o ventanas no son más que matrices que representan la morfología de los patrones buscados como texturas en la imagen. Estas máscaras estarán especializadas en aspectos distintos como son la búsqueda de bordes, líneas, puntos, ángulos o cualquier otra estructura regular.

Las máscaras surgen de la multiplicación de pares de filtros. El objetivo de su utilización es cualificar posibles variaciones en las texturas de la imagen. Las combinaciones de los filtros dan nombre a cada máscara. Siendo por ejemplo la máscara llamada "L3E3" la correspondiente de combinar los filtros L3 y E3 definidos anteriormente.

Cada máscara se especializa en la detección de un aspecto en concreto. A modo de ejemplo la máscara L3E3 es capaz de detectar los bordes verticales mientras que la máscara E3L3 detecta los bordes horizontales. Existen multitud de combinaciones de filtros que dan lugar a máscaras especializadas en detectar aspectos en concreto como puntos, ondas, rizos, etc.

3.3 Explicación del Método de Law

Se expone a continuación la metodología paso a paso del Método de Laws para el cálculo de la energía de las texturas en imágenes:

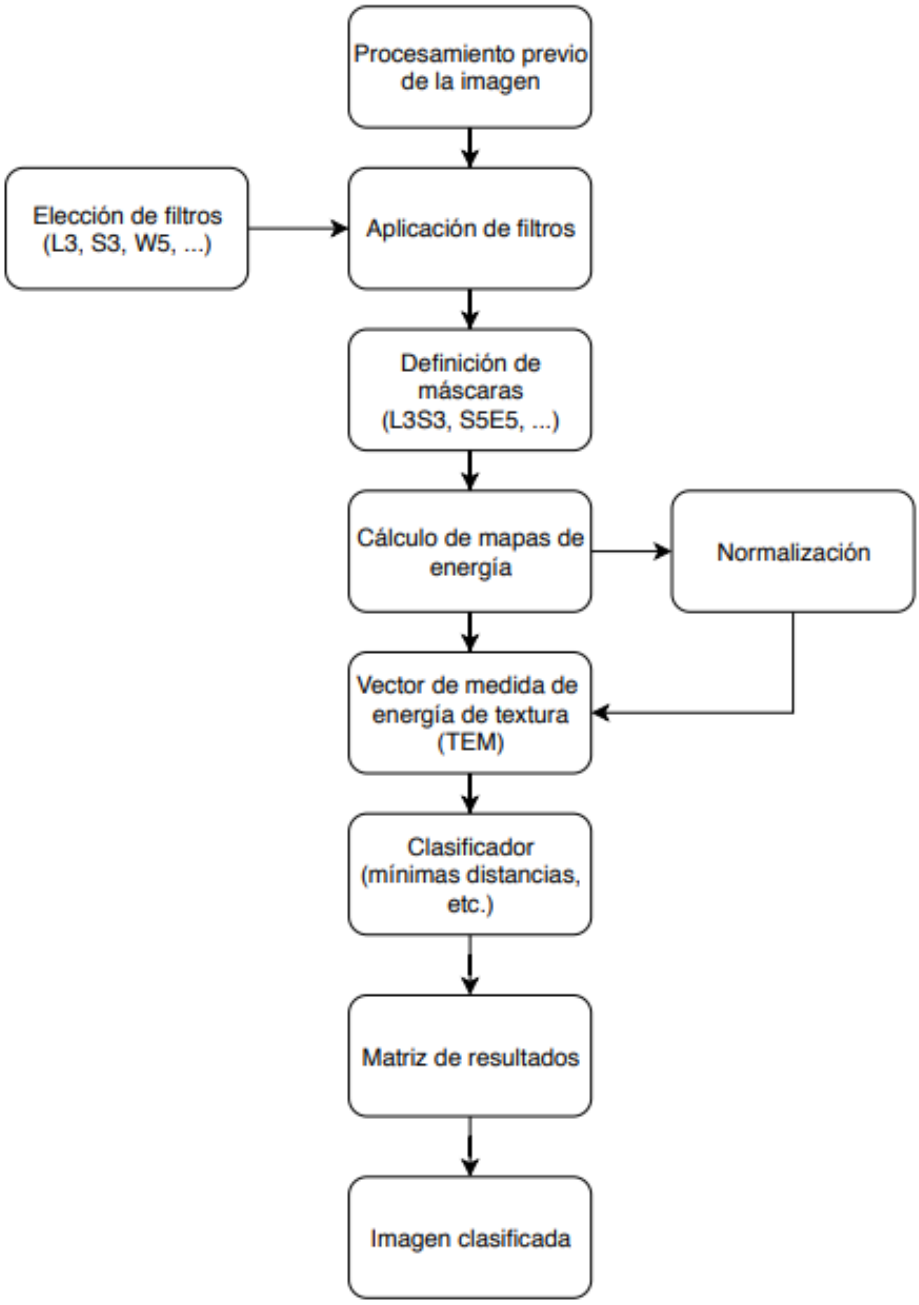


Figura 7

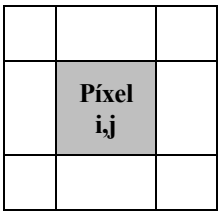
- **Procesamiento previo de la imagen**

Para poder aplicar de forma correcta todo el método se necesita previamente tratar la imagen a utilizar. Lo habitual es convertir la imagen a escala de grises para que la variable color no afecta al método. También es conveniente eliminar los efectos de la iluminación. Es muy común disminuir la escala de las imágenes a tratar para, de esta forma, aumentar el número de veces que aparecen las distintas texturas.

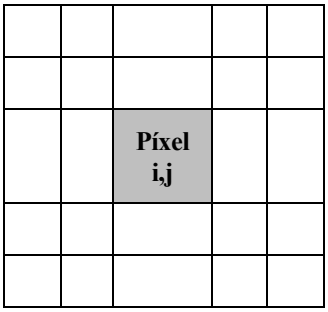
- **Aplicación de los filtros**

Como primera decisión se elige el tamaño de los filtros que se usarán. Esto está íntimamente relacionada con el rango de vecindad que se estudia alrededor de cada píxel. Como ejemplos de cuadros de vecindad alrededor

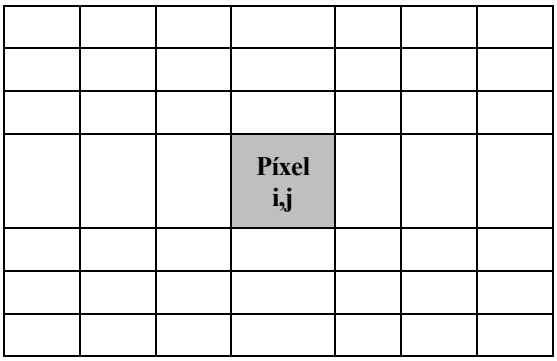
del pixel central se pueden encontrar rangos de 3x3, 5x5, 7x7, 9x9, etc.



Región 3x3



Región 5x5



Región 7x7

El paso siguiente es la definición de los términos de cada filtro. Se deberá tener en cuenta que un filtro de tamaño n estará compuesto de n términos. Estos términos pueden ser definidos a gusto del usuario, pero existen ciertas reglas que ayudan al proceso de análisis. Como regla general (ya que esto facilitará los futuros procesos de normalización) los términos de todos los filtros, a excepción de los filtros L, se definen de forma que su sumatorio total sea igual a cero. Esta regla es de aplicación a todos los filtros menos los denominados filtros L *(Level*). Los términos de estos filtros no tienen por qué sumar cero. A continuación, se muestra unos ejemplos de filtros:

$$L3 = [1, \quad 2, \quad 1] \rightarrow \sum_{i=1}^{3} L3(i) = 4$$

$$S5 = [-1, \quad 0, \quad 2, \quad 0, \quad -1] \rightarrow \sum_{i=1}^{5} S5(i) = 0$$

$$E7 = [-1, \quad -4, \quad -5, \quad 0, \quad 5, \quad 4, \quad 1] \rightarrow \sum_{i=1}^{7} E7(i) = 0$$

$$L5 = [1, \quad 4, \quad 6, \quad 4, \quad 1] \rightarrow \sum_{i=1}^{7} L5(i) = 16$$

Una vez definidos los filtros que se utilizarán se deberán combinar para poder generar las máscaras necesarias. Posteriormente se concolucionará cada máscara con la imagen ya preprocesada

- Aplicación de las máscaras

Cada filtro definido es capaz de detectar una variación concreta de la imagen (bordes, líneas, etc.), por lo general estas variaciones son básicas. La combinación de estos filtros puede significar la detección de patrones de un grado más alto de complejidad. De esta forma combinando los filtros L3 y E3 por ejemplo, se pueden detectar líneas verticales de la imagen. Otras combinaciones supondrán la detección de otros patrones como ondas, líneas horizontales, puntos, etc. La combinación es totalmente arbitraria y la define el usuario de la mejor forma que considere. Si bien existen ciertas combinaciones que provocan resultados más satisfactorios que otros. Como cada combinación representará un conjunto de características para cada píxel, una opción muy utilizada es la combinación de todos los filtros entre sí. De esta forma se obtendrían todas las posibles características de cada píxel.

$L3L3 = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	$L3E3 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$L3S3 = \begin{bmatrix} -1 & 2 & -1 \\ -2 & 4 & -2 \\ -1 & 2 & 1 \end{bmatrix}$
$E3L3 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$	$E3E3 = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	$E3S3 = \begin{bmatrix} 1 & -2 & 1 \\ 0 & 0 & 0 \\ -1 & 2 & -1 \end{bmatrix}$
$S3L3 = \begin{bmatrix} -1 & -2 & -1 \\ 2 & 4 & 2 \\ -1 & -2 & -1 \end{bmatrix}$	$S3E3 = \begin{bmatrix} 1 & 0 & -1 \\ -2 & 0 & 2 \\ 1 & 0 & -1 \end{bmatrix}$	$S3S3 = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$

Figura 8

En la imagen anterior se muestra, a modo de ejemplo, todas las máscaras resultado de la combinación de todos los filtros de región 3x3. Las matrices resultado de la combinación de filtros tendrán un tamaño a razón de los filtros que lo componen. Así, en una región de NxN se definirán filtros de N dimensiones y las máscaras resultantes de combinar estos filtros tendrán un tamaño NxN.

Finalmente, estas matrices serán convolucionadas con la matriz representativa de la imagen.

- **Cálculos de mapas de energía**

Una vez convolucionada la imagen con las máscaras se calcularán los mapas de energía de cada imagen. El procedimiento consiste en analizar los valores vecinos de cada pixel (por ejemplo, en un rango local de 15x15 píxeles). Se sumarán los valores absolutos de cada pixel de la vecindad descrita y se asignará el valor al pixel central. El resultado es un nuevo conjunto de imágenes.

- **Normalización**

Todas las máscaras de convolución tienen un sumatorio igual a cero excepto las máscaras provenientes de la combinación de filtros L (*Level*) como por ejemplo L5L5. Es habitual que estas máscaras no se empleen para calcular los mapas de energía de texturas, pero si como proceso de normalización. Gracias a este proceso conseguiremos asegurar que todos los valores de los píxeles de cada imagen este comprendidos entre 0 y 1. Es por ello que la imagen resultante de la convolución de la máscara L5L5 con la imagen preprocesada sea descartada ya que no aporta información.

- **Combinación de medidas similares**

Un paso posterior y muy usado es la combinación de pares de mapas de energía de texturas para poder extraer información relevante. Habitualmente con este paso se busca obtener imágenes invariantes a rotaciones.

- **Obtención del vector de medida de energía de texturas (TEM)**

Como último paso en este proceso se puede obtener las características texturales de cada píxel. De los procesos anteriores hemos conseguidos distintas imágenes (mapa de energía de texturas) que aportan distinta información para cada píxel. Se puede realizar una media de estos valores para obtener una imagen global combinación de cada imagen anteriormente creada.

- **Evaluación de características**

Como paso adicional se puede evaluar la capacidad de discriminación de cada característica principal (media de los valores obtenidos en el paso anterior) usando un clasificador de distancias (Clasificador de mínimas distancias Euclídeas)

3.4 Problemas y limitaciones del Método de Laws

Muchas de las limitaciones existentes en este método están relacionadas con las máscaras definidas por Laws. Al tratare de máscaras ideales provenientes de vectores que buscan diferencias de primer orden como sería la detección de bordes, o de segundo orden como sería la detección de puntos, en las imágenes reales la casuística es mucho más compleja. A pesar de disponer distintas máscaras especializas en distintos procesos de detección la detección de otras texturas complejas se vuelve errónea.

Otro de los principales problemas que se pueden encontrar en este método es la introducción de errores en zonas de transición de las imágenes. Se trata de zonas delicadas donde se limitan diferentes texturas. Este método, al usar ciertas regiones definidas por un tamaño fijo (3x3, 5x5, 7x7, etc.) puede llegar a dar valores intermedios de estas dos regiones resultado de la mezcla de texturas. Estos valores quedan fuera de los valores normales existentes en la imagen pudiendo influir en errores. Métodos posteriores al de Laws como el propuesto por Hsaio y Sawchuk en 1989 consiguen solventar estos fallos.

Además, el método de Laws no está adaptado a cambios de escala o rotaciones.

4 IMPLEMENTACIÓN DEL MÉTODO DE LAWS EN MATLAB

En cada aumento de las condiciones así como en la fabricación de nuevas herramientas, el trabajo humano se convierte en versión abreviada.

- Babbage, Charles -

Una vez expuesto los diferentes métodos existentes para el análisis de texturas y las aplicaciones habituales de imágenes y explicado extensamente el método del que se basa este texto: Método de Laws, se plasmará en este capítulo la herramientas y metodología empleada para la implementación completa del método descrito en una interfaz gráfica.

El objetivo es la adecuación de todos los pasos descritos en el capítulo anterior (3.-Método de Laws) en una herramienta informática capaz de automatizar el proceso, y facilitar el uso al usuario final. Se presentan por tanto unos objetivos claros descritos a continuación los cuales la interfaz gráfica a programar será capaz de resolver:

- Programación de una interfaz gráfica sencilla para facilitar el uso al usuario
- El proceso deberá estar faseado de la misma forma en la que se ha faseado el método. De esta forma el uso será más intuitivo y ameno.
- Se debe procurar en la medida de lo posible la claridad a la hora de la toma y muestra de datos al usuario.
- La interfaz gráfica deberá se programada en Matlab por ser una herramienta informática de programación más que adecuada para la elaboración de grandes cálculos con matrices y funciones.

El programa desarrollado se ha denominado *ATI* de las siglas en inglés *Images Texture Analysis*. A continuación, se explica cómo ha sido el desarrollo completo de esta interfaz

En las siguientes líneas de este capítulo se explicarán entre otros el porqué de la elección del programa de codificación elegido, la metodología u orden propios implantados en el código, así como el desarrollo del flujo de operaciones necesarios para poder trasladar la idea de Laws a un programa funcional.

4.1 Elección del programa y requisitos previos. Software

Para el desarrollo de todos los elementos que componen la parte software de este TFM, se ha empleado la aplicación MATLAB de MathWorks en su versión R2017b. MATLAB es una herramienta matemática, con un lenguaje de programación propio (lenguaje *M*), ampliamente utilizada en el ámbito científico, particularmente en el marco de la ingeniería. El lenguaje M está orientado a matrices y, dado que, como se indicará más adelante, las imágenes digitales pueden considerarse matemáticamente como tales, se facilita sobremanera tratar con este tipo de elementos. Además, el potencial de MATLAB radica en que se puede complementar con numerosos *toolboxes* especializados que se han ido desarrollando a lo largo de los años. En concreto, el *Image Processing Toolbox* está enfocado al tratamiento de imágenes y contiene una gran cantidad de funciones predefinidas para tal fin. MATLAB también dispone de herramientas que permiten el desarrollo de interfaces gráficas de usuario (generalmente conocidas como *GUI*, del inglés *Graphical User Interface*) sin demasiada dificultad.

Figura 9

Un manual de la GUI desarrollada se adjunta en el *Anexo A*, al final de este documento. No obstante, se prescindirá de cualquier tipo de explicación referente al funcionamiento del software en lo relativo a las funciones predefinidas, tanto de la aplicación principal como del *toolbox* especializado, más allá de los propios comentarios desarrollados que complementan al código, dado que no es ese el objetivo del proyecto.

El programa creado debe ejecutarse en *Matlab 2017b* o versiones superiores siendo necesario siempre que *Matlab* este instalado en el equipo previamente. Por ello se ha creado también una versión de *ATI* capaz de ser reproducida sin *Matlab* es decir que no necesita tener instalado *Matlab* en el equipo. Esta versión se basa en la aplicación *MATLAB Compiler Runtime (MCR)* que es capaz de realizar todas las operaciones y cálculos sin necesidad de tener instalado *Matlab*. Esta opción con MCR garantiza todas las operaciones del programa *ATI* excepto una, la creación de informes en pdf. Debido a las licencias que usa MCR no puede crear documento

pdf por lo que esta versión de *ATI* no podrá crear informes de las imágenes analizadas.

4.2 Definiciones previas. Metodología

Para la creación de cualquier programa que se precie se hace necesario seguir unas pautas que permitan infundir un orden a todo el desarrollo de los trabajos. Este orden permite dividir el trabajo total de la programación en problemas más sencillos de resolver además de permitir disminuir las posibles fuentes de errores a la hora del desarrollo. Al conjunto de pasos que conforman este orden intrínseco que se ha llevado a cabo en el desarrollo del código se le ha llamado "Metodología" y se ha intentado clasificar según su naturaleza para poder mejorar su estudio. Esta metodología atiende a todos los puntos del desarrollo del código, desde la escritura misma del código pasando por la organización de las variables que en él se utilizan hasta llegar a puntos como son la organización de las carpetas donde se localizan los archivos o la interfaz empleada para el uso del programa.

A continuación, se desarrolla la metodología seguida en las distintas partes del desarrollo completo del programa *ATI:*

- Metodología de carpetas:
- Metodología de variables
- Metodología de funciones
- Metodología de interfaces

Si bien cabe decir que no se tratan de metodologías absolutas, sino que son una forma de organización propia escogida para el desarrollo de este programa en concreto. Conocerlas resulta útil para poder realizar correcciones en el programa, o implantar futuras mejoras en él. Se presentan por ello, como las reglas básicas que se han seguido para la elaboración del programa.

4.2.1 Metodología de carpetas

Como metodología de carpetas empleada que se ha escogido ha buscado la simplificación y la minimización de errores tanto en el desarrollo del programa como en su utilización por parte del usuario. El programa se compone de un único archivo ejecutable en Matlab en cuál será el iniciador de toda la interfaz creada. Por otra parte, todas las funciones, variables, archivos y documentos se han almacenado dependiendo de su posterior uso. De esta forma se ha incurrido en la creación de las siguientes carpetas internas del programa por las cuales la interfaz se moverá para desempeñar su trabajo:

- **Carpeta Program**

  Es la carpeta principal y dondes e encuentran los archivos principales del programa. En ella se aloja todas las funciones principales y auxiliares del programa y las imágenes propias de la interfaz del programa.

- **Carpeta Imágenes**

  Es la carpeta de referencia donde se almacenan las imágenes a analizar. A pesar de ello el usuario podrá subir al programa cualquier imagen que se encuentre alojada en su PC. De manera predeterminada se incluyen con el programa unas imágenes de referencia.

- **Carpeta Proyectos guardados**

  Será en esta carpeta donde el usuario podrá guardar los datos de su proceso de análisis de imágenes para poder ser usado en un futuro. El archivo fruto del guardado del proyecto podrá ser transportado y utilizado

en cualquier otro PC por el mismo programa desarrollado.

4.2.2 Metodología de variables

La metodología empleada ha requerido de la utilización de numerosas variables que, a lo largo del desarrollo del programa deben ser:

- Definidas por el programa o por el usuario
- Empleadas para realizar cálculos
- Transportadas por las diferentes funciones del programa

Es por ello que se ha elegido la utilización de una variable global única capaz de almacenar toda la información del programa. Esta información es definida, modificada, transportada y eliminada a lolargo de toda la utilización del programa. La variable elegida se le ha dado el nombre de "Ima", se trata de una variable tipo estructura la cual posee distintas capas de información las cuales se explicarán a continuación.

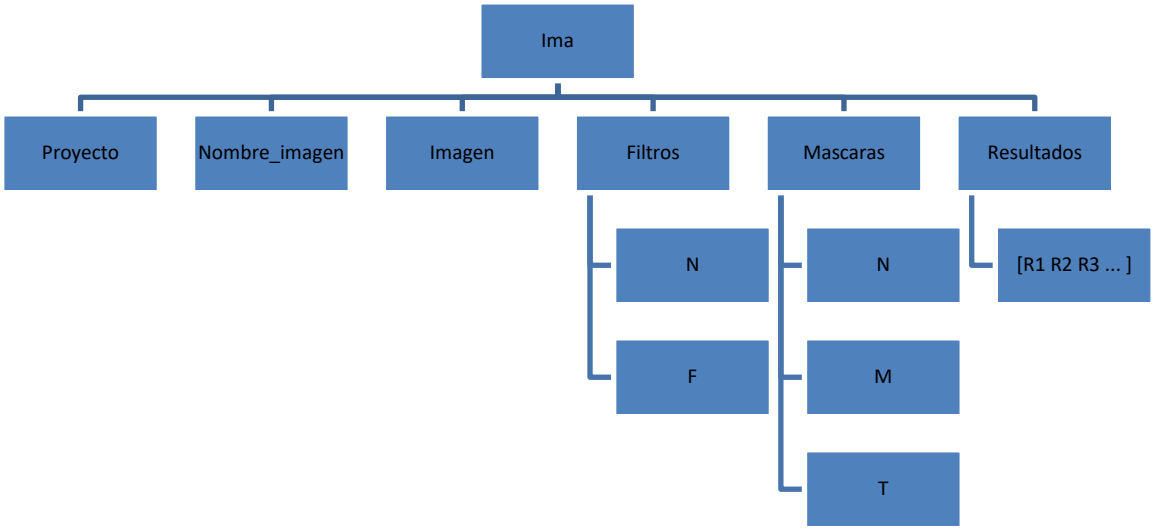


Figura 10

La variable "Ima" posee 6 capas distintas orientadas al almacenado de datos específicos de las acciones realizadas por el usuario en el programa. Las capas son:

- **Proyecto**

  Esta subvariable de tipo string está destinada a guardar el nombre del proyecto completo una vez es guardado por el usuario.

- **Nombre_imagen**

  Esta subvariable de tipo string está destinada a guardar el nombre de la imagen que el usuario introdujo para ser analizada.

- **Imagen**

Subvariable de tipo matricial en donde se almacenará la imagen subida por el usuario al programa. Esto hace que no sea necesario guardar por separado la imagen analizada y los datos del programa. En una sola variable se dispone también de la imagen que el usuario analizó.

- **Filtros**

Esta subvariable almacenará los datos de los filtros introducidos por el usuario. Es de tipo estructura y está formada a su vez por dos subvariables más:

  - N: que almacenará el número de filtros definidos por el usuario
  - F: subvariable de tipo *cell* que almacenará los nombres de los filtros y su valor como vector (ejemplo: E5= [-1 -2 0 2 1]).

- **Máscaras**

Esta subvariable almacenará los datos de las máscaras definidas el usuario. Es de tipo estructura y está formada a su vez por tres subvariables más:

  - N: que almacenará el número de máscaras definidas por el usuario
  - F: subvariable de tipo *cell* que almacenará los nombres de las máscaras, su tipología (simples o dobles) y su definición según la combinación de los filtros que la componen (ejemplo: Mascara 3: E5/S5).
  - T: Subvariable de tipo *cell* que almacenará el texto mostrado por pantalla de los nombres de las máscaras (ejemplo: "Máscara 3: E5*S5").

- **Resultados**

Esta subvariable de tipo *cell* almacenará los vectores resultados de la convolución de cada máscara con la imagen a analizar.

4.2.3 Metodología de funciones

Si bien todo el código se ha desarrollado con forme a funciones especializadas en tareas concretas cabe diferenciar dos tipos de funciones: funciones principales y funciones auxiliares.

Las funciones principales son todas aquellas que aportan información relevante en el proceso seguido por la metodología de Law. Estas funciones son fundamentales y sin ellas no se podría implementar de forma completa el método descrito. Se trata de funciones que calculan la convoluciones con las máscaras, el cardado de los filtros, etc. En su totalidad están definidas dentro del script *Prototipo_26* y son:

- Función B_Aceptar_Filtros: Esta función se ejecuta una vez que el usuario pulsa la opción "Aceptar filtros" dentro de la pestaña de filtros en la interfaz principal. Una vez introducidos los filtros, sus nombres y sus términos según la mejor opción que el usuario haya elegido (de manera manual o de manera automática) los filtros están listos para ser almacenados dentro de la variable principal *Ima.* Este es uno de los pasos más importantes en todo el programa ya que se guardarán los filtros con los que posteriormente el usuario podrá definir las máscaras. Este proceso completo ha dividido en tres partes diferenciadas para su correcta codificación:
  - 1) Lectura de los filtros introducidos: Como primer paso se procede a leer el número de términos de los filtros introducidos (3, 5, 7, 9 u 11), este número también marcará cuantos

filtros se deberán de leer de esta pestaña. Acto seguido se leerán cada nombre de filtro introducido y sus términos correspondientes.

- 2) Guardado de los filtros: Una vez leídos los filtros se procederá a su guardado dentro de la variable tipo estructura *Ima*. Este proceso de guardado es fundamental y se realizará conforme a la estructura de variables planteadas en el punto anterior "*4.2.2 Metodología de variables*". Esto servirá para futuros procesos de transporte y lectura de estas variables en los siguientes pasos del programa.
- 3) Mostrar resumen y comprobación de filtros: Si todo este proceso de lectura y guardado se realiza de forma correcta y sin fallos se procederá a mostrar un check de correcto en la parte de resumen dentro de la interfaz principal. A su vez se mostrarán los nombres de cada filtro (sin sus términos) para que el usuario pueda ver de forma clara qué filtros están cargados y activos (aptos para poder definir máscaras con ellos) en todo momento.

```
function B_Aceptar_Filtros_Callback(hObject, eventdata, handles)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Función para aceptar los filtros introducidos %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Recuperar Ima %
    Ima=get(handles.figure1,'UserData');
  %%%%%%%%%%%%%%%%%%

  % Recuperación datos pestañas %
    Datos_Pestanas=get(handles.Para_Pestanas,'UserData');
    Matriz_BEN=Datos_Pestanas.Pestana2.Matriz_BEN;
    Matriz_TN=Datos_Pestanas.Pestana2.Matriz_TN;
    Matriz_TI=Datos_Pestanas.Pestana2.Matriz_TI;
    Matriz_TF=Datos_Pestanas.Pestana2.Matriz_TF;
    Matriz_BEF=Datos_Pestanas.Pestana2.Matriz_BEF;
    Lista_Terminos=Datos_Pestanas.Pestana2.Lista_Terminos;
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Guardar todos los filtros definidos en la variable Ima %
     N_Terminos=(2*get(Lista_Terminos,'Value'))+1;
     Ima.Filtros.N=N_Terminos;
     set(handles.figure1,'UserData',Ima);
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Proceso de lectura de filtros introducidos %
     faltan=0;
     for i=[1:1:N_Terminos]
           if isempty(get(Matriz_TF{1,i},'String'))
               faltan=1;
           else
%               MF(i,:)=str2num(get(Matriz_TF{1,i},'String'));
           end
     end

     Filtros_Correctos=0;
     if faltan==0
       for i=[1:1:N_Terminos]
           Ima.Filtros.F{i,1}=get(Matriz_TN{1,i},'String');
           Ima.Filtros.F{i,2}=str2num(get(Matriz_TF{1,i},'String'));
       end
       Filtros_Correctos=1;
       set(handles.T_Requisito2,'Visible','on');
       Requisito2=imread('Bien.png');
       axes(handles.T_Requisito2);
       imshow(Requisito2)
     else
         warndlg('Faltan terminos por rellenar','Mensaje');
     end

     set(handles.figure1,'UserData',Ima);
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Mostrar texto en pantalla de resumenes %
    Texto={['Número de términos de los filtros: ',num2str(Ima.Filtros.N)],Ima.Filtros.F{:,1}};
    set(handles.Texto_Resumen_Filtros,'String',Texto)
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  assignin('base','Ima',Ima);
```

- Función B_Aceptar_Máscaras: Esta función se ejecuta una vez que el usuario pulsa la opción "Aceptar máscaras" dentro de la pestaña de máscaras en la interfaz principal. Una vez cargados los filtros y definidas las máscaras según los filtros elegidos por el usuario, las máscaras están listas para ser aplicadas a la imagen. Este es un paso importante en el programa ya que se calcularán todas las gráficas de resultados que se mostrarán en la pestaña posterior. Este proceso completo ha dividido en tres partes diferenciadas para su correcta codificación:
    - 1) Lectura de las máscaras introducidas: Tras la definición de cada máscara, el programa deberá leer cada máscara existente en el listado de máscaras. Cada máscara estará compuesta de un nombre, una definición de máscara (ejemplo: "*L5E5*"), una matriz, producto de la definición de los filtros que la componen y una matriz de resultados producto de la aplicación de dicha matriz a la imagen a analizar.
    - 2) Guardado de las máscaras: Una vez leídas las máscaras y sus caracteristicas se procederá a su guardado dentro de la variable tipo estructura *Ima*. Este proceso de guardado es fundamental y se realizará conforme a la estructura de variables planteadas en el punto anterior "*4.2.2 Metodología de variables*".
    - 3) Mostrar resumen, comprobar y aplicar máscaras: Si todo este proceso de lectura y guardado se realiza de forma correcta y sin fallos se procederá a mostrar un check de correcto en la parte de resumen dentro de la interfaz principal. A su vez se mostrarán los nombres de cada máscara (sin sus términos) para que el usuario pueda ver de forma clara qué máscaras están cargadas y activas (se podrán usar para elaborar grafica de resultados) en todo momento. Llegados a este punto se procederá a calcular todas las posibles combinaciones de pares de máscaras según el número de máscaras introducidas. Este resultado condicionará el número de gráficas de puntos que se muestren en la pestaña posterior. Acto seguido se calculará la aplicación de cada máscara a la imagen y su representación en graficas de pares de máscaras. Estos resultados serán mostrados como gráficas en la siguiente pestaña.

```
function B_Aceptar_Mascaras_Callback(hObject, eventdata, handles)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Función para aceptar las máscaras definidas y calcular resultados %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Recuperar Ima %
    Ima=get(handles.figure1,'UserData');
  %%%%%%%%%%%%%%%%%%

  % Recuperación datos pestañas %
    Datos_Pestanas=get(handles.Para_Pestanas,'UserData');
    Matriz_BEN=Datos_Pestanas.Pestana2.Matriz_BEN;
    Matriz_TN=Datos_Pestanas.Pestana2.Matriz_TN;
    Matriz_TI=Datos_Pestanas.Pestana2.Matriz_TI;
    Matriz_TF=Datos_Pestanas.Pestana2.Matriz_TF;
    Matriz_BEF=Datos_Pestanas.Pestana2.Matriz_BEF;
    Lista_Terminos=Datos_Pestanas.Pestana2.Lista_Terminos;

    Texto_Mascaras=Datos_Pestanas.Pestana3.Texto_Mascaras;
    N_Mascaras=Datos_Pestanas.Pestana3.N_Mascaras;
    TxtMascaras=get(Texto_Mascaras,'UserData');
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Borrado de las graficas de proyecciones anteriores %
    %G1=axes('Parent',Pestanas(1),'Position',[0 0 1 1]);  %Colocar graficas en pestaña 1%
    %set(G1,'Visible','off');
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Mostrar texto en pantalla de resumenes %
    if isempty(TxtMascaras)
        msgbox('No existen máscaras','Error')
    else
         Texto={['Número de máscaras: ',num2str(Ima.Mascaras.N)],TxtMascaras{:,1}};
         set(handles.Texto_Resumen_Mascaras,'String',Texto)

    end
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Creación de gráficas pestaña 4 para proyecciones de caracteristicas %
     NMascaras=Ima.Mascaras.N;
%      NMascaras=6;
     Espacio_filas=((1-(1/NMascaras))*10)/100;
     Espacio_columnas=Espacio_filas;
     Tamano=(1-(Espacio_filas*(NMascaras+1)))/NMascaras;
     InicioX=0.1;
     InicioY=0.1;
     PosX=InicioX;
     PosY=InicioY;
     Graficas_fila=NMascaras-1;
     Graficas_columna=NMascaras-1;
     Resultados=Calcula_Resultados(Ima);
     Rx=1;
     Ry=2;
     for i=[1:1:Graficas_fila]
        PosY=InicioY;
        for j=[1:1:Graficas_columna]

Graficas_proyecciones{j,i}=axes('Parent',Datos_Pestanas.Pestana4.Marco,'PickableParts','all','HitTest','on
','ButtonDownFcn',{@BAGP_Callback,Ima,Rx,Ry},'Position',[PosY PosX Tamano, Tamano]);  %Colocar graficas en
pestaña 1%
             set(Graficas_proyecciones{j,i},'Visible','on');
             axes(Graficas_proyecciones{j,i})
             plot(Resultados{1,Rx},Resultados{1,Ry},'.')
             xlabel(['R',num2str(Rx)]);
             ylabel(['R',num2str(Ry)]);
             set(Graficas_proyecciones{j,i},'ButtonDownFcn',{@BAGP_Callback,handles,Rx,Ry})
             PosY=PosY++Tamano+Espacio_columnas;
             Ry=Ry+1;
        end
        Rx=Rx+1;
        Ry=Ry-Graficas_columna+1;
        PosX=PosX++Tamano+Espacio_filas;
        Graficas_columna=Graficas_columna-1;
```

- Función Generar: Esta función se ejecuta una vez que el usuario pulsa la opción "Generar" dentro de la zona de resumen en la interfaz principal. Esta opción solo estará disponible si los filtros han sido cargados y son rectos, y si las máscaras han sido aplicadas y son correctas. Una vez ejecutada esta función se mostrará una pequeña interfaz de formulario para que el usuario introduzca el nombre con el que desea guardar el proyecto. Hecho esto el proyecto completo, junto con la matriz de resultados será guardado en la carpeta de Proyectos guardados expuesta en el punto *4.2.1 Metodología de carpetas.*

```
function Generar_Callback(hObject, eventdata, handles)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Función para generar y guardar la matriz de resultados %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Recuperar Ima %
    Ima=get(handles.figure1,'UserData');
  %%%%%%%%%%%%%%%%%

  % Recuperación datos pestañas %
    Datos_Pestanas=get(handles.Para_Pestanas,'UserData');
    Grupo_Pestanas=Datos_Pestanas.Grupo_Pestanas;
    Datos_Pestanas.Grupo_Pestanas.SelectedTab;
    Pestanas(1)=Datos_Pestanas.Pestana1.Marco;
    Pestanas(2)=Datos_Pestanas.Pestana2.Marco;
    Pestanas(3)=Datos_Pestanas.Pestana3.Marco;
    Pestanas(4)=Datos_Pestanas.Pestana4.Marco;
    Pestanas(5)=Datos_Pestanas.Pestana4.Marco;

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Creacción de la matriz de resultados %
    Resultados=[];
    for i=[1:1:Ima.Mascaras.N]
        V=Ima.Resultados{1,i};
        Resultados=[Resultados;V'];
    end
    Ima.Matriz_resultados=Resultados;
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Guardar datos en Ima %
    set(handles.figure1,'UserData',Ima);
    assignin('base','Ima',Ima);
  %%%%%%%%%%%%%%%%%%%%%%%%
  Guardar_ClickedCallback(hObject, eventdata, handles)
```

- Función para generar informe: Esta función realizará un informe en pdf que será guardado en la carpeta de "*Informes de archivos*"

```
function Generar_Informe_Callback(hObject, eventdata, handles)
% hObject    handle to Generar_Informe (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    W=what;
    DirectorioA=W.path;
    DirectorioN=[DirectorioA '\Informes de archivos'];
    DirectorioN=[DirectorioN '\'];

    opts.format='pdf';
    opts.showCode=false;
    opts.outputDir=DirectorioN;

    publish('Crea_informe',opts)
```

A pesar del empleo de estas funciones principales se vuelve imprescindible el uso de funciones auxiliares que faciliten el uso de la interfaz por parte del usuario. Estas funciones están fuera del proceso descrito por la metodología de Laws pero resultan de interés. Son las encargadas de labores como el guardado de los datos, el movimiento entre pestañas, la visualización de imágenes y gráficas, etc. Muchas de ellas están definidas dentro del script *Prototipo_26*y otras se encuentran fuera de este script para simplificar su uso. A continuación, se describe cada función auxiliar y su uso:

- Función Examinar: Esta función se emplea como resultado de pulsar el botón "Examinar" en la interfaz principal. La función abre la carpeta para cargar imágenes al programa. Una vez abierta la carpeta el usuario podrá seleccionar la imagen deseada (con formatos .png y . jpg). Seleccionado la imagen se procede a ser cargada en la variable principal *Ima* y se muestra dicha imagen en la pestaña "Imagen original" dentro de la interfaz principal.

```
function Examinar_Callback(hObject, eventdata, handles)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Función para el botón examinar %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Recuperar Ima %
    Ima=get(handles.figure1,'UserData');
  %%%%%%%%%%%%%%%%%%

  % Redirigir a directorio de imágenes %
    DirectorioA=cd();
    cd('..');
    DirectorioN=cd();
    cd([DirectorioN,'\Imagenes']);
    [Imagen_seleccionada,Directorio_Imagen]=uigetfile({'*.png; *.jpg'});
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Comprobación de la imagen introducida y cargado de la imagen %
    if Imagen_seleccionada==0
        set(handles.T_Requisito1,'Visible','off');
        cd(DirectorioA);
    else
        cd(DirectorioA);
        set(handles.T_Requisito1,'Visible','on');
        Requisito1=imread('Bien.png');
        axes(handles.T_Requisito1);
        imshow(Requisito1)

        set(handles.Nombre_imagen_seleccionada,'String',Imagen_seleccionada)
        cd(Directorio_Imagen);
        Imagen=imread(Imagen_seleccionada);
        set(handles.Examinar,'UserData',Imagen);

    % Recuperación datos pestañas %
      Datos_Pestanas=get(handles.Para_Pestanas,'UserData');
      Pestanas(1)=Datos_Pestanas.Pestana1.Marco;
      G1=Datos_Pestanas.Pestana1.Grafica;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Colocación de la imagen original en Pestaña 1 %
       set(G1,'Visible','on');
       Ima.Nombre_imagen=Imagen_seleccionada;
       Ima.Imagen=Imagen;
       axes(G1)
       imshow(Imagen);  %Representar imagen
       axis off
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    set(handles.figure1,'UserData',Ima)
    assignin('base','Ima',Ima);
    cd(DirectorioA);
end
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

- Función Cargar: Esta función se desempeña como resultado de pulsar la opción de "Cargar" dentro de la barra de herramientas.

Esta función está destinada a cargar los datos guardados de procesos de análisis de imágenes anteriores realizados por el usuario. La función abrirá la carpeta de proyectos guardados para que el usuario pueda elegir qué proyecto desea cargar. Hecho esto se dispondrá a cargar la variable *Ima* del proyecto guardado en un nuevo proyecto que será cargado en la interfaz principal. De esta forma el usuario podrá cargar todas las variables que ya definió una vez (la imagen, los filtros, las máscaras, etc.) así como volver a ver los resultados de un proyecto anterior (gráfica de puntos, informe, etc.). A su vez el usuario podrá modificar datos de esos proyectos antiguos para ver cómo cambian los r

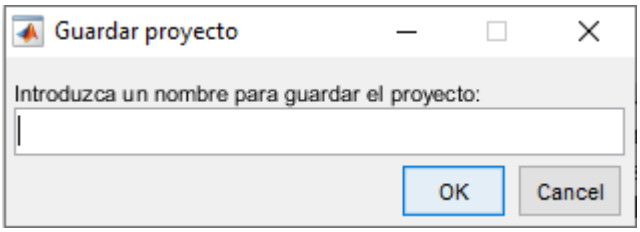
```
function Cargar_ClickedCallback(hObject, eventdata, handles)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Función para cargar un proyecto guardado %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Recuperación datos pestañas %
    Datos_Pestanas=get(handles.Para_Pestanas,'UserData');
    Grupo_Pestanas=Datos_Pestanas.Grupo_Pestanas;
    Datos_Pestanas.Grupo_Pestanas.SelectedTab;
    Pestanas(1)=Datos_Pestanas.Pestana1.Marco;
    Pestanas(2)=Datos_Pestanas.Pestana2.Marco;
    Pestanas(3)=Datos_Pestanas.Pestana3.Marco;
    Pestanas(4)=Datos_Pestanas.Pestana4.Marco;
    Pestanas(5)=Datos_Pestanas.Pestana4.Marco;

    G1=Datos_Pestanas.Pestana1.Grafica;

    Matriz_BEN=Datos_Pestanas.Pestana2.Matriz_BEN;
    Matriz_TN=Datos_Pestanas.Pestana2.Matriz_TN;
    Matriz_TI=Datos_Pestanas.Pestana2.Matriz_TI;
    Matriz_TF=Datos_Pestanas.Pestana2.Matriz_TF;
    Matriz_BEF=Datos_Pestanas.Pestana2.Matriz_BEF;
    Lista_Terminos=Datos_Pestanas.Pestana2.Lista_Terminos;

    Texto_Mascaras=Datos_Pestanas.Pestana3.Texto_Mascaras;
    N_Mascaras=Datos_Pestanas.Pestana3.N_Mascaras;
    TxtMascaras=get(Texto_Mascaras,'UserData');

    N_Terminos=[1;1;1;1;2;2;3;3;4;4;5];
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Redirigir a carpeta de proyectos guardados %
    DirectorioA=cd();
    cd('..');
    DirectorioA2=cd();
    DirectorioN=[DirectorioA2,'\Proyectos guardados'];
    cd(DirectorioN);
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Cargado del proyecto %
    [Proyecto_seleccionado,Directorio_Proyecto]=uigetfile({'*.mat'});
    if Proyecto_seleccionado==0
        cd(DirectorioA);
    else
        load(Proyecto_seleccionado);
        cd(DirectorioA);
        assignin('base','Ima',Ima);
        set(handles.figure1,'UserData',Ima);
        cd(DirectorioA);
        Texto=['Proyecto ',Proyecto_seleccionado,'.mat',' cargado correctamente'];
        Bien=imread('Bien.png');
        msgbox(Texto,'Cargado correctamente','custom',Bien);
```

```
        % Cargar imagen %
          if isempty(Ima.Imagen)
             set(handles.T_Requisito1,'Visible','off');
          else
             set(handles.Nombre_imagen_seleccionada,'String',Ima.Nombre_imagen)
             set(handles.T_Requisito1,'Visible','on');
             Requisito1=imread('Bien.png');
             axes(handles.T_Requisito1);
             imshow(Requisito1)

             set(handles.Examinar,'UserData',Ima.Imagen);
             % Colocación de la imagen original en Pestaña 1 %
               set(G1,'Visible','on');
               axes(G1)
               imshow(Ima.Imagen);  %Representar imagen
               axis off
             %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
          end

        %%%%%%%%%%%%%%%%%%

        % Cargar filtros %
            if isempty(Ima.Filtros)

            else
              set(Lista_Terminos,'Value',N_Terminos(Ima.Filtros.N))
              PopupMenu_Terminos_Callback(hObject, eventdata, handles)
              for i=[1:1:Ima.Filtros.N]
                  set(Matriz_TF{1,i},'String',num2str(Ima.Filtros.F{i,2}));
                  set(Matriz_TN{1,i},'String',Ima.Filtros.F{i,1});
              end

             % Actualización de datos %
               Datos_Pestanas.Pestana2.Matriz_TF=Matriz_TF;
               set(handles.Para_Pestanas,'UserData',Datos_Pestanas);
               assignin('base','Datos_Pestanas',Datos_Pestanas);
             %%%%%%%%%%%%%%%%%%%%%%%%%%

             % Mostrar texto en pantalla de resumenes %
               Texto={['Número de términos de los filtros:
',num2str(Ima.Filtros.N)],Ima.Filtros.F{:,1}};
               set(handles.Texto_Resumen_Filtros,'String',Texto)
               Requisito2=imread('Bien.png');
               axes(handles.T_Requisito2);
               imshow(Requisito2)
             %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            end
        %%%%%%%%%%%%%%%%%%%

        % Cargar máscaras %
          if isempty(Ima.Mascaras)

          else
           % Mostrar mascaras en la interfaz %
             Ima=evalin('base','Ima');

             set(Texto_Mascaras,'String',Ima.Mascaras.T);
             set(Texto_Mascaras,'UserData',Ima.Mascaras.T);
           %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            % Actualización de datos %
              set(handles.Para_Pestanas,'UserData',Datos_Pestanas);
              assignin('base','Datos_Pestanas',Datos_Pestanas);
            %%%%%%%%%%%%%%%%%%%%%%%%%%
            set(handles.T_Requisito3,'Visible','on');
            Requisito3=imread('Bien.png');
            axes(handles.T_Requisito3);
            imshow(Requisito3)
            % Guardar datos en Ima %
              set(handles.figure1,'UserData',Ima);
              assignin('base','Ima',Ima);
            %%%%%%%%%%%%%%%%%%%%%%%%%%
            B_Aceptar_Mascaras_Callback(hObject, eventdata, handles)
          end
        %%%%%%%%%%%%%%%%%%%%

        % Cargar resultados %
        %%%%%%%%%%%%%%%%%%%%%%%
    end
  %%%%%%%%%%%%%%%%%%%%%%%%%%
```

- Función Guardar: Esta función se desempeña como resultado de pulsar la opción de "Guardar" dentro de la barra de herramientas.

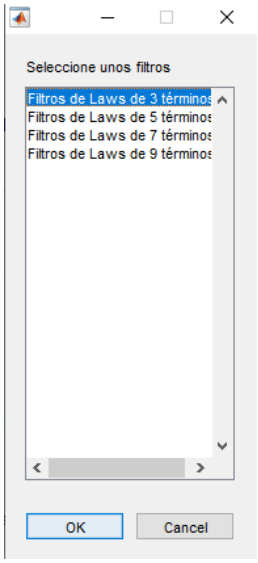
Esta función está destinada a guardar todos los datos que definen un proyecto creado por el usuario. Como se habló anteriormente, se ha intentado que todas las variables fundamentales (tanto datos de entrada, parámetros etc. como datos de salida, resultados) se han guardado en la variable principal *Ima*. De esta forma resulta fácil guardar todo un proyecto ya que solo se necesita almacenar externamente al programa esta variable *Ima*.

La forma de guardado se ha elegido como un archivo ".mat" ya que este tipo de archivos están especializados al almacenamiento de variables del programa Matlab. Al ejecutarse esta función se puede observar una pequeña interfaz tipo formulario en la cual se pregunta al usuario el nombre que desea imponer a su proyecto para poder ser guardado.




```
function Guardar_ClickedCallback(hObject, eventdata, handles)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Función para guardar el proyecto %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Recuperar Ima %
    Ima=get(handles.figure1,'UserData');
  %%%%%%%%%%%%%%%%%%

  % Pedir nombre para guardar proyecto %
    Nombre_guardar=inputdlg('Introduzca un nombre para guardar el proyecto:','Guardar proyecto', [1
50]);
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Proceso de guardado %
    if isempty(Nombre_guardar)

    else
       Ima.Nombre=Nombre_guardar{1,1}
       DirectorioA=cd();
       cd('..');
       DirectorioA2=cd();
       DirectorioN=[DirectorioA2,'\Proyectos guardados'];

       cd(DirectorioN);
       save(Nombre_guardar{1,1},'Ima');
       cd(DirectorioA);
       Bien=imread('Bien.png');
       Texto=['Proyecto guardado on nombre: ',Nombre_guardar{1,1},'.mat'];
       msgbox(Texto,'Guardado correctamente','custom',Bien);

    end
  %%%%%%%%%%%%%%%%%%%%%%%%

  % Guardar datos en Ima %
    set(handles.figure1,'UserData',Ima);
    assignin('base','Ima',Ima);
  %%%%%%%%%%%%%%%%%%%%%%%%
```

- Función Cargar_Filtros_pred: Esta función se ejecuta cada vez que el usuario presiona la opción de filtros predeterminados en la pestaña de filtros. Ya que los filtros definidos por Laws son comúnmente usados se ha optado por incluir una función que permita su cargado rápido. Una vez pulsada la opción de filtros predeterminados aparecerá una ventana indicativa para que el usuario pueda elegir qué filtro predeterminado de Laws quiere cargar.



Las opciones presentadas por defecto son: Filtros de Laws de 3, 7, 9 y 11 términos. Estas son las opciones más comunes y permiten al usuario no tener que definir cada vez los filtros de Laws que desea utilizar.

En el código de esta función se puede ver como primeramente se recuperan los datos de la pestaña a usar y posteriormente se definen todos los filtros de Laws anteriormente indicados. Finalmente, con la elección realizada por el usuario se cargan los filtros elegidos a la segunda pestaña de la interfaz principal.

```
function B_Cargar_Filtros_pred_Callback(hObject, eventdata, handles)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Función para cargar los filtros predeterminados de Laws %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Recuperación datos pestañas %
    Datos_Pestanas=get(handles.Para_Pestanas,'UserData');
    Matriz_BEN=Datos_Pestanas.Pestana2.Matriz_BEN;
    Matriz_TN=Datos_Pestanas.Pestana2.Matriz_TN;
    Matriz_TI=Datos_Pestanas.Pestana2.Matriz_TI;
    Matriz_TF=Datos_Pestanas.Pestana2.Matriz_TF;
    Matriz_BEF=Datos_Pestanas.Pestana2.Matriz_BEF;
    Lista_Terminos=Datos_Pestanas.Pestana2.Lista_Terminos;
%     Lista_FM1=Datos_Pestanas.Pestana3.Lista_FM1;
%     Lista_FM2=Datos_Pestanas.Pestana3.Lista_FM2;
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Cargado del filtro predeterminado elegido en la pestaña 3 %
    N_filtro_pred=Filtro_Predetermidado();
%   N_Terminos=5;
    Nombres_Laws{1}={'L3','E3','S3'};
    Nombres_Laws{2}={'L5','E5','S5','W5','R5'};
    Nombres_Laws{3}={'L7','E7','S7','W7','R7','O7'};
    Nombres_Laws{4}={'L9','E9','S9','R9','W5','F9','F9','F9','F9'};
    Nombres_Laws{5}={'L11','E11','S11','R11','W11','F11','F11','F11','F11','F11','F11'};

    N_Terminos_Laws=[1;2;3;4;5];

    MF_Laws{1}=[1 2 1;-1 2 1;-1 0 -1;-1 -2 1;1 -4 1];
    MF_Laws{2}=[1 4 6 4 1;-1 -2 0 2 1;-1 0 2 0 -1;-1 2 0 -2 1;1 -4 6 -4 1];
    MF_Laws{3}=[1 6 15 20 15 6 1;-1 -4 -5 0 5 4 1;-1 -2 1 4 1 -2 -1;-1 0 3 0 -3 0 1;1 -2 -1 4 -1 -2 1;-1 6 -15 20
-15 6 -1];

    if N_filtro_pred==0

    else
       Nombres_Laws=Nombres_Laws{N_filtro_pred};
       MF_Laws=MF_Laws{N_filtro_pred};
       set(Lista_Terminos,'Value',N_Terminos_Laws(N_filtro_pred))
       PopupMenu_Terminos_Callback(hObject, eventdata, handles)
%        if N_Terminos==5
          for i=[1:1:length(MF_Laws(1,:))]
               set(Matriz_TF{1,i}    ,'String',num2str(MF_Laws(i,:)));
               set(Matriz_TN{1,i},'String',Nombres_Laws{i});
          end
%        end
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Una vez volcados los datos de los filtros predeterminados el usuario también puede cambiar sus nombres o el valor de sus términos a su gusto. Esto permite realizar un estudio de cómo varían los cambios en ciertos términos de cada filtro de Laws.

- Función Exportar_filtros: Esta función se ha creado para aportar la opción de guardar los filtros definidos en unos proyectos y poder ser usados en otros. Esta función se ejecuta cuando se pulsa la opción “guardar filtros” de la segunda pestaña de la interfaz principal. Una vez pulsada esta opción, se presenta al usuario un pequeño formulario para que sea introducido el nombre con el que se quiere que sea guardados los filtros. Acto seguido se procede a la lectura tanto del tamaño de los filtros, sus nombres como de sus términos. Estos datos se guardan en forma de un archivo “.mat” en la carpeta que el usuario desee para poder ser cargado posteriormente.

```
  function B_Exportar_filtros_Callback(hObject, eventdata, handles)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Función para guardar los filtros introducidos %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Recuperación datos pestañas %
    Datos_Pestanas=get(handles.Para_Pestanas,'UserData');
    Matriz_BEN=Datos_Pestanas.Pestana2.Matriz_BEN;
    Matriz_TN=Datos_Pestanas.Pestana2.Matriz_TN;
    Matriz_TI=Datos_Pestanas.Pestana2.Matriz_TI;
    Matriz_TF=Datos_Pestanas.Pestana2.Matriz_TF;
    Matriz_BEF=Datos_Pestanas.Pestana2.Matriz_BEF;
    Lista_Terminos=Datos_Pestanas.Pestana2.Lista_Terminos;
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Recuperar Ima %
    Ima=get(handles.figure1,'UserData');
  %%%%%%%%%%%%%%%%%%
  N=get(Lista_Terminos,'Value'); % Leo el numero de filtros introducidos
  Exportar_Filtros(Datos_Pestanas,N);
```

- Función Importar_filtros: Esta función se ejecuta cada vez que el usuario elige la opción "Cargar filtros" dentro de la pestaña de filtros en la interfaz principal. Este código se plantea como complemento a la función anterior. Una vez que el usuario ha guardado los filtros que desea este podrá volverlos a cargar en el mismo proyecto o en otro distinto.

```
   function B_Importar_filtros_Callback(hObject, eventdata, handles)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Función para cargar filtros al programa %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Recuperación datos pestañas %
    Datos_Pestanas=get(handles.Para_Pestanas,'UserData');
    Matriz_BEN=Datos_Pestanas.Pestana2.Matriz_BEN;
    Matriz_TN=Datos_Pestanas.Pestana2.Matriz_TN;
    Matriz_TI=Datos_Pestanas.Pestana2.Matriz_TI;
    Matriz_TF=Datos_Pestanas.Pestana2.Matriz_TF;
    Matriz_BEF=Datos_Pestanas.Pestana2.Matriz_BEF;
    Lista_Terminos=Datos_Pestanas.Pestana2.Lista_Terminos;
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Recuperar Ima %
    Ima=get(handles.figure1,'UserData');
  %%%%%%%%%%%%%%%%%%

  % Cargado de filtros %
    [N,Nombres,Terminos]=Importar_Filtros();
    if N==0

    else
      Lista_Terminos_Filtros=[0 0 1 0 5 0 3 0 4 0 5];
      set(Lista_Terminos,'Value',Lista_Terminos_Filtros(N))
      PopupMenu_Terminos_Callback(hObject, eventdata, handles)
      for i=[1:1:N]
          set(Datos_Pestanas.Pestana2.Matriz_TN{1,i},'String',Nombres{1,i});
          set(Datos_Pestanas.Pestana2.Matriz_TF{1,i},'String',Terminos{1,i});
      end
    end
  %%%%%%%%%%%%%%%%%%%%%%%%

  % Actualización de datos %
    set(handles.Para_Pestanas,'UserData',Datos_Pestanas);
    assignin('base','Datos_Pestanas',Datos_Pestanas);
  %%%%%%%%%%%%%%%%%%%%%%%%%%
```

La diferencia entre esta función de cargado de filtros y la función de cargados de filtros predeterminados es clara. En la primera se requiere que el usuario previamente halla guardado los filtros que posteriormente serán cargados. Sin embargo, la opción de cargado de los filtros predeterminados no requiere de archivo externo, sino que los datos de cada filtro de Laws están almacenados en el mismo código.

- Función BAGP: Esta función se plantea para poder abrir en una ventana alterna una gráfica de resultados seleccionada. Una vez cargados los filtros y aplicadas las máscaras se obtendrán en la pestaña de *Proyecciones y características* las distintas gráficas de puntos fruto de enfrentar pares de resultados. Usualmente el número de gráficas que deben aparecer en esta pestaña es elevado y dependerá del número de máscaras aplicadas. Por ello, se plantea una opción para ampliar las gráficas que el usuario desee para que se puedan ver con claridad y poder analizar sus resultados.

```
  function BAGP_Callback(hObject, eventdata, handles,Rx,Ry)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Función para ampliar gráfica de resultados seleccionada %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 % Recuperar Ima%
   Ima=get(handles.figure1,'UserData');
   Resultados=Ima.Resultados
 %%%%%%%%%%%%%%%%%

 % Crear nueva figura con la gráfica selecionada %
   figure
   plot(Resultados{1,Rx},Resultados{1,Ry},'.')
   xlabel(['R',num2str(Rx)]);
   ylabel(['R',num2str(Ry)]);
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

El código de esta función consiste en una ampliación del gráfico seleccionado en una ventana alternativa.

- Función BAGI: De forma predeterminada en la primera pestaña se muestra la imagen original a analizar. El usuario siempre tiene acceso a esta imagen en esta pestaña a lo largo del programa, pero puede resultar de utilidad poder representar la imagen original en una ventana alternativa independiente de la interfaz principal.

```
  function BAGP_Callback(hObject, eventdata, handles,Rx,Ry)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Función para ampliar gráfica de resultados seleccionada %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 % Recuperar Ima%
   Ima=get(handles.figure1,'UserData');
   Resultados=Ima.Resultados
 %%%%%%%%%%%%%%%%%

 % Crear nueva figura con la gráfica selecionada %
   figure
   plot(Resultados{1,Rx},Resultados{1,Ry},'.')
   xlabel(['R',num2str(Rx)]);
   ylabel(['R',num2str(Ry)]);
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

El código consiste en representar en una ventana alterna la imagen original.

- Función Boton_BAM: Esta función es la encargada de realizar el proceso de definición de una nueva máscara a partir de los filtros introducidos. La función se ejecuta cada vez que se elige la opción de "Añadir máscara" en la pestaña de máscaras. En este momento se despliegan una serie de interfaces que facilitan la definición de las máscaras al usuario.

```
function Boton_BAM_Callback(hObject, eventdata, handles,fila)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Función para añadir una nueva máscara %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Recuperación datos pestañas %
    Datos_Pestanas=get(handles.Para_Pestanas,'UserData');
    Matriz_BEN=Datos_Pestanas.Pestana2.Matriz_BEN;
    Matriz_TN=Datos_Pestanas.Pestana2.Matriz_TN;
    Matriz_TI=Datos_Pestanas.Pestana2.Matriz_TI;
    Matriz_TF=Datos_Pestanas.Pestana2.Matriz_TF;
    Matriz_BEF=Datos_Pestanas.Pestana2.Matriz_BEF;
    Lista_Terminos=Datos_Pestanas.Pestana2.Lista_Terminos;

    Texto_Mascaras=Datos_Pestanas.Pestana3.Texto_Mascaras;
    N_Mascaras=Datos_Pestanas.Pestana3.N_Mascaras;
    TxtMascaras=get(Texto_Mascaras,'UserData');
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Recuperar Ima %
    Ima=get(handles.figure1,'UserData');
  %%%%%%%%%%%%%%%%%%

    assignin('base','Ima',Ima);

  % Introducción de la nueva máscara %
   Grado_Mascara=questdlg('La máscara nueva es:','Máscara nueva','Simple','Doble','No thank you');

   if isempty(Grado_Mascara)
   else
       for i=[1:1:Ima.Filtros.N]
           str{i,1}=Ima.Filtros.F{i,1};
       end

       if strcmp(Grado_Mascara,'Simple')==1  % si la máscara es simple
          waitfor(GUI_Mascara_Simple)
       end

       if strcmp(Grado_Mascara,'Doble')==1  % si la máscara es doble
          waitfor(GUI_Mascara_Doble());
       end
   end
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Mostrar mascaras en la interfaz %
     Ima=evalin('base','Ima');
%      TxtMascaras={TxtMascaras;Ima.Mascaras.T}
     set(Texto_Mascaras,'String',Ima.Mascaras.T);
     set(Texto_Mascaras,'UserData',Ima.Mascaras.T);
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Actualización de datos %
    set(handles.Para_Pestanas,'UserData',Datos_Pestanas);
    assignin('base','Datos_Pestanas',Datos_Pestanas);
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Guardar datos en Ima %
    set(handles.figure1,'UserData',Ima);
    assignin('base','Ima',Ima);
  %%%%%%%%%%%%%%%%%%%%%%%%
```

El código consiste en desplegar estas interfaces y recoger la información introducida. Primero se recoge si la máscara introducida es simple o doble. Dependiendo de la opción se desplegará una interfaz u otra. En esta segunda interfaz se presentará al usuario todas las opciones posibles para definir su máscara. En esta parte el código recogerá la definición de la máscara y esta será mostrada

en la lista de máscaras introducidas existente en la tercera pestaña de la interfaz principal.

Por definición cada máscara introducida se definirá como: "*Máscara <Nº>: <combinación de filtros>*" para poder referenciar bien cada máscara. El término *Nº* se define automáticamente para cada máscara según su posición dentro de la lista de máscaras.

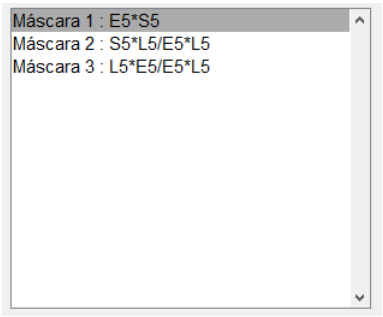
- Función Boton_BEM: Una vez introducidas las máscaras se plantea la adición de una opción para poder eliminar las máscaras que se deseen. El usuario podría haber errado en la introducción de alguna máscara o simplemente podría querer eliminar una máscara en específico. Esta función se ejecuta cada vez que el usuario pulsa la opción de "Eliminar máscara" dentro de la pestaña de máscaras. La primera condición que se debe dar para que se debe dar para la ejecución del código es que existan máscaras definidas, de no ser así no se podrá eliminar ninguna máscara. Una vez cumplida esta opción se presentará al usuario en una pequeña interfaz la misma lista de máscaras existentes en la pestaña de máscaras. El usuario deberá elegir qué máscara quiere eliminar. Hecho esto la máscara será eliminada y el listado de máscaras será reordenado. De esta forma si existen 5 máscaras y se elimina la número 3, las máscara de la posición 1 y 2 no se verían afectadas por el cambio pero las máscaras 4 y 5 se verían elevadas respecto a su posición original.

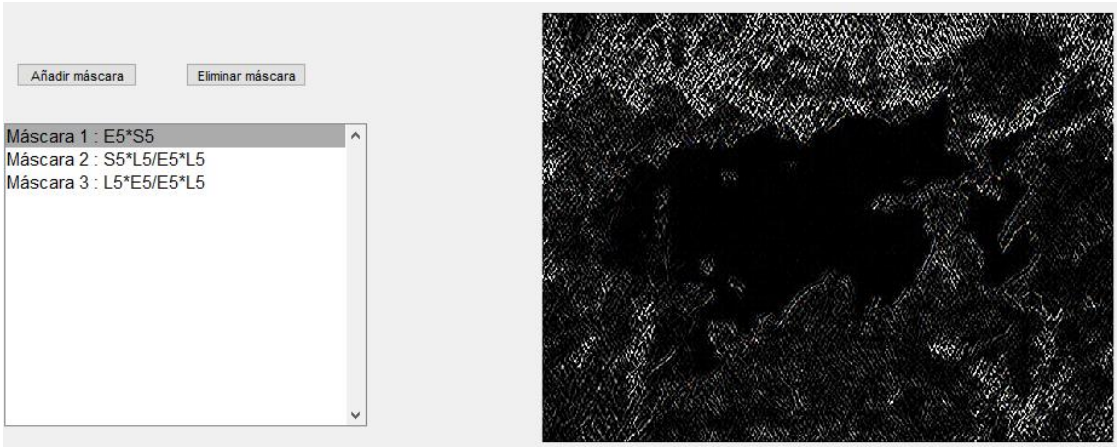
```
 function Boton_BEM_Callback(hObject, eventdata, handles,fila)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Función para eliminar una máscara %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Recuperación datos pestañas %
    Datos_Pestanas=get(handles.Para_Pestanas,'UserData');
    Matriz_BEN=Datos_Pestanas.Pestana2.Matriz_BEN;
    Matriz_TN=Datos_Pestanas.Pestana2.Matriz_TN;
    Matriz_TI=Datos_Pestanas.Pestana2.Matriz_TI;
    Matriz_TF=Datos_Pestanas.Pestana2.Matriz_TF;
    Matriz_BEF=Datos_Pestanas.Pestana2.Matriz_BEF;
    Lista_Terminos=Datos_Pestanas.Pestana2.Lista_Terminos;

    Texto_Mascaras=Datos_Pestanas.Pestana3.Texto_Mascaras;
    N_Mascaras=Datos_Pestanas.Pestana3.N_Mascaras;
    TxtMascaras=get(Texto_Mascaras,'UserData');
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Recuperar Ima %
    Ima=get(handles.figure1,'UserData');
  %%%%%%%%%%%%%%%%%

    assignin('base','Ima',Ima);

  % Selección de la máscara a eliminar %

    Pos_Mascara=Borrar_Mascara(Ima);
    if Pos_Mascara==0

    else
       Ima=evalin('base','Ima');
       Ima.Mascaras.N=Ima.Mascaras.N-1;
       Ima.Mascaras.T(Pos_Mascara,:)=[];
       Ima.Mascaras.M(Pos_Mascara,:)=[];
    end

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Mostrar mascaras en la interfaz %
%      TxtMascaras={TxtMascaras;Ima.Mascaras.T}
    if Pos_Mascara==0

    else
      if Ima.Mascaras.N==0
           set(Texto_Mascaras,'String','No existen máscaras definicas');
           set(Texto_Mascaras,'UserData','No existen máscaras definicas')
      else
        set(Texto_Mascaras,'String',Ima.Mascaras.T);
        set(Texto_Mascaras,'UserData',Ima.Mascaras.T);
      end
    end
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

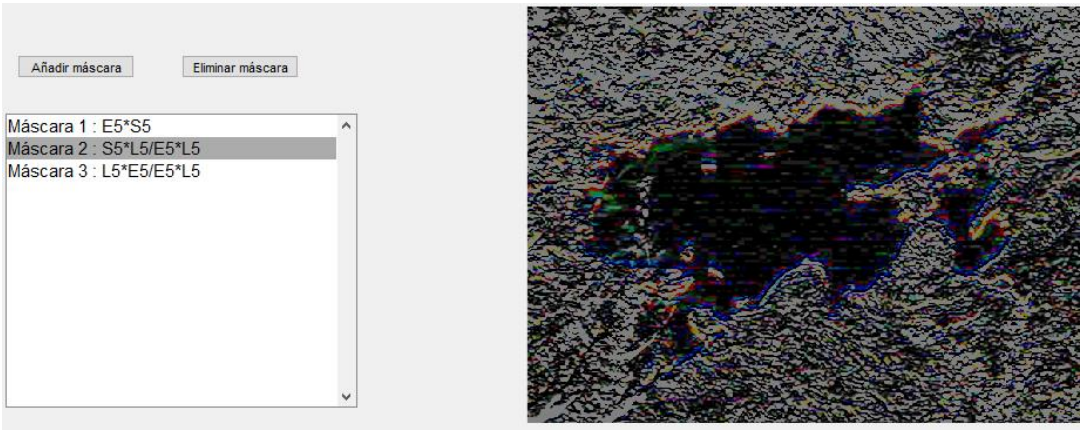
```
% Actualización de datos %
  if Pos_Mascara==0

  else
  set(handles.Para_Pestanas,'UserData',Datos_Pestanas);
  assignin('base','Datos_Pestanas',Datos_Pestanas);
  end
%%%%%%%%%%%%%%%%%%%%%%%%%

% Guardar datos en Ima %
  set(handles.figure1,'UserData',Ima);
  assignin('base','Ima',Ima);
%%%%%%%%%%%%%%%%%%%%%%%%%
```

- Función_Boton_BLM: Como se ha explicado anteriormente la aplicación de una máscara a una imagen resalta ciertos aspectos de la misma. Si una máscara especializada en detectar líneas rectas es aplicada a una imagen dará como resultado una imagen en las líneas rectas están resaltadas. Por ello se ha programado una función capaz de aplicar una máscara seleccionada a la imagen original y mostrar por pantalla el resultado.

  El código se ejecuta con la selección de la máscara deseada dentro del listado de máscaras existentes ya definidas.







El código lee la máscara seleccionada del listado y se procede a su convolución con la imagen original. Hecha esta convolución se ha reservado un espacio dentro de la pestaña de máscaras de la interfaz principal para albergar una gráfica donde serán representados los resultados. De esta forma el usuario puede visualizar la aplicación directa de cada máscara introducida a su imagen para poder ajustar sus términos o eliminarla.

```
function Boton_BLM_Callback(hObject, eventdata, handles)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Función para aplicar máscara seleccionada a la imágen %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Recuperación datos pestañas %
    Datos_Pestanas=get(handles.Para_Pestanas,'UserData');
    Matriz_BEN=Datos_Pestanas.Pestana2.Matriz_BEN;
    Matriz_TN=Datos_Pestanas.Pestana2.Matriz_TN;
    Matriz_TI=Datos_Pestanas.Pestana2.Matriz_TI;
    Matriz_TF=Datos_Pestanas.Pestana2.Matriz_TF;
    Matriz_BEF=Datos_Pestanas.Pestana2.Matriz_BEF;
    Lista_Terminos=Datos_Pestanas.Pestana2.Lista_Terminos;

    Texto_Mascaras=Datos_Pestanas.Pestana3.Texto_Mascaras;
    N_Mascaras=Datos_Pestanas.Pestana3.N_Mascaras;
    TxtMascaras=get(Texto_Mascaras,'UserData');
    Pestanas(3)=Datos_Pestanas.Pestana3.Marco;
    G3=Datos_Pestanas.Pestana3.Grafica;
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Recuperar Ima %
    Ima=get(handles.figure1,'UserData');
    Imagen=Ima.Imagen;
  %%%%%%%%%%%%%%%%%%
```

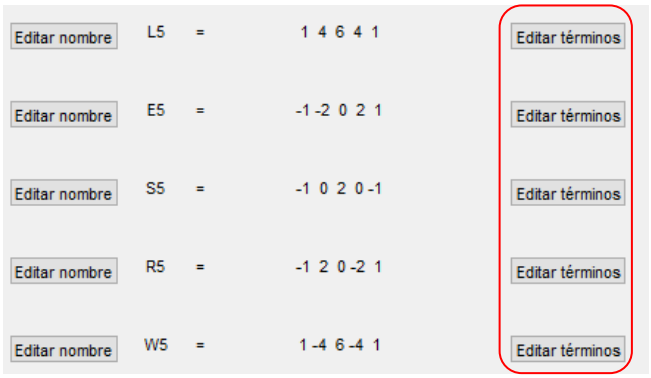
```
% Representación de la imagen con la mascara seleccionada %
   if Ima.Mascaras.N==0

   else
     N=get(Texto_Mascaras,'Value');
     if Ima.Mascaras.M{N,2}==1  % Si es una mascara simple
         for i=[1:1:Ima.Filtros.N]  % Busqueda del primer filtro simple
             if strcmp(Ima.Filtros.F{i,1},Ima.Mascaras.M{N,3})==1
                 F1=Ima.Filtros.F{i,2};
             end
         end
         for i=[1:1:Ima.Filtros.N]  % Busqueda del segundo filtro simple
             if strcmp(Ima.Filtros.F{i,1},Ima.Mascaras.M{N,4})==1
                 F2=Ima.Filtros.F{i,2};
             end
         end
         Mascara=F1'*F2;
         Imagen_Mascara=imfilter(Imagen,Mascara,'conv','symmetric');
     end

     if Ima.Mascaras.M{N,2}==2  % Si es una mascara doble
         for i=[1:1:Ima.Filtros.N]  % Busqueda del primer filtro doble
             if strcmp(Ima.Filtros.F{i,1},Ima.Mascaras.M{N,3})==1
                 F1=Ima.Filtros.F{i,2};
             end
         end
         for i=[1:1:Ima.Filtros.N]  % Busqueda del segundo filtro doble
             if strcmp(Ima.Filtros.F{i,1},Ima.Mascaras.M{N,4})==1
                 F2=Ima.Filtros.F{i,2};
             end
         end
         for i=[1:1:Ima.Filtros.N]  % Busqueda del tercer filtro doble
             if strcmp(Ima.Filtros.F{i,1},Ima.Mascaras.M{N,5})==1
                 F3=Ima.Filtros.F{i,2};
             end
         end
         for i=[1:1:Ima.Filtros.N]  % Busqueda del cuarto filtro doble
             if strcmp(Ima.Filtros.F{i,1},Ima.Mascaras.M{N,6})==1
                 F4=Ima.Filtros.F{i,2};
             end
         end
         Mascara1=F1'*F2;
         Mascara2=F3'*F4;
         Imagen_Mascara1=imfilter(Imagen,Mascara1,'conv','symmetric');
         Imagen_Mascara2=imfilter(Imagen,Mascara2,'conv','symmetric');
         Imagen_Mascara=(Imagen_Mascara1+Imagen_Mascara2)./2;
     end
   end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Colocación de la imagen con mascara en Pestaña 3 %
  if Ima.Mascaras.N==0

  else
   set(G3,'Visible','on');
   axes(G3)
   imshow(Imagen_Mascara);  %Representar imagen
   axis off
  end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Actualización de datos %
 if Ima.Mascaras.N==0

 else
  Datos_Pestanas.Pestana3.Texto_Mascaras=Texto_Mascaras;
  Datos_Pestanas.Pestana3.N_Mascaras=N_Mascaras;
  set(handles.Para_Pestanas,'UserData',Datos_Pestanas);
  assignin('base','Datos_Pestanas',Datos_Pestanas);
 end
%%%%%%%%%%%%%%%%%%%%%%%%%%


% Guardar datos en Ima %
  set(handles.figure1,'UserData',Ima);
  assignin('base','Ima',Ima);
%%%%%%%%%%%%%%%%%%%%%%%%%
```

- Función Boton_ENF: Esta función se ejecuta cuando el usuario pulsa el botón "Editar nombre" para cada filtro existente dentro de la segunda pestaña de la interfaz principal. Una vez elegido el número de términos de los filtros se representan en esta pestaña tantos botones de edición de nombre de filtro como términos elegidos. Cada botón realizará la misma operación para el filtro al que está asociado.

Esta función despliega un pequeño formulario en el cual el usuario puede modificar el nombre del filtro que ha seleccionado. Una vez definido el nombre este se muestra en la pestaña junto a los términos del filtro seleccionado.

```
function Boton_ENF_Callback(hObject, eventdata, handles,fila)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Función para editar el nombre del filtro seleccionado %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Recuperación datos pestañas %
    Datos_Pestanas=get(handles.Para_Pestanas,'UserData');
    Matriz_BEN=Datos_Pestanas.Pestana2.Matriz_BEN;
    Matriz_TN=Datos_Pestanas.Pestana2.Matriz_TN;
    Matriz_TI=Datos_Pestanas.Pestana2.Matriz_TI;
    Matriz_TF=Datos_Pestanas.Pestana2.Matriz_TF;
    Matriz_BEF=Datos_Pestanas.Pestana2.Matriz_BEF;
    Lista_Terminos=Datos_Pestanas.Pestana2.Lista_Terminos;
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Introducción del nuevo nombre %
    NombreF=inputdlg('Introduzca un nombre para el filtro:','Editar nombre filtro', [1 50]);
    set(Matriz_TN{1,fila},'String',NombreF{1,1});
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Actualización de datos %
    Datos_Pestanas.Pestana2.Matriz_TN=Matriz_TN;
    set(handles.Para_Pestanas,'UserData',Datos_Pestanas);
    assignin('base','Datos_Pestanas',Datos_Pestanas);
  %%%%%%%%%%%%%%%%%%%%%%%%%%
```

- Función Boton_ETF: Esta función es análoga a la función anterior de edición de nombre de filtros. Esta función se ejecuta cuando el usuario pulsa el botón "Editar términos" para cada filtro existente dentro de la segunda pestaña de la interfaz principal. Una vez elegido el número de términos de los filtros se representan en esta pestaña tantos botones de edición de términos de filtro como términos elegidos. Cada botón realizará la misma operación para el filtro al que está asociado. Esta función despliega un pequeño formulario en el cual el usuario puede introducir los términos del filtro que ha seleccionado. Una vez definido esto, se muestra en la pestaña junto al nombre del filtro seleccionado.




```
function Boton_ETF_Callback(hObject, eventdata, handles,fila)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Función para editar los términos del filtro seleccionado %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Recuperación datos pestañas %
    Datos_Pestanas=get(handles.Para_Pestanas,'UserData');
    Matriz_BEN=Datos_Pestanas.Pestana2.Matriz_BEN;
    Matriz_TN=Datos_Pestanas.Pestana2.Matriz_TN;
    Matriz_TI=Datos_Pestanas.Pestana2.Matriz_TI;
    Matriz_TF=Datos_Pestanas.Pestana2.Matriz_TF;
    Matriz_BEF=Datos_Pestanas.Pestana2.Matriz_BEF;
    Lista_Terminos=Datos_Pestanas.Pestana2.Lista_Terminos;
%     Lista_FM1=Datos_Pestanas.Pestana3.Lista_FM1;
%     Lista_FM2=Datos_Pestanas.Pestana3.Lista_FM2;
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Introducción del nuevo término %
    Filtro=inputdlg('Introduzca los términos del nuevo filtro:','Editar filtro', [1 50]);
    set(Matriz_TF{1,fila},'String',Filtro{1,1});
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Actualización de datos %
    Datos_Pestanas.Pestana2.Matriz_TF=Matriz_TF;
    set(handles.Para_Pestanas,'UserData',Datos_Pestanas);
    assignin('base','Datos_Pestanas',Datos_Pestanas);
  %%%%%%%%%%%%%%%%%%%%%%%%%
```

4.2.4 Metodología de interfaces

Para mejorar la experiencia de uso del usuario y simplificar los posibles errores en el desarrollo y utilización del programa se ha optado por la creación de varias interfaces complementarias. Cada interfaz se entra en una serie de tareas concretas que se apoyaran entre sí para poder elaborar el proceso de análisis de la imagen.

Para la creación estas interfaces se ha empleado la herramienta "GUIDE" provista en MATLAB y la cual permite crear de forma gráfica una interfaz con distintos elementos como botones, gráficas, selectores, imágenes, etc. que después podrá ser programada. La metodología habitual de creación de interfaces comienza con la elaboración gráfica del diseño básico de la interfaz. Este diseño podrá ser modificado con el código posterior. En esta fase se incluirán los gráficos, botones y herramientas principales. Como segunda etapa se plantea la asignación de tareas concretas del código a cada gráfico, botón y herramienta dispuesta. Esto provoca un trasvase del control del código (introducción de variables, selección de opciones) a la interfaz y por ende al usuario.

A continuación, se muestra la interrelación de las distintas interfaces y cómo se traspasan y cargan los datos:

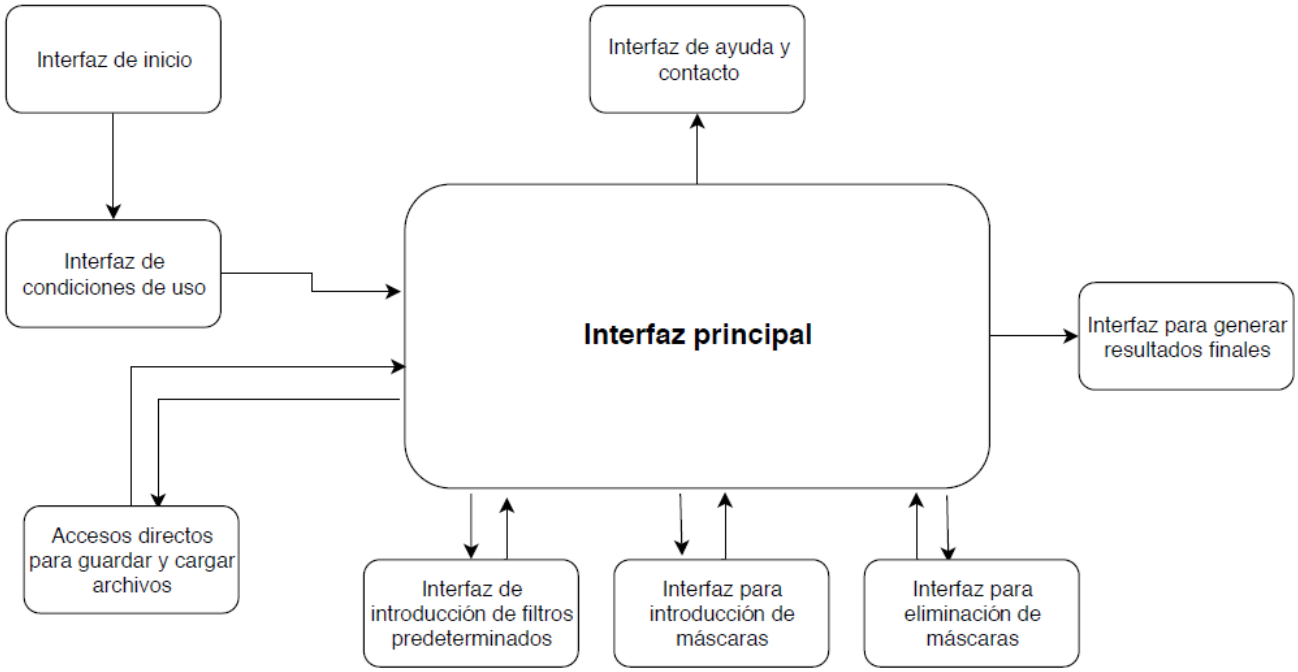


Figura 11

Cada interfaz además de realizar su tarea sirve como organizador de la información tanto para el usurario final como para el desarrollador al dividir la metodología global en pasos más sencillos. Se presenta a continuación un listado de las interfaces programadas:

- **Interfaz principal** (*Prototipo_26*): Es la interfaz destinada al proceso de introducción de los parámetros necesarios para el análisis de la imagen mediante la metodología de Laws. En ella se podrá seleccionar la imagen que se quiere analizar, sus filtros, definir las máscaras y generar la matriz de resultados para futuros pasos de clasificación.

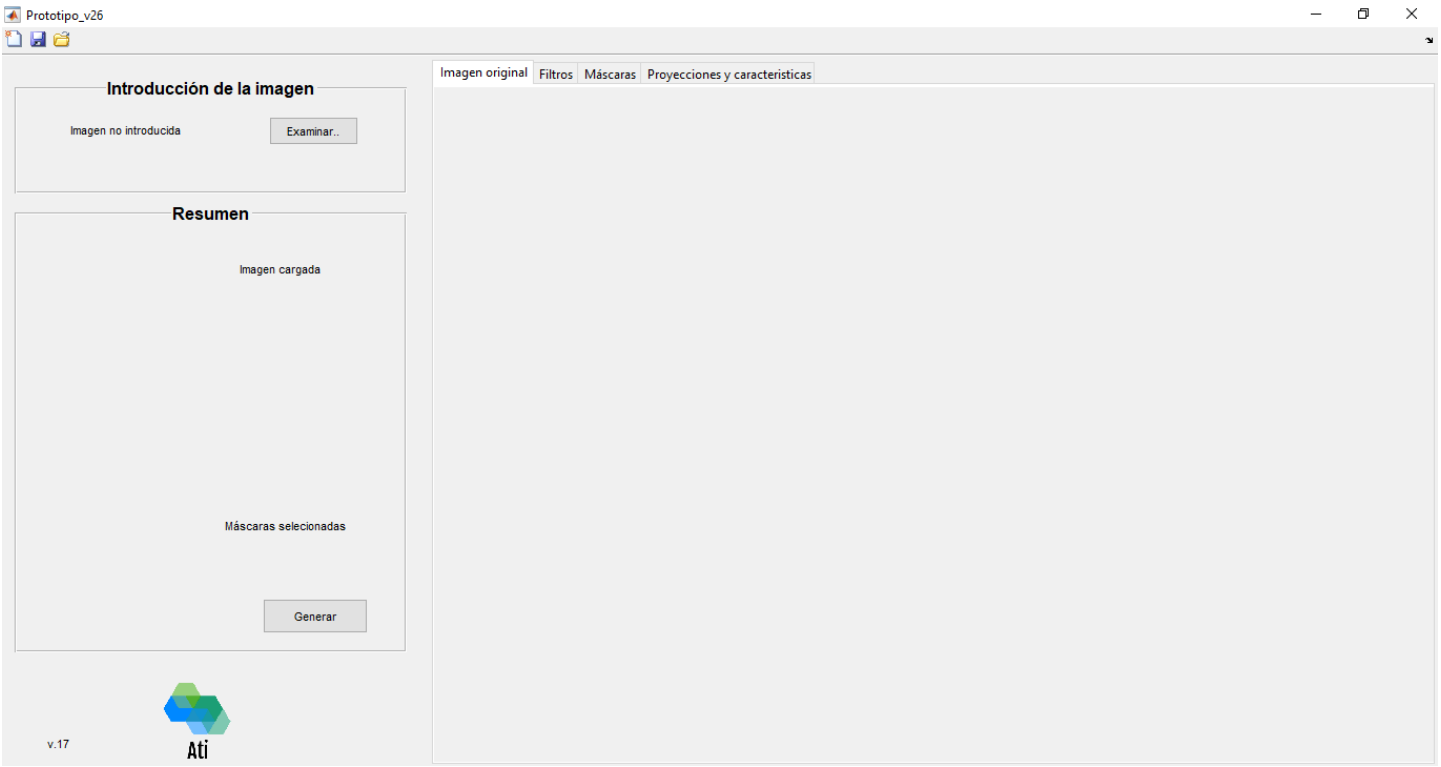


Figura 12

- **Interfaz de inicio** (*Entrada.m*): Es la interfaz inicial cargada al inicio del programa. Además de servir como presentación, se emplea esta interfaz como inicializador de todas las variables necesarias además de centrar al programa MATLAB en la carpeta principal de este analizador (carpeta program).

Figura 13

- **Interfaz de condiciones de uso** (*Condiciones_de_uso.m*): Es la segunda interfaz que se muestra después de la interfaz de inicio. En ella se presenta al usuario las condiciones de utilización y los requisitos mínimos para el correcto funcionamiento del programa desarrollado.

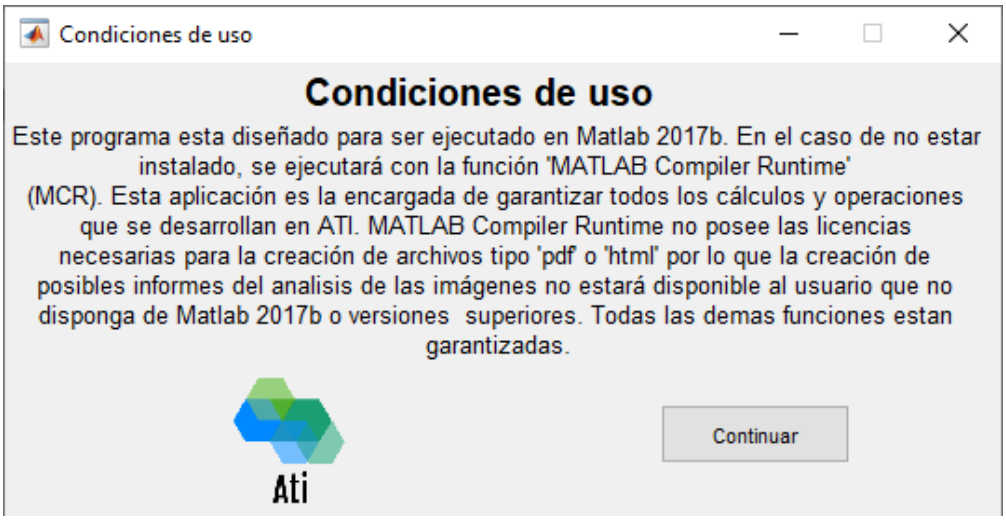


Figura 14

- **Interfaces de introducción de filtros y máscaras** (*GUI_Mascara_Simple.m* y *GUI_Mascara_Doble.m*): Este conjunto de interfaces se han creado para facilitar la introducción y definición de los filtros y las máscaras al usuario. Las interfaces se nutrirán de los datos introducidos previamente en la interfaz principal y cargarán en ella los datos definidos de filtros y máscaras.

-

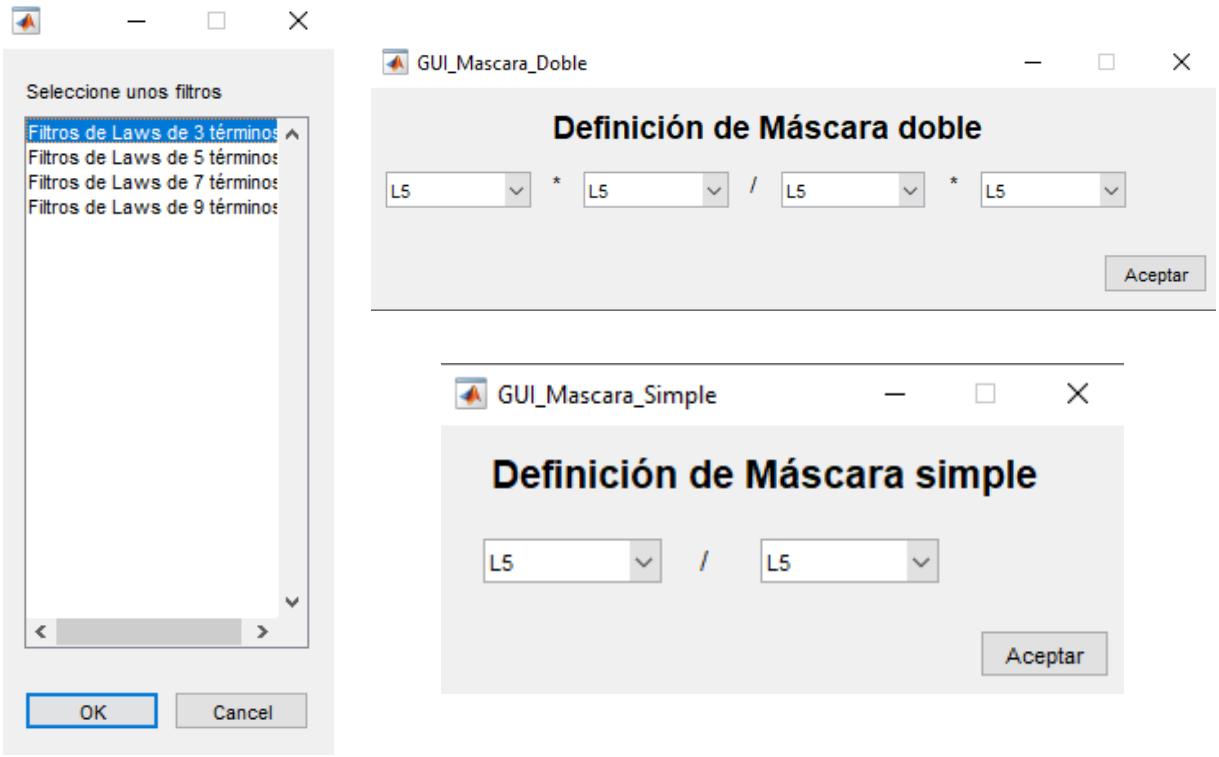


Figura 15

- **Interfaz de ayuda y contacto** (*AcercaDe.m*): Esta interfaz auxiliar se presenta como ayuda al usuario aportando información del programa, ámbito del proyecto, e información de contacto para facilitar ayuda.

4.2.5 Metodología de pestañas

Una de las partes principales y usadas de la interfaz principal son las pestañas. Se ha elegido la implementación y programación de pestañas dado la cantidad de espacio necesario para el uso del programa (datos a introducir por el usuario, filtros, visualización de imágenes y gráficas, etc.). Es por ello y como se implica en la metodología de la interfaz que la información se ha organizado por pestañas.

Dentro del programa principal cada pestaña debe ser programada y codificada con toda la información que en ella debe aparecer y que ella debe tratar. Es por ello que ha reservado una zona en el código principal para la creación de todas las pestañas y otras zonas para su uso.

```
% Definición de las pestañas %
   Grupo_Pestanas = uitabgroup('Parent',handles.figure1,'Position',[0.3 .005 .70 0.99]);
   Pestanas(1) = uitab(Grupo_Pestanas,'Title','Imagen original');
   Pestanas(2) = uitab(Grupo_Pestanas,'Title','Filtros');
   Pestanas(3) = uitab(Grupo_Pestanas,'Title','Máscaras');
   Pestanas(4) = uitab(Grupo_Pestanas,'Title','Proyecciones y caracteristicas');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

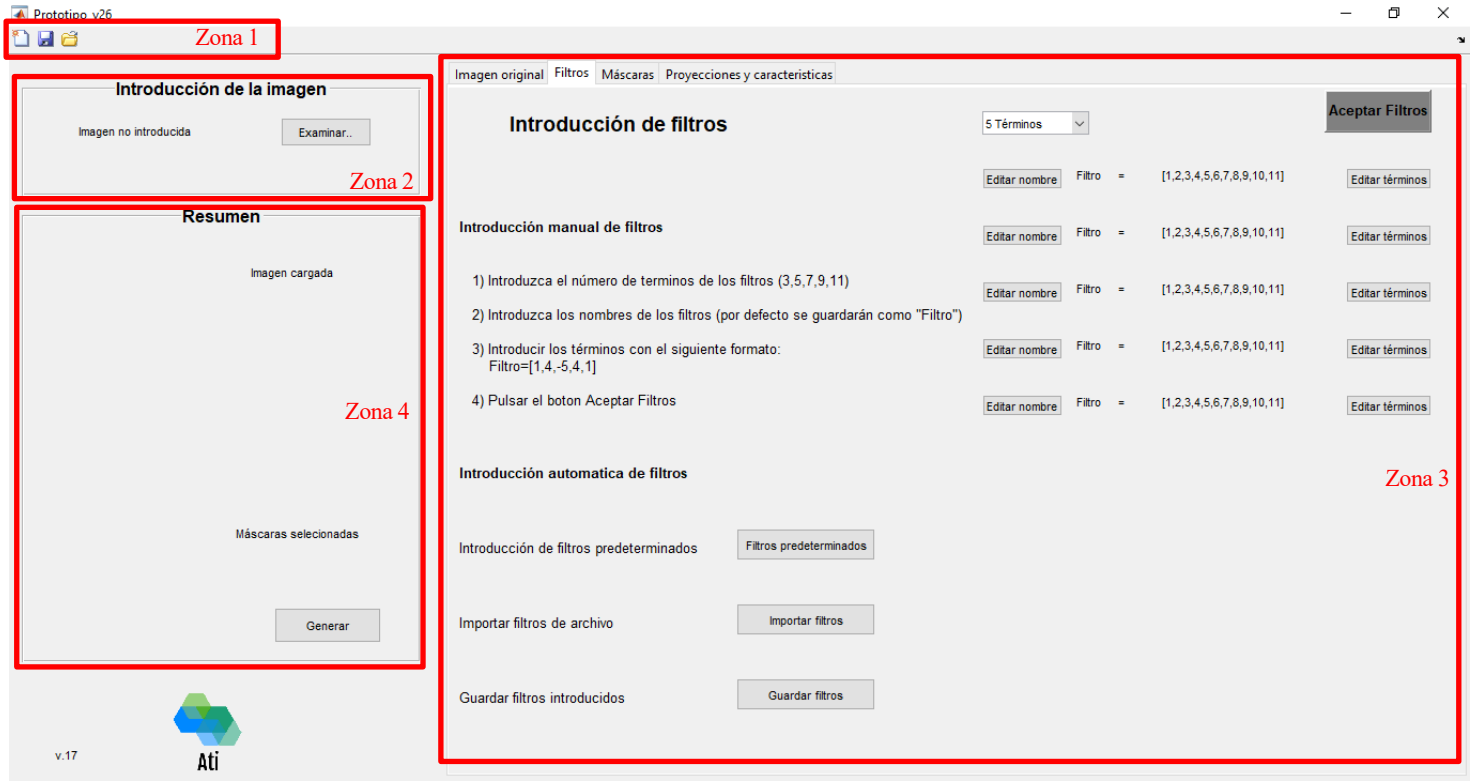
Cada pestaña debe ser creada y almacenada en una variable llamada *Datos_Pestanas*. En ella se definirán todas las pestañas, sus componentes y sus usos. Esta metodología se parece a la seguida con la variable *Ima* puesto que con solo una variable tipo estructura se puede almacenar la información de todas las pestañas. Es esta variable la que será transportada a lo largo del programa con el fin de ser utilizada según convenga. En ciertas funciones se leerán datos introducidos en ciertas pestañas (los cuales han sido almacenados en la variable *Datos_Pestanas*) y en otras se escribirán datos en las pestañas para poder mostrar resultados al usuario.

```
% Recuperación datos pestañas %
  Datos_Pestanas=get(handles.Para_Pestanas,'UserData');
  Matriz_BEN=Datos_Pestanas.Pestana2.Matriz_BEN;
  Matriz_TN=Datos_Pestanas.Pestana2.Matriz_TN;
  Matriz_TI=Datos_Pestanas.Pestana2.Matriz_TI;
  Matriz_TF=Datos_Pestanas.Pestana2.Matriz_TF;
  Matriz_BEF=Datos_Pestanas.Pestana2.Matriz_BEF;
  Lista_Terminos=Datos_Pestanas.Pestana2.Lista_Terminos;

  Texto_Mascaras=Datos_Pestanas.Pestana3.Texto_Mascaras;
  N_Mascaras=Datos_Pestanas.Pestana3.N_Mascaras;
  TxtMascaras=get(Texto_Mascaras,'UserData');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

4.3 Explicación de la interfaz principal

A continuación, se describirá cada parte de la interfaz o entorno interactivo que se ha diseñado en este programa. Como se ha expuesto anteriormente, este método implica multitud de parámetros a tener en cuenta como son la imagen a analizar, los filtros, las máscaras, etc. A su vez y por tratarse de un programa que analiza imágenes la parte visual ha cobreado mucha importancia. La interfaz creada se ha partido de un diseño con pestaña interactivas las cuales ayuden a la mejor introducción de los datos por parte del usuario.

La interfaz puede dividirse en 4 grandes partes diferenciadas:



- **Zona 1:** Configuración y ayuda global del programa

En esta zona se alojarán las herramientas principales de la configuración del programa. Estas herramientas podrán usarse para refrescar el programa y generar un nuevo proyecto, guardar el proyecto actual y ser cargado posteriormente y obtener una guía necesaria que sirva como referencia de la utilización del programa por parte del usuario.

- **Zona 2:** Introducción de la imagen a analizar

En esta zona se introducirá la imagen que se desea analizar. Por defecto el programa autodirige al usuario a la carpeta de imágenes de ejemplo, pero este podrá cargar cualquier imagen que se encuentra alojada en su PC. El programa puede trabajar sin problema con imágenes con extensión ".png" o ".jpg", no siendo válidas extensiones distintas.

- **Zona 3**: Introducción de parámetros y visualización de resultado

Es la zona principal del programa. En ella el usuario podrá introducir o cambiar todos los datos necesarios para un correcto análisis de la imagen tales como filtros, máscaras, etc. Al tratarse de una metodología progresiva y contener varios parámetros se ha optado por la división del proceso del método en distintas pestañas. Cada pestaña se centrará en un punto principal del método de Laws presentando facilitando al

usuario el uso de la interfaz. Las pestañas que se han considerado son:

- Pestaña 1: Imagen original

  Una vez cargada la imagen a analizar correctamente se podrá ver en esta pestaña teniéndola así de referencia para la correcta elección de los filtros y máscaras en los futuros pasos.

- Pestaña 2: Filtros

  Esta pestaña está destinada a la introducción de los filtros. El usuario posee dos opciones, introducción de los filtros de manera manual (definiendo término a término) e introducción de los filtros de manera automática. Se le presenta al usuario una breve descripción con ejemplos de cómo se introducirían los filtros de manera manual.

  El usuario podrá elegir en todo momento la longitud de los filtros a introducir siendo las posibilidades de 3, 5, 7, 9 u 11 términos en cada filtro. El nombre de cada filtro también podrá ser elegido por cada usuario para poder servirle de referencia en pasos futuros. Una vez introducidos los filtros el usuario podrá guardar esos filtros para poder ser usados en diferentes imágenes en otras ocasiones simplificando así la introducción de términos.

  Otra forma de introducción de filtros es de forma automática. Se le presenta al usuario unos filtros de Laws predeterminados de 3, 5, 7 y 9 términos. A su vez también podrá cargar filtros que haya guardado anteriormente para poder ser usados ahora. Esta metodología simplifica la introducción de cada termino de los filtros.

  Por último, una vez introducido o hechas las modificaciones necesarias a los filtros el usuario deberá pulsar el botón "Aceptar filtros" para que estos sean cargados correctamente en el programa permitiendo así continuar con los siguientes pasos.

- Pestaña 3: Máscaras

  Una vez introducidos de forma correcta todos los filtros el usuario deberá definir las máscaras utilizar. Como ya se definió anteriormente se presenta al usuario la posibilidad de definir máscaras simples o dobles. Para ello se deberá pulsar en "Añadir máscara". En este punto y una vez elegida la opción de máscara simple o doble, el usuario podrá definir la combinación más conveniente de los filtros anteriormente definidos. Cada filtro esta guardado con el nombre que el usuario le asignó.

  Una vez definida la máscara el usuario podrá visualizar la aplicación directa de dicha máscara a la imagen a analizar. Esto ayuda a entender mejor la utilidad de cada máscara definida.

  A su vez, el usuario también podrá eliminar cualquier máscara definida previamente con el botón "Eliminar máscara"

  Una vez todas las máscaras estén definidas, el usuario deberá pulsar el botón "Aceptar máscaras" para que estas sean cargadas correctamente en el programa permitiendo así continuar con los siguientes pasos.

- Pestaña 4: Proyecciones y características

  Estando las máscaras correctamente introducidas y el cálculo de convolución de cada máscara con la imagen realizado se podrá acceder a esta última pestaña. En ella se mostrarán, de forma piramidal los gráficos resultados correspondientes a todas las posibles confinaciones de cada máscara con la imagen por parejas. De este modo se obtendrán $\frac{(n-1)\cdot n}{2}$ gráficas de resultados por cada n máscaras que se apliquen a la imagen.

- **Zona 4:** Control global del proceso

  En la utilización del programa se llevan a cabo comprobaciones correspondientes a la correcta introducción de los datos por parte del usuario en la Zona 3. Es en esta Zona 4 donde se mostrará un resumen de los datos introducidos (imagen cargada, filtros a emplear, máscaras elegidas) en todo momento para le sea más fácil el desplazamiento a través de las pestañas por parte del usuario.

4.4 Flujo de operación del programa

En este apartado se explicará el flujo de operación principal del programa. Como aspecto fundamental es necesario entender que el programa se trata de una interfaz de entrada de datos por parte del usuario los cuales son tratados acordes con sus indicaciones (filtros y máscaras definidos, etc.) y que como producto final se obtiene una serie de resultados que el usuario puede extraer para poder generar un análisis posterior de la imagen introducida.

Con cada pestaña creada en la interfaz se intenta guiar al usuario por el flujo de operación para introducir los datos necesarios. La correcta introducción de estos datos es fundamental ya que permite al usuario conocer el proceso que seguirá la imagen introducida.

Como ya se ha comentado, el flujo interno del programa es el descrito en el capítulo anterior (3.- Método de Laws). A continuación, se expone un diagrama de flujo en el que se puede observar el flujo principal de desarrollo del programa. A su vez se exponen los flujos auxiliares que sirven de apoyo al flujo principal y que serán explicados extensamente el siguiente apartado.

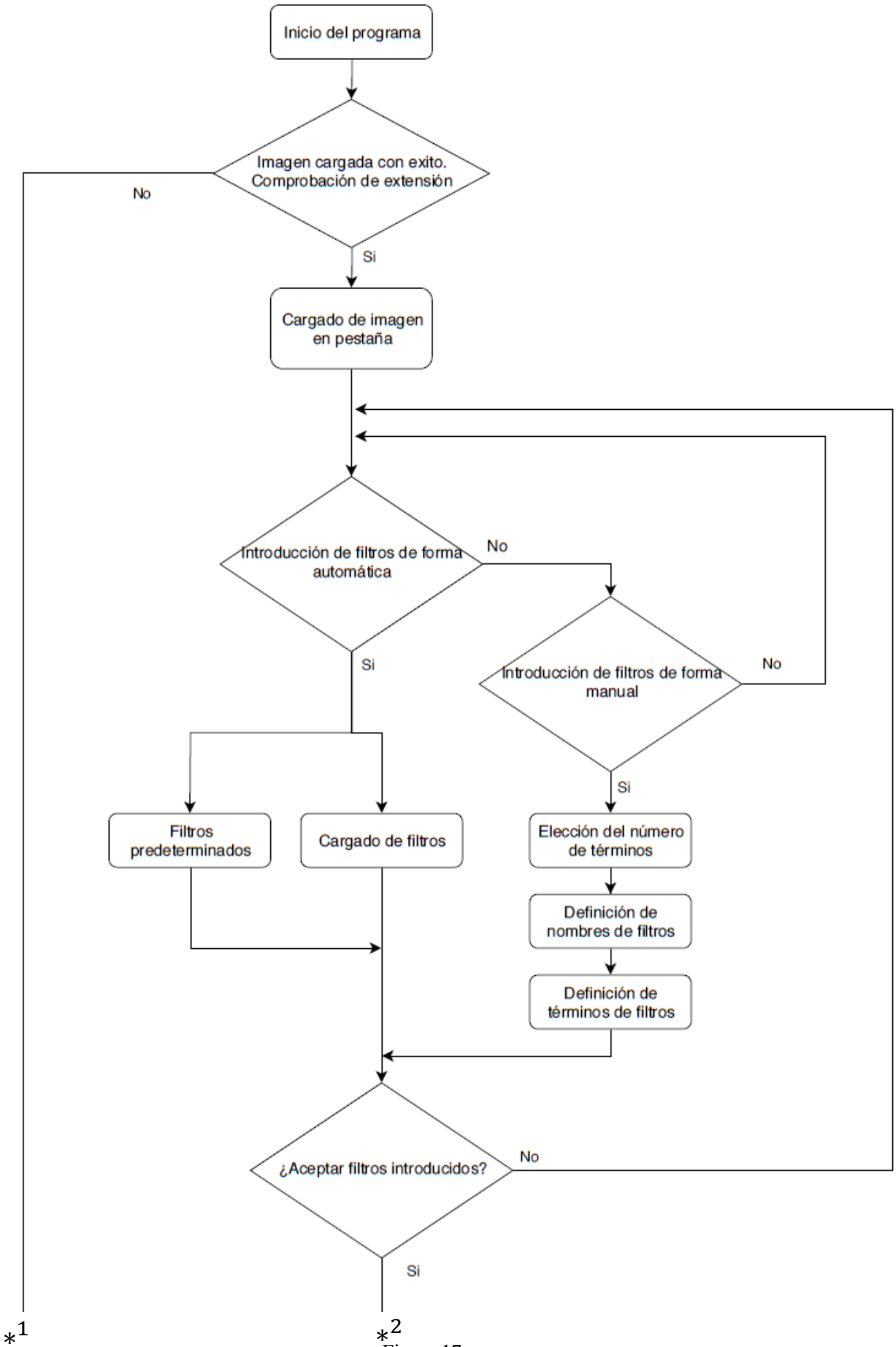
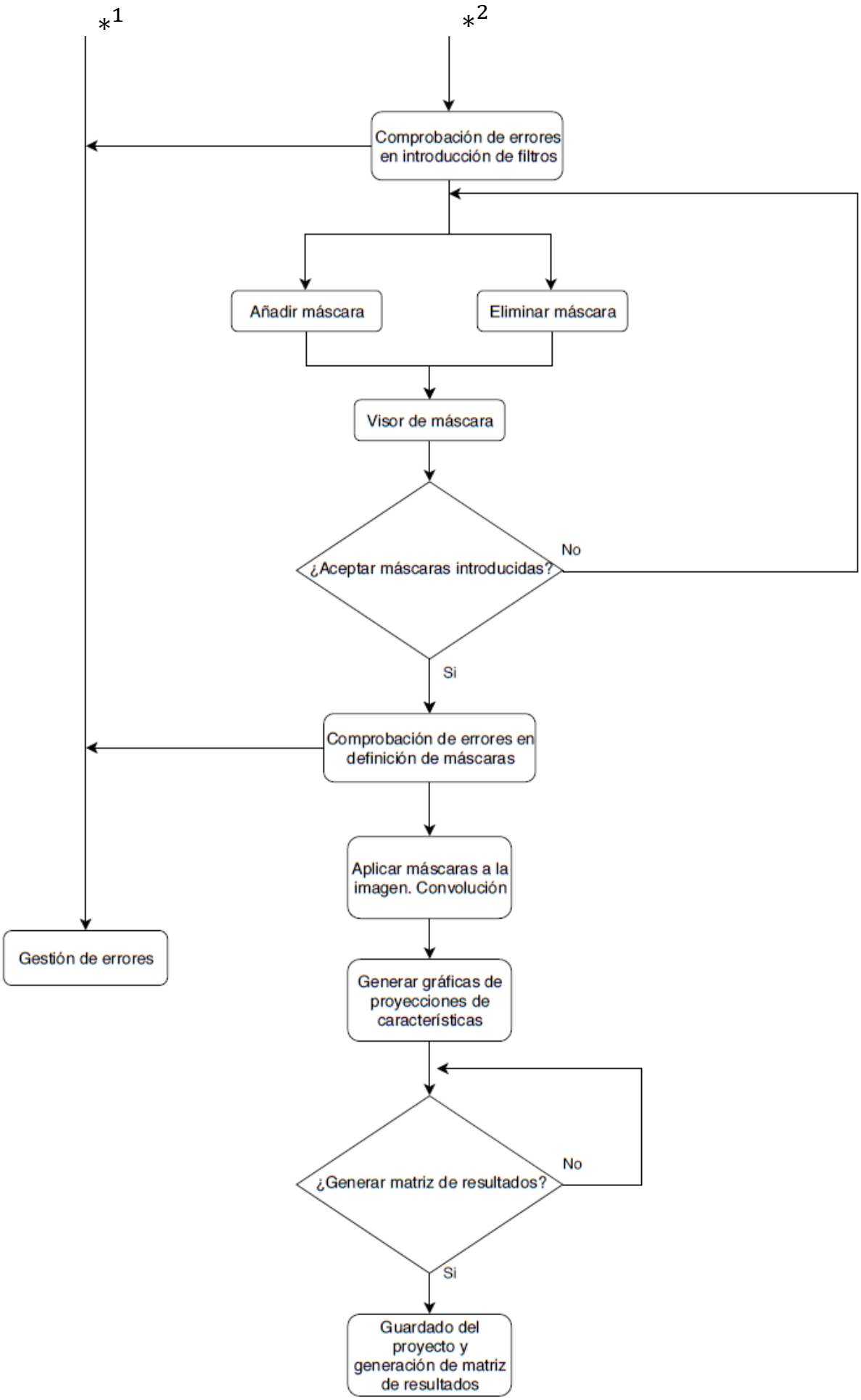


Figura 17

∗ (*Continuación en siguiente* hoja)


*1
*2
Comprobación de errores en introducción de filtros
Añadir máscara
Eliminar máscara
Visor de máscara
¿Aceptar máscaras introducidas?
No
Si
Comprobación de errores en definición de máscaras
Aplicar máscaras a la imagen. Convolución
Gestión de errores
Generar gráficas de proyecciones de características
¿Generar matriz de resultados?
No
Si
Guardado del proyecto y generación de matriz de resultados

Como pasos principales se presentan:

4.4.1 Introducción de la imagen a tratar

Como medida inicial se pide al usuario la introducción de la imagen desde un directorio al programa. Esta imagen será tratada inicialmente para mejorar su paso a través de los sucesivos cálculos de la programación. La imagen se transforma en una imagen en blanco y negro para poder ser tratada posteriormente

Hecho esto la imagen es guardada en la variable principal *Ima* estando así lista para continuar el proceso de análisis por el método Laws.

4.4.2 Definición de los filtros

La siguiente pestaña está enfocada al segundo paso en el proceso del método Laws, la introducción y definición de los filtros con los que la imagen será tratada. El usuario podrá introducir los filtros de manera automática o manual. Una vez introducidos estos serán guardados dentro de la variable principal *Ima*. La correcta definición y guardado de estos filtros es clave, ya que estos filtros servirán para la definición de las máscaras que se emplearán para el tratamiento de la imagen.

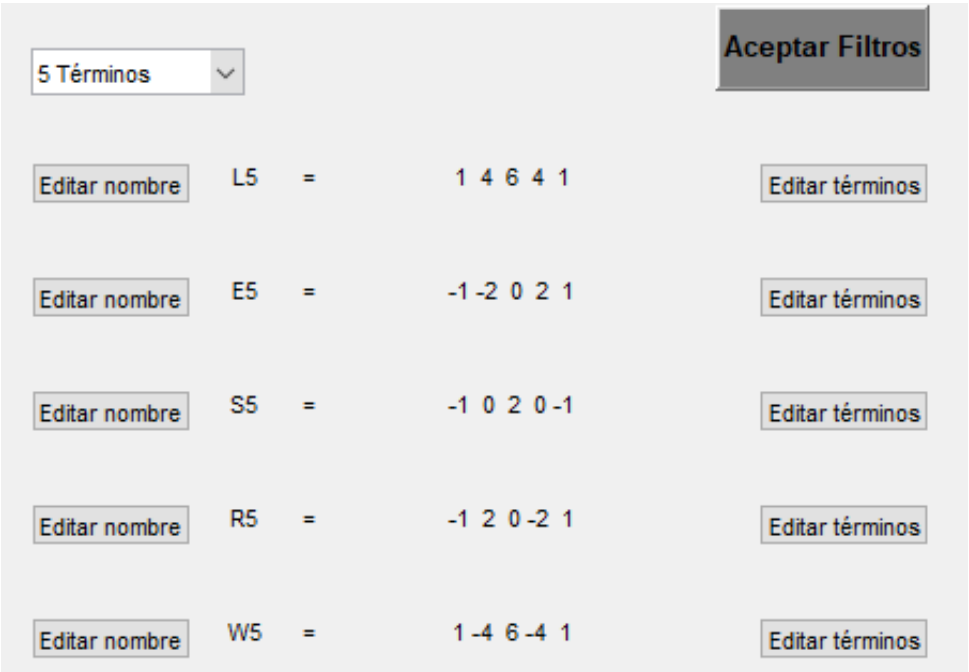


Figura 19

4.4.3 Definición de las mascaras

Una vez definidos y guardados los filtros el siguiente paso es definir las máscaras con las que será tratada la imagen.

Se definen dos tipos distintos de máscaras, las máscaras simples compuestas por la combinación de dos filtros

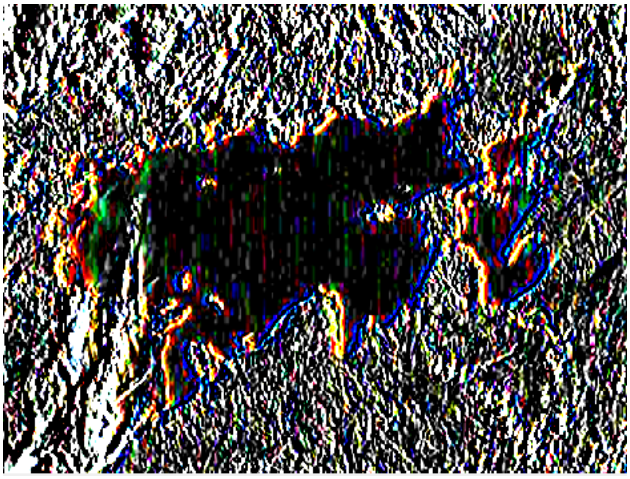
$$Máscara_{Simple} = Filtro_1 \ / \ Filtro_2$$

Y las máscaras dobles compuestas por la combinación de cuatro filtros:

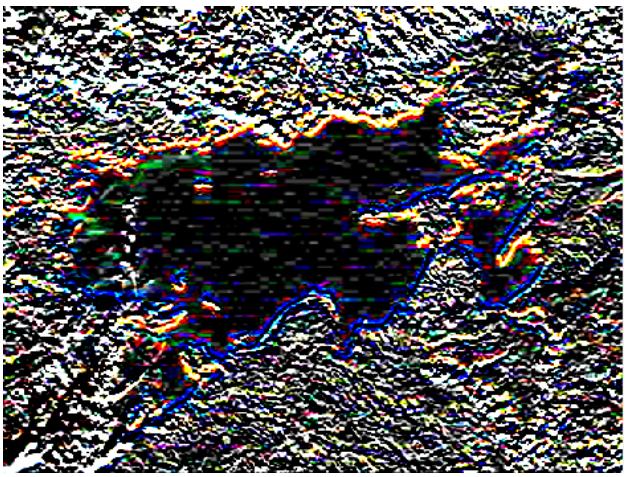
$$Máscara_{Doble} = Filtro_1 * Filtro_2 \ / \ Filtro_3 * Fitro_4$$

Se podrán definir tantas máscaras como el usuario desee, si bien hay que entender que cada máscara está destinada a la localización de una característica concreta de la imagen (texturas con líneas rectas, texturas punteadas, etc.). Cada máscara definida por el usuario será guardada en la variable principal *Ima*. Una vez que el usuario haya definido todas las máscaras que desee el programa convolucionará la imagen con cada máscara dando como resultado una nueva imagen.

En este punto y si se han empleado las máscaras adecuadas es posible la identificación de ciertas zonas con texturas similares en la imagen. Si bien cada máscara convolucionada con la imagen es capaz de detectar una característica concreta, resulta insuficiente para la caracterización completa y exhaustiva de la imagen.



Máscara L5E5



Máscara E5L5

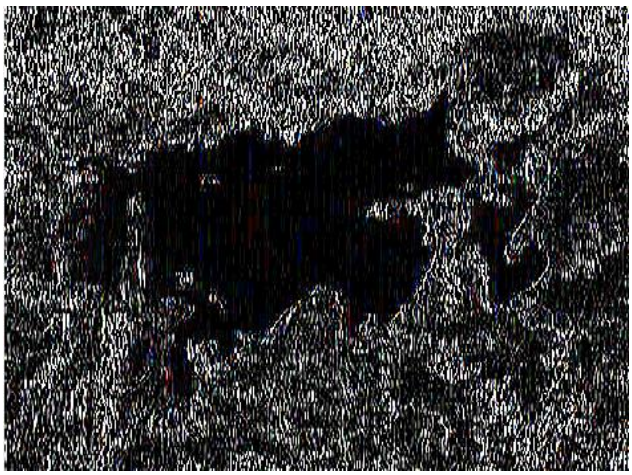


Figura 20

4.4.4 Muestra de resultados

La finalización natural del proceso completo del método de Laws es la muestra de los resultados de la imagen convolucionada con todos los posibles pares de máscaras definidas. De todo el conjunto de máscaras introducidas se realizan todas las combinaciones posibles para la convolución de la imagen con cada máscara. Así de definir 4 máscaras existen 6 posibilidades de combinación distintas de máscaras sobre la imagen.

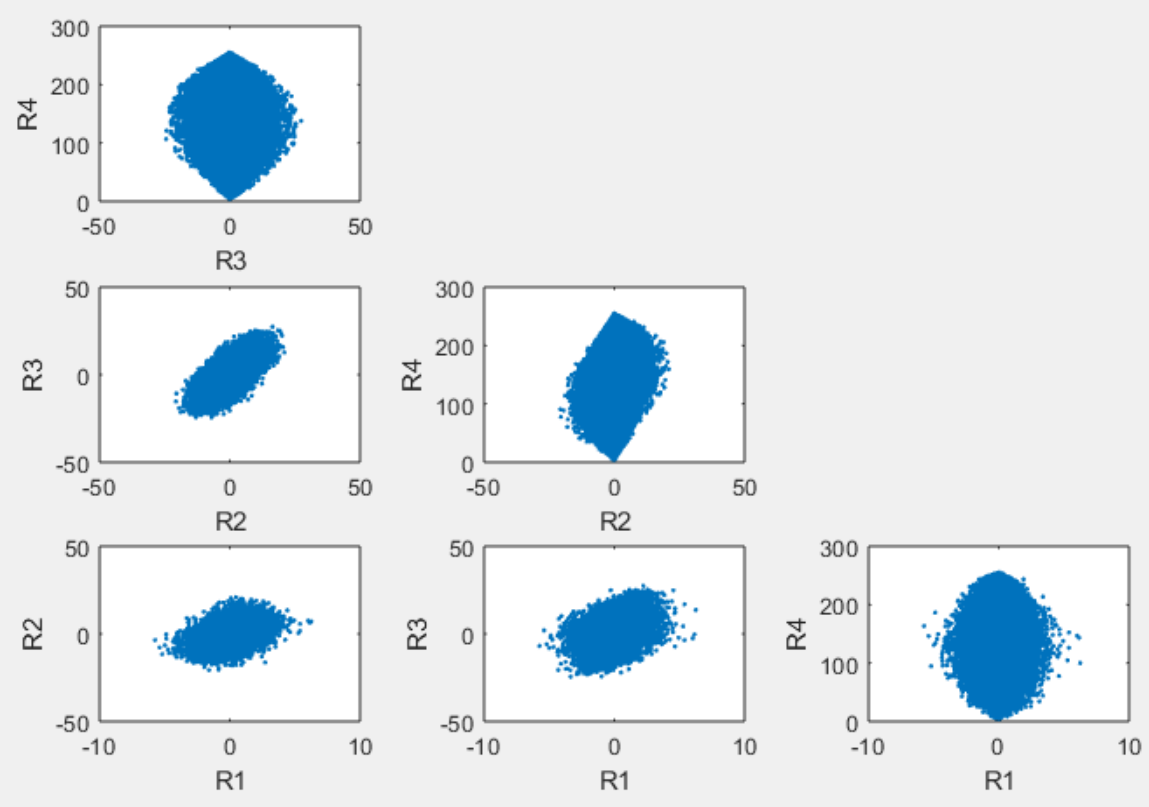


Figura 21

Se convoluciona la imagen con todas las máscaras definidas y los resultados se plasman en gráficas bidimensionales donde cada eje es un resultado distinto.

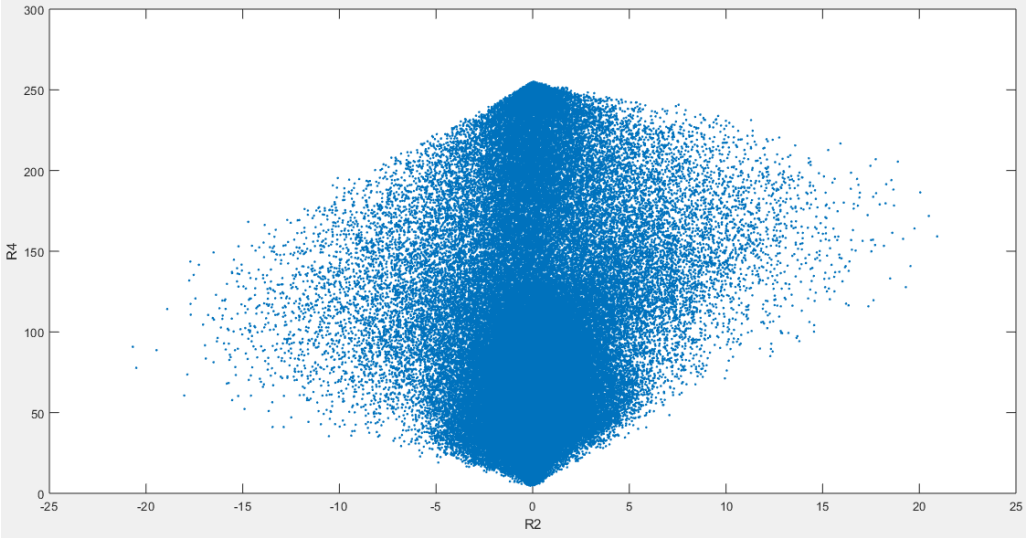


Figura 22

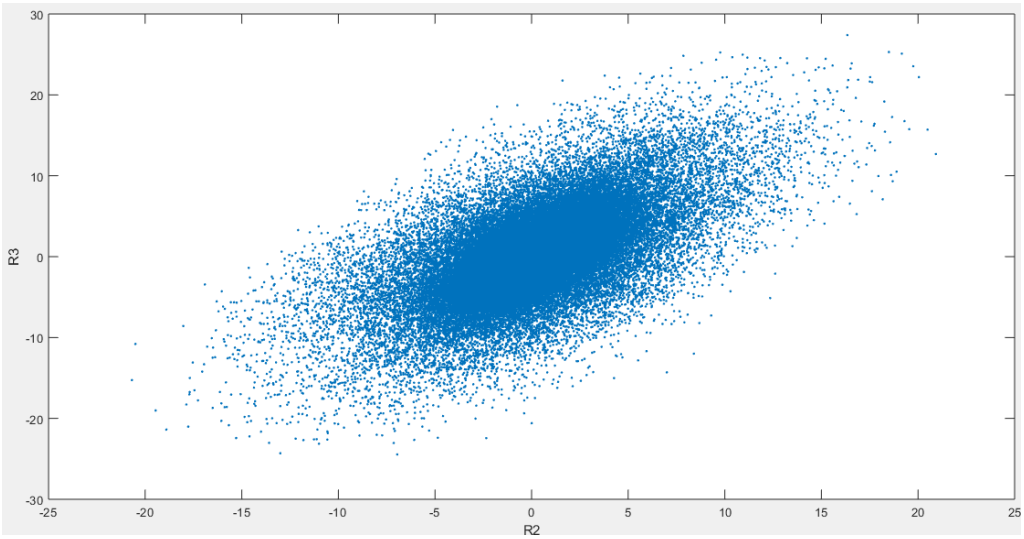


Figura 23

Como resultado final se obtiene una matriz la cual sirve como punto de partida para los clasificadores en los futuros pasos del análisis de la imagen.

4.5 Limitaciones de programación

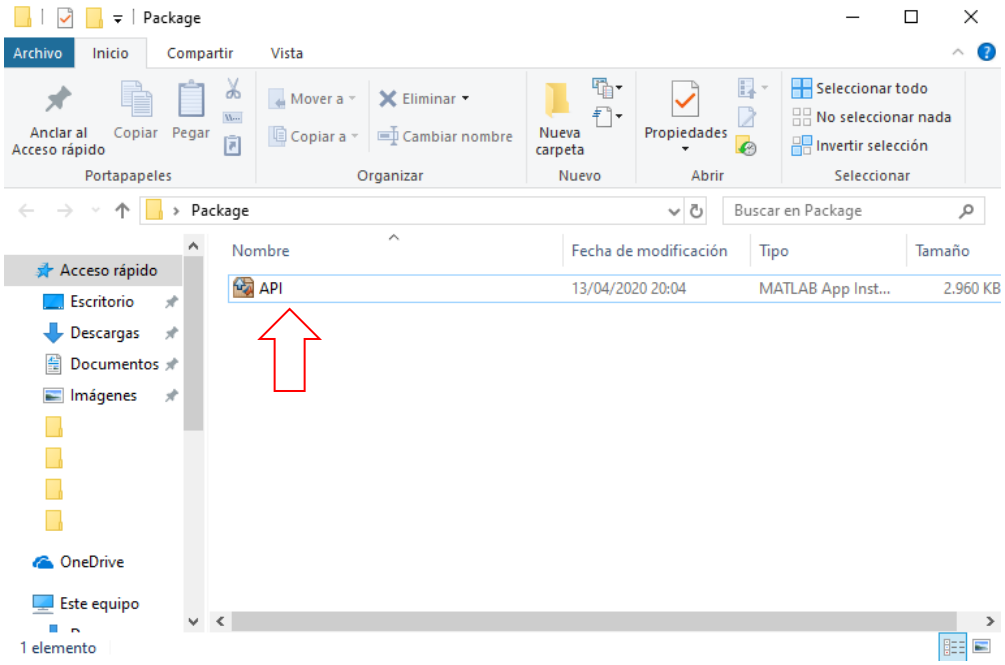
Como se ha desarrollado en puntos anteriores, el programa *ATI* esta escrito íntegramente en el programa Matlab. Esto puede resultar en limitaciones propias del programa o de la metodología seguida. A continuación, se expondrán ciertas limitaciones, algunas corregidas mediante soluciones de compromiso y otras que serán posibles puntos de mejora para un futuro.

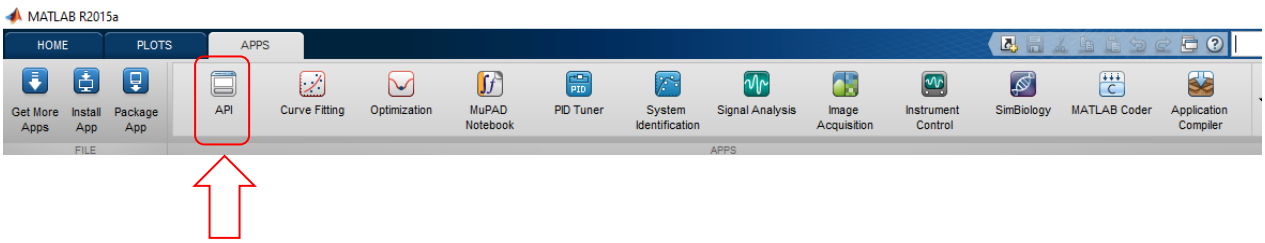
- Limitación propia de Matlab: Todos los códigos y archivos desarrollados para el programa *ATI* están elaborados en el programa Matlab. Esto además de presentar ciertas ventajas puede incluir algunas limitaciones. Por su constitución, el programa *ATI* requiere ser reproducido en un ordenador el cual incluya una versión de Matlab 2015a o superior. Matlab es un programa comercia por lo que requiere de una suscripción propio por parte del usuario limitando el uso del programa *ATI*.

  Para el uso de un usuario nuevo de este programa requeriría primero la suscripción al sistema comercial de Matlab Word y posteriormente la descarga e instalación del programa completo Matlab. Esto significa una gran inversión de tiempo y recursos para la utilización única de este programa. Este punto plantea numerosas desventajas frente a otros posibles programas que puedan ejecutarse directamente en cualquier ordenador (disponga o no de Matlab) y que no requieran de ninguna inversión económica.

  Es por ello que se plantea la búsqueda de algún método de compilación y empaquetamiento del programa creado para poder ser ejecutado en cualquier ordenador disponga o no de Matlab. La solución encontrada se basa en una herramienta propia que incluye Matlab llamada MCR (MATLAB Compiler Runtime). Esta herramienta está pensada para la elaboración integral de programas de forma que se puedan ejecutar de forma independiente a Matlab.

La forma de utilización de esta herramienta es la siguiente: una vez elaborado todos los códigos necesarios, archivos y carpetas que conforman el programa definitivo se procede a su empaquetamiento en un solo programa el cual se convertirá en el instalador. Este programa englobará todos los archivos y carpetas de nuestro programa además de incorporar un archivo especial llamado MCR (MATLAB Compiler Runtime). Este archivo se trata de un instalador el cual incluye las funciones básicas de Matlab. De esta forma, en un único instalador dispondremos de todos los archivos de nuestro programa, así como un pequeño instalador de las funciones básicas de Matlab requeridas para poder ejecutar el programa en cualquier ordenador, tenga o no instalado Matlab.





De esta forma un usuario final podría hacerse acopio de este archivo instalador. Una vez en su ordenador el usuario instalaría primero el programa MCR (MATLAB Compiler Runtime) para poder disponer de las funciones básicas de Matlab para poder ejecutar programas externos como *ATI* sin

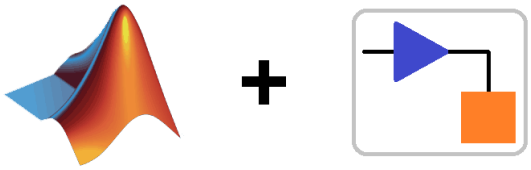
tener que instalar el programa Matlab completo. Hecho esto ya podría instalar el programa *ATI* y disponer de su uso.

Esta instalación permite el uso de todas las funciones presentes en el programa *ATI* menos una. La creación de informes es una herramienta que no puede operar solo con la instalación de MCR (MATLAB Compiler Runtime) por lo que esta función queda deshabilitada en esta versión del programa *ATI*, requiriendo la instalación de Matlab completo si se desea la elaboración de informes pdf de las imágenes analizadas.

- Limitación de tratamiento de imágenes: Como tal se ha indicado en puntos anteriores que Matlab es un programa especializado en el uso de matrices. Esto plantea muchas ventajas a la hora de implantar un método como el de Laws en un programa así. Al tratarse de un método en el cual se deben realizar numerosas operaciones con matrices (convoluciones, etc.) eso resulta una ventaja computacional. Otros programas no están tan especializados y requieren en un mayor gasto de recursos.

A pesar de ello Matlab no es un programa especializado en tratamiento de imágenes. Merece estudio comprobar si existiera un programa capaz de añadir más opciones de edición de imágenes que resultarán útiles para este problema. En este caso Matlab requiere de mucha programación previa para poder añadir opciones de edición de imágenes como serían cambiar escala, poner en escala de grises, realizar cambios en saturación contraste, etc. Es por ello que a priori este es una limitación y un gran punto de mejora para el futuro.

- Limitación de programa en Guide: Se ha empleado la herramienta Guie de Matlab para la creación de la interfaz de Matlab. Esto se ha elegido así por ser la mejor herramienta que posee Matlab para este propósito. A pesar de disponer de ventajas también podemos hablar de ciertas limitaciones propias que podrían afectar al desarrollo y uso del programa. Temas como el aspecto visual, utilización o distribución de la información dentro de la interfaz plantean la pregunta de si existen mejores formas de realizar este trabajo.



MATLAB
SIMULINK®

Existe la herramienta Simulink también incluida en Matlab la cual puede presentar un gran abanico de posibilidades para la programación de distintos métodos. Merece estudio analizar las mejoras que se pueden aportar mediante el uso de Simulink a este problema frente al uso de una interfaz como la que se ha optado.

5 CONCLUSIONES Y PUNTOS DE MEJORA

Como conclusiones se puede afirmar que el objetivo de la creación de una interfaz funcional para el análisis de imágenes mediante el método de Laws se ha cumplido con éxito. Si bien, usando los medios disponibles en Matlab y empleando para ello la teoría estudiada en el método de Laws se ha desarrollado una interfaz capaz de procesar las imágenes introducidas. La propuesta de distribución en ventanas adyacentes hace que el programa sea más intuitivo y fácil de usar a la hora de moverse por él.

La elaboración de pruebas mediante el clasificador introducido (mínimas distancias) ha reflejado resultados muy aproximados a la realidad esperada. Se ha podido comprobar cómo afectan la utilización de distintos filtros y máscaras en el análisis de la imagen y cómo afecta a la identificación y clasificación de distintos objetos existentes en la imagen.

Son muchos los aspectos potenciales a mejorar del programa creado. Respetando la metodología implantada y el proceso seguido, se puede hablar de ciertos puntos de mejora que elevarían las funcionalidades del programa:

- **Implantación de otras metodologías de análisis**

  Si bien la única metodología que ha sido introducida es de Laws, en un futuro este programa podría incluir otras. Se podrían implantar distintas metodologías las cuales el usuario podría elegir previamente para el estudio de su imagen. Estas nuevas metodologías introducidas buscarían otros descriptores como descriptores obtenidos por métodos estadísticos, descriptores LBP, etc. Cada descriptor distinto podría ser usado según convenga en cada imagen a elección del usuario. Éste a su vez podría comprobar la influencia de la utilización de un descriptor u otro para una imagen en concreto. De esta forma, problemas existentes provenientes de la metodología de Laws (máscaras ideales, etc.) podrían ser solventados con el empleo de un método distinto.

- **Implementación de más tipología de clasificadores al programa**

  Como aspecto adicional se añadió un pequeño clasificador sencillo al programa *ATI* basado en la mínima distancia euclidea. Este clasificador presenta ventajas como son su pequeño gasto computacional o su sencillez de programación. Sin embargo, puede llegar a ser insuficiente a la hora de clasificar ciertas propiedades. Es por ello que se plantea como punto de mejora añadir más

clasificadores. Los clasificadores nuevos podrán ser de distinta tipología (clasificadores basados en redes neuronales, clasificadores lineales, etc.). El usuario podría elegir el tipo de clasificador a utilizar en cada caso, así como sus parámetros necesarios para su uso. A su vez, como ya se comentaba en el punto anterior, se podría realizar un estudio para ver cómo afecta el empleo de un clasificador u otro en el análisis de una imagen en concreto.

- **Implantación de procesado de varias imágenes**

El programa *ATI* desarrollado solo puede procesar una imagen dada. A pesar de las ayudas auxiliares introducidas, como son los procesos de guardado de filtros, máscaras, etc. se puede volver un poco tedioso la manera de procesar varias imágenes. Como punto de mejora se plantea la posibilidad de aumentar el número de imágenes a procesar de manera seguida. De esta forma el usuario podría introducir un conjunto de imágenes a analizar para que el programa las analizara con la metodología precisa y de forma conjunta. Se podría también clasificar las imágenes introducidas por tipología agrupándolas en grupos con propiedades similares.

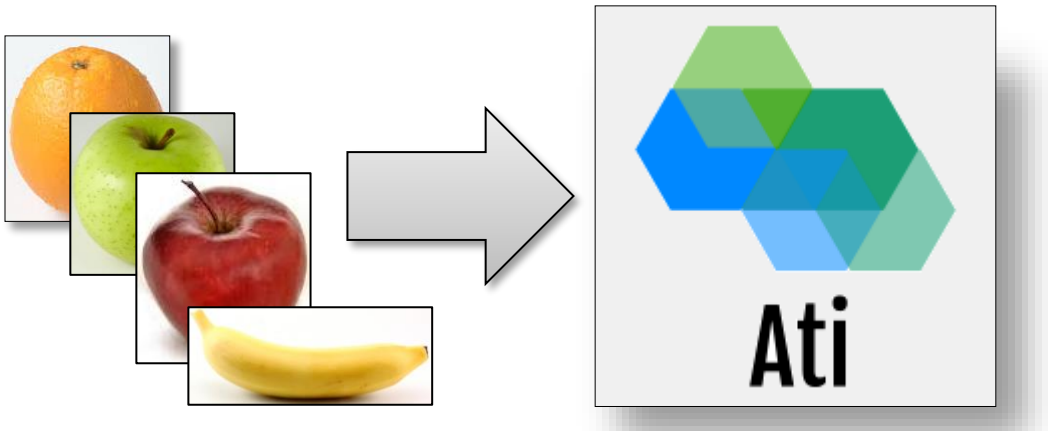


Figura 24

- **Implantación en otros dispositivos**

El programa desarrollado presenta limitaciones en cuanto a los dispositivos en los que debe de localizarse. Al tratarse de un programa desarrollado en Matlab, necesita de un dispositivo tipo Pc con un sistema operativo compatible como por ejemplo Windows. Esto limita su abanico de utilización ya que es necesario guardar las imágenes a analizar y cargarlas en el programa. Esto orienta la utilización de este método a posteriori, en procesos de análisis de históricos etc. Como posible mejora se podría plantear la adaptación del código desarrollado a nuevas plataformas que permitan su utilización en otros dispositivos. Un ejemplo de ello sería a codificación del programa *ATI* en lenguaje Python o Android para poder ser cargado en dispositivos móviles con sistema operativo Android. Esto podría añadir problemas por ser Matlab un lenguaje muy adaptado al uso de matrices, pero también podría traer ventajas. Entre estas ventajas se puede plantear la utilización más dinámica de la aplicación, pudiendo realizar fotografías con el dispositivo y posteriormente ser cargadas directamente al programa.

Figura 25

A su vez se podría pensar en la adaptación del programa en su uso con videos. Podría emplearse en la identificación de objetos a tiempo real en videos tomados con dispositivos móviles.

6 ANEXOS

6.1 Anexo A: Guía de usuario

A continuación se adjunta una guía de usuario para la correcta utilización del programa creado.

MANUAL DE USUARIO PROGRAMA ATI

Guía y uso

Descripción breve

Manual de usuario para la aplicación y uso del programa Ati

jesus medran castro
ati-asistencia@gmail.com

Índice

1 - USO DE LA INTERFAZ

Bienvenido al manual de usuario del programa *Ati*. En este pequeño documento le enseñaremos como utilizar fácilmente el programa y cómo interpretar sus resultados. Se expone a modo de ejemplo el análisis sencillo de una imagen.

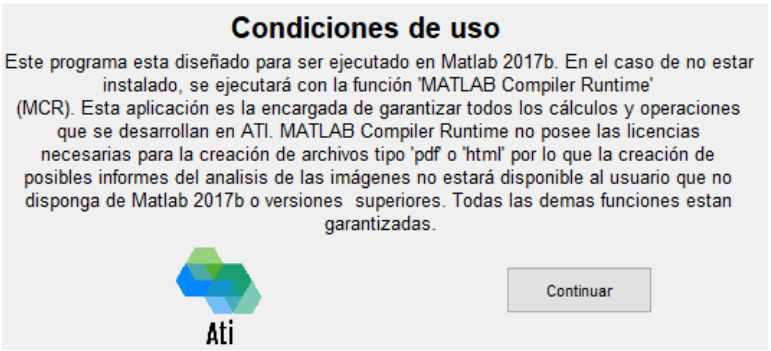
Se indicarán una serie de pasos genéricos para poder analizar cualquier imagen. Cabe decir que el uso e interpretación de los datos de este programa está enfocado para personas con un nivel básico en el análisis de imágenes por medios digitales..

1.1 Instalación

Para facilitar la instalación de *Ati* al usuario se ha adjuntado el documento "Procedimiento de instalción.txt" donde se dice paso a paso cómo se debe instalar este programa.

Ati está diseñado para ejecutarse en Matlab 2017b o versiones superiores como una script de código. Es por ello que se incluye el archivo "ATI_Matlab" el cual podrá ejecutarse directamente en Matlab.

No obstante *Ati* también incluye una versión ejecutable para usuarios que no posean Matlab. Para ello será necesario instalar previamente el programa MCR (MATLAB Compiler Runtime) el cual hará posible el cálculo y las operaciones que se realizan dentro de *Ati* sin disponer de Matlab. Esta versión, sin embargo, no posee licencias de creación de archivos ".pdf" o ".html" por lo que quedan suspendida la posibilidad de crear un informe de la imagen analizada. Las demás funcionalidades se mantienen.

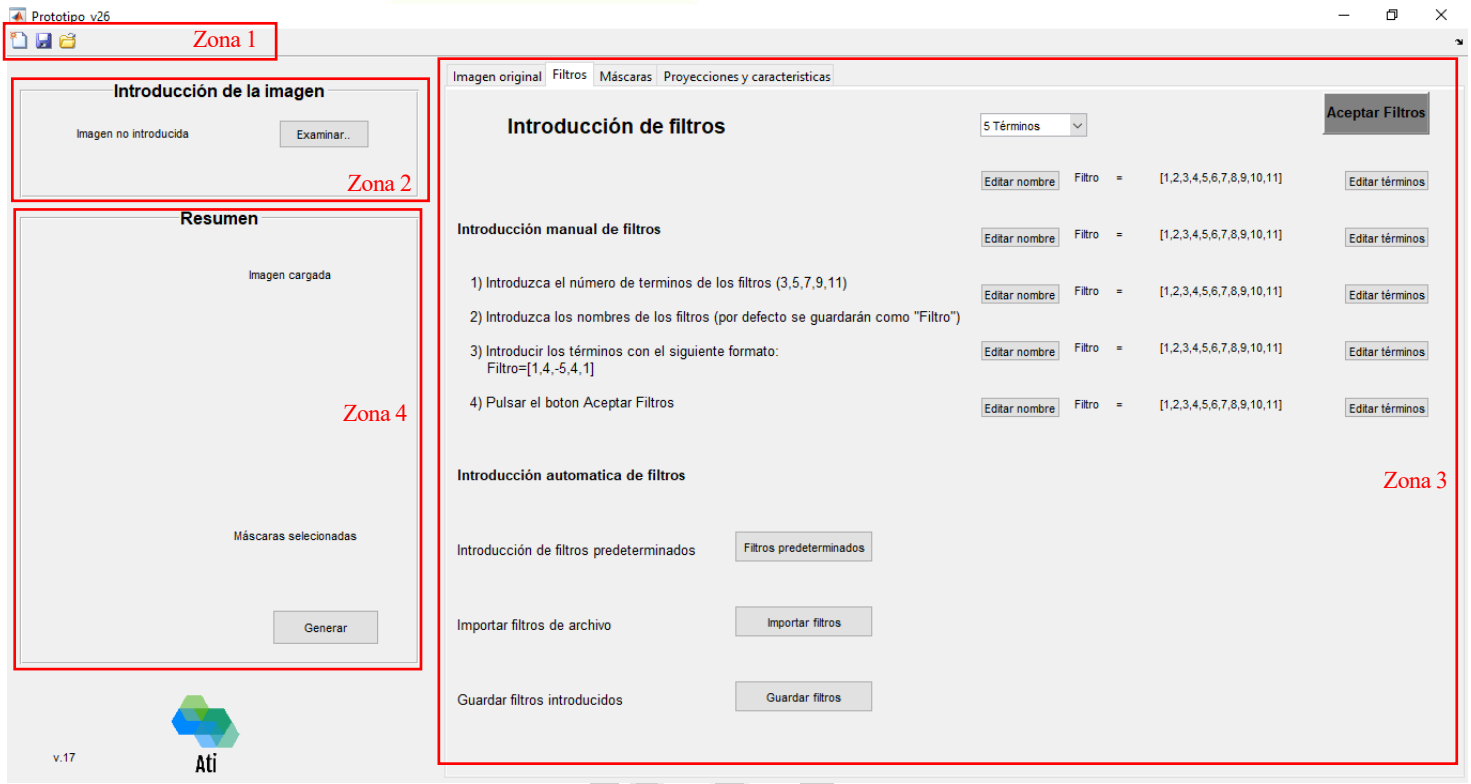


1.1.1 Nuevo proyecto

Se recomienda que para la realización de un nuevo análisis de una imagen se pulse previamente el botón "Nuevo" de esta forma el programa se reseteara y dejara todos los cuadros vacíos a la espera de la introducción de nuevos datos.

1.1.2 Partes de la interfaz

La interfaz puede dividirse en 4 grandes partes diferenciadas:



- **Zona 1:** Configuración y ayuda global del programa

En esta zona se alojarán las herramientas principales de la configuración del programa. Estas herramientas podrán usarse para refrescar el programa y generar un nuevo proyecto, guardar el proyecto actual y ser cargado posteriormente y obtener una guía necesaria que sirva como referencia de la utilización del programa por parte del usuario.

- **Zona 2:** Introducción de la imagen a analizar

En esta zona se introducirá la imagen que se desea analizar. Por defecto el programa autodirige al usuario a la carpeta de imágenes de ejemplo, pero este podrá cargar cualquier imagen que se encuentra alojada en su PC. El programa puede trabajar sin problema con imágenes con extensión ".png" o ".jpg", no siendo válidas extensiones distintas.

- **Zona 3**: Introducción de parámetros y visualización de resultado

Es la zona principal del programa. En ella el usuario podrá introducir o cambiar todos los datos necesarios para un correcto análisis de la imagen tales como filtros, máscaras, etc. Al tratarse de una metodología progresiva y contener varios parámetros se ha optado por la división del proceso del método en distintas pestañas. Cada pestaña se centrará en un punto principal del método de Laws presentando facilitando al usuario el uso de la interfaz. Las pestañas que se han considerado son:

- Pestaña 1: Imagen original

  Una vez cargada la imagen a analizar correctamente se podrá ver en esta pestaña teniéndola así de referencia para la correcta elección de los filtros y máscaras en los futuros pasos.

- Pestaña 2: Filtros

  Esta pestaña está destinada a la introducción de los filtros. El usuario posee dos opciones, introducción de los filtros de manera manual (definiendo término a término) e introducción de los filtros de manera automática. Se le presenta al usuario una breve descripción con ejemplos de cómo se introducirían los filtros de manera manual.

  El usuario podrá elegir en todo momento la longitud de los filtros a introducir siendo las posibilidades de 3, 5, 7, 9 u 11 términos en cada filtro. El nombre de cada filtro también podrá ser elegido por cada usuario para poder servirle de referencia en pasos futuros. Una vez introducidos los filtros el usuario podrá guardar esos filtros para poder ser usados en diferentes imágenes en otras ocasiones simplificando así la introducción de términos.

  Otra forma de introducción de filtros es de forma automática. Se le presenta al usuario unos filtros de Laws predeterminados de 3, 5, 7 y 9 términos. A su vez también podrá cargar filtros que haya guardado anteriormente para poder ser usados ahora. Esta metodología simplifica la introducción de cada termino de los filtros.

  Por último, una vez introducido o hechas las modificaciones necesarias a los filtros el usuario deberá pulsar el botón "Aceptar filtros" para que estos sean cargados correctamente en el programa permitiendo así continuar con los siguientes pasos.

- Pestaña 3: Máscaras

  Una vez introducidos de forma correcta todos los filtros el usuario deberá definir las máscaras utilizar. Como ya se definió anteriormente se presenta al usuario la posibilidad de definir máscaras simples o dobles. Para ello se deberá pulsar en "Añadir máscara". En este punto y una vez elegida la opción de máscara simple o doble, el usuario podrá definir la combinación más conveniente de los filtros anteriormente definidos. Cada filtro esta guardado con el nombre que el usuario le asignó.

  Una vez definida la máscara el usuario podrá visualizar la aplicación directa de dicha

máscara a la imagen a analizar. Esto ayuda a entender mejor la utilidad de cada máscara definida.

A su vez, el usuario también podrá eliminar cualquier máscara definida previamente con el botón “Eliminar máscara”

Una vez todas las máscaras estén definidas, el usuario deberá pulsar el botón “Aceptar máscaras” para que estas sean cargadas correctamente en el programa permitiendo así continuar con los siguientes pasos.

- Pestaña 4: Proyecciones y características

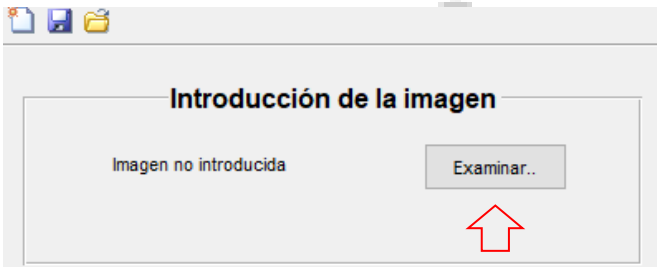
  Estando las máscaras correctamente introducidas y el cálculo de convolución de cada máscara con la imagen realizado se podrá acceder a esta última pestaña. En ella se mostrarán, de forma piramidal los gráficos resultados correspondientes a todas las posibles confinaciones de cada máscara con la imagen por parejas. De este modo se obtendrán $\frac{(n-1)\cdot n}{2}$ gráficas de resultados por cada n máscaras que se apliquen a la imagen.

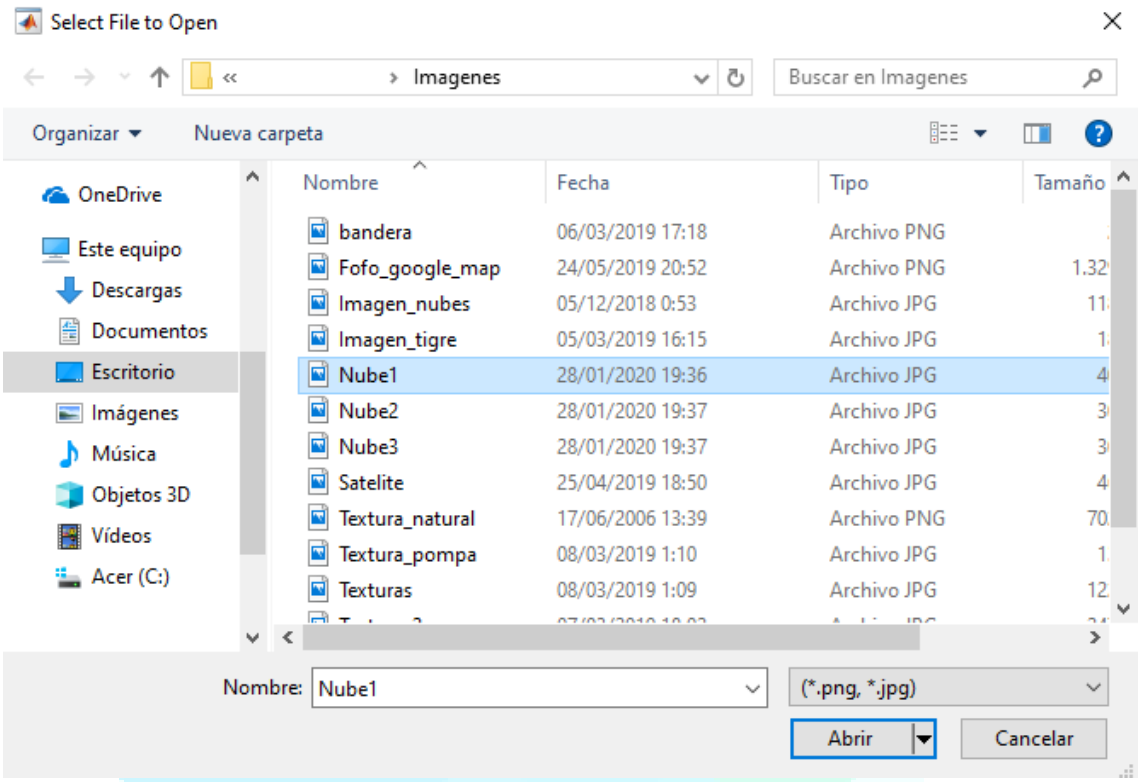
- **Zona 4:** Control global del proceso

En la utilización del programa se llevan a cabo comprobaciones correspondientes a la correcta introducción de los datos por parte del usuario en la Zona 3. Es en esta Zona 4 donde se mostrará un resumen de los datos introducidos (imagen cargada, filtros a emplear, máscaras elegidas) en todo momento para le sea más fácil el desplazamiento a través de las pestañas por parte del usuario.

1.1.3 Cargado de imagen

Dentro de la zona X de la interfaz se encuentra la opción "Examinar". Pulsando esta opción el usuario podrá seleccionar una imagen para ser analizada.

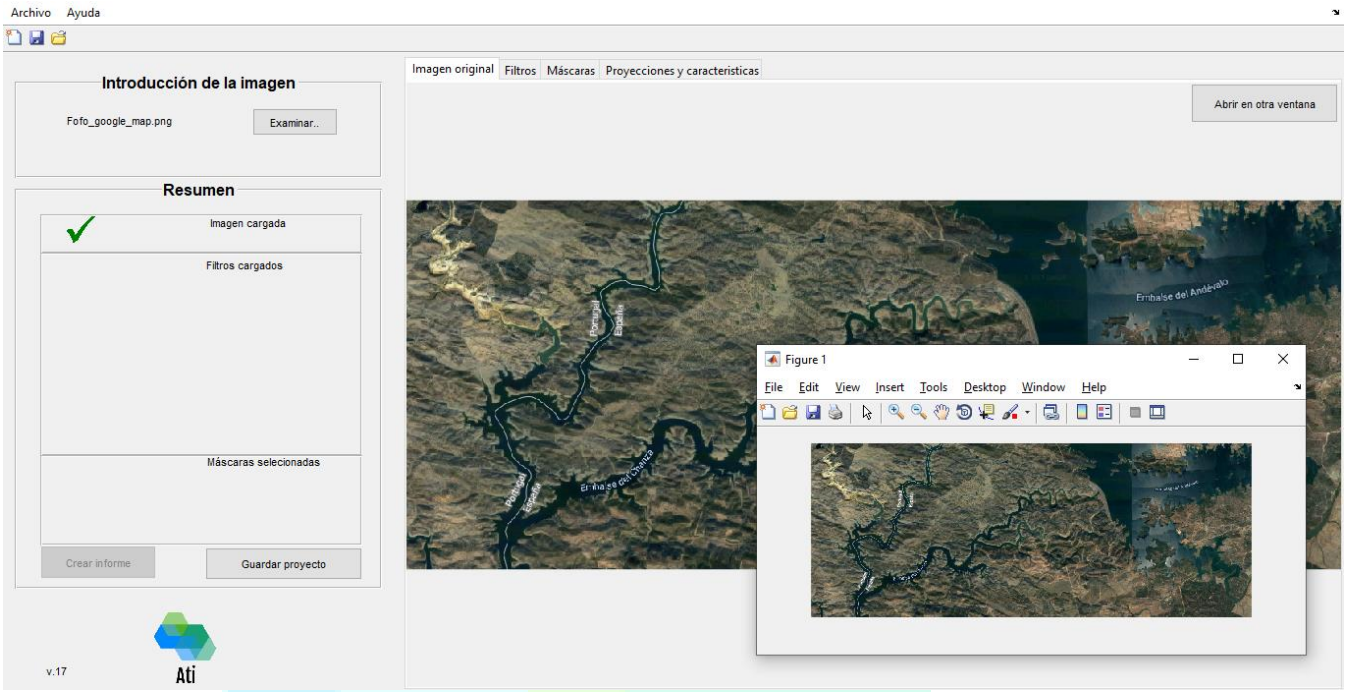




Por defecto se abrirá la carpeta de imágenes del programa Ati, sin embargo el usuario podrá cargar cualquier imagen que se encuentre en su ordenador con las extensiones ".png" y ".jpg". El uso de cualquier otra extensión no garantiza el correcto funcionamiento del programa.

Una vez seleccionada la imagen esta se mostrará en la pestaña "Filtros".

Siempre que se desee se podrá abrir, de manera adicional, la imagen cargada en otra venta complementaria. Para ello el usuario deberá pulsar la opción *Abrir en otra ventana*.



1.1.4 Definición de filtros

El siguiente paso natural en el uso del programa es la definición de los filtros. Toda esta parte se desarrollará en la pestaña "Filtros". El usuario contará con tres formas distintas para poder introducir los filtros. Estas tres formas son:

· Introducción de filtros de forma manual

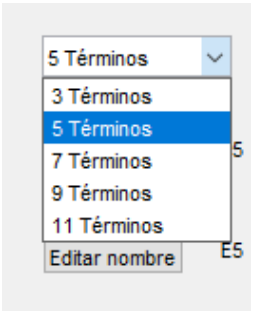
· Introducción de filtros de Laws predeterminados

· Cargar filtros guardados

Si bien el usuario determinará qué opción se adapta mejor para el proceso de análisis de la imagen introducida. A continuación se expone la utilización de cada modelo de introducción de filtros:

- **<u>Introducción de filtros de forma manual</u>**

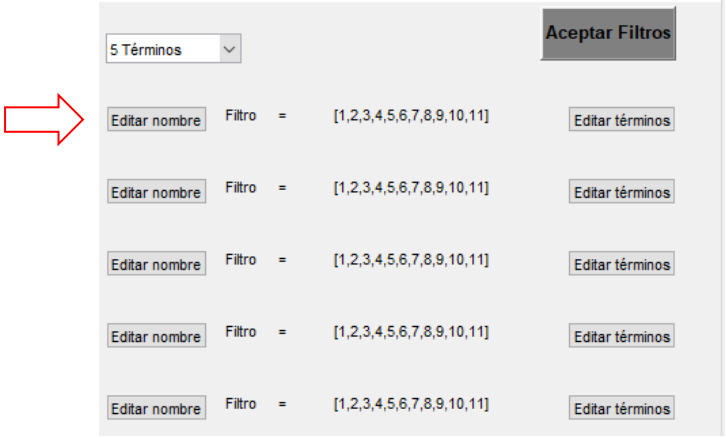
  Para introducir de forma manual los filtros el usuario primero deberá determinar el número de filtros que quiere introducir. Para ello se podrá elegir una opción de las presentes en el botón desplegable "términos". Las opciones que se muestran son para filtros de tamaño 3, 5, 7, 9 y 11.

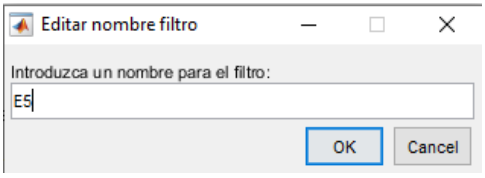


Una vez elegida una opción se mostrarán en esta misma pestaña el número de filtros elegidos

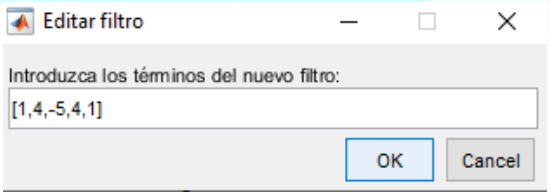
En el siguiente paso se deberá introducir filtro a filtro su nombre y los términos que lo componen. Para ello por cada filtro se puede observar dos botones: “Editar nombre” y "Editar términos"

El botón "Editar nombre" permitirá al usuario asignar un nombre para el filtro seleccionado. Este paso es importante ya que este nombre asignado será el que se use para futuros pasos como la definición de las máscaras.



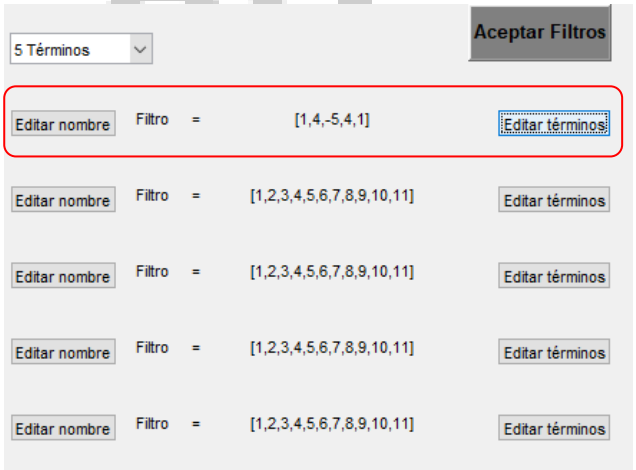


Como siguiente paso se deberá introducir los términos que componen cada filtro. Para ello se usará el botón "Editar términos". Hecho esto se mostrará al usuario una ventana para introducir todos términos que componen el filtro.



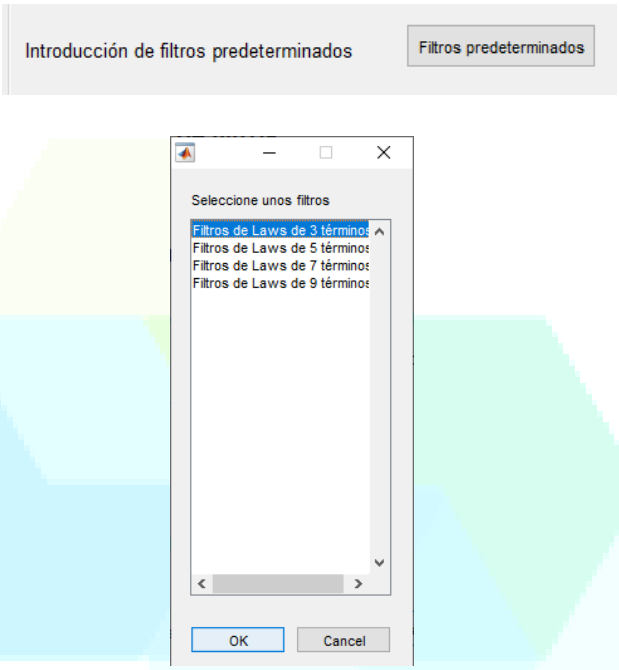
Para la introducción se debe introducir los términos englobándolos entre corchetes "[ ]" y separándolos por comas ",". A continuación se muestra un ejemplo de introducción de filtro:

Filtro= [1,4,-5,4,1]



- **Introducción de filtros de Laws predeterminados**

Los filtros de Laws son usualmente empleados para el análisis de imágenes. Por ello se presenta esta opción. El usuario podrá pulsar la opción "Filtros de Laws predetermiados" para poder ver las 5 opciones de filtros de Laws que están definidas en el programa.



En esta ventana se deberá seleccionar una de las 5 opciones que se plantean:

· Filtros de Laws predeterminados de 3 términos

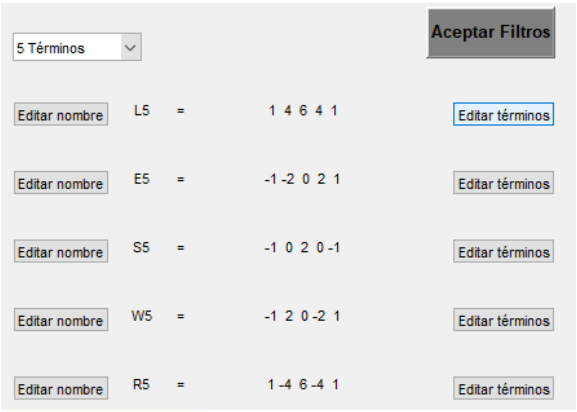
· Filtros de Laws predeterminados de 5 términos

· Filtros de Laws predeterminados de 7 términos

· Filtros de Laws predeterminados de 9 términos

· Filtros de Laws predeterminados de 11 términos

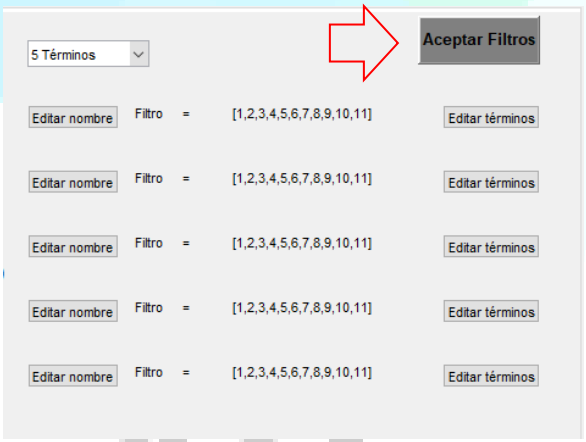
Seleccionado el número de términos que más se adecue al análisis se cargarán los términos de cada filtro en la pestaña "Filtros" como si hubieran sido introducidos de forma manual.



El usuario podrá editar tanto los nombres de los filtros como el valor de sus términos con los botones "Editar nombre" y "editar términos".

- **Cargar filtros guardados**

Como última opción se plantea la introducción de los filtros de manera automática cargando los filtros ya definidos y guardados en proyectos anteriores. Para ello el usuario deberá pulsar la opción "Cargar filtros" dentro de la pestaña "Filtros".



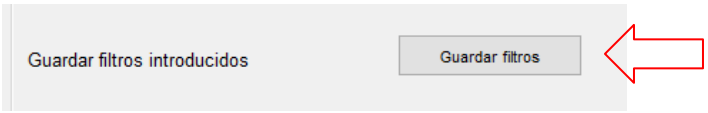
automáticamente se le presentará al usuario una ventana donde se mostrará el contenido de la carpeta "*Filtros guardados*" en la cual el usuario previamente habrá guardado filtros de un proyecto anterior. El usuario podrá cargar cualquier archivo de filtros guardados almacenado en su ordenador siempre que se trate de un archivo ".mat" y con las características específicas requeridas por el programa *Ati*.

Como siguientes pasos tras la introducción de los filtro se presentan dos posibles opciones que serían el guardado de los filtros introducidos o cargar los filtros en el programa de forma definitiva. Ambos pasos no podrán ser empleados hasta que no se introduzcan los filtros por parte del usuario. En las siguientes líneas se explican estas opciones.

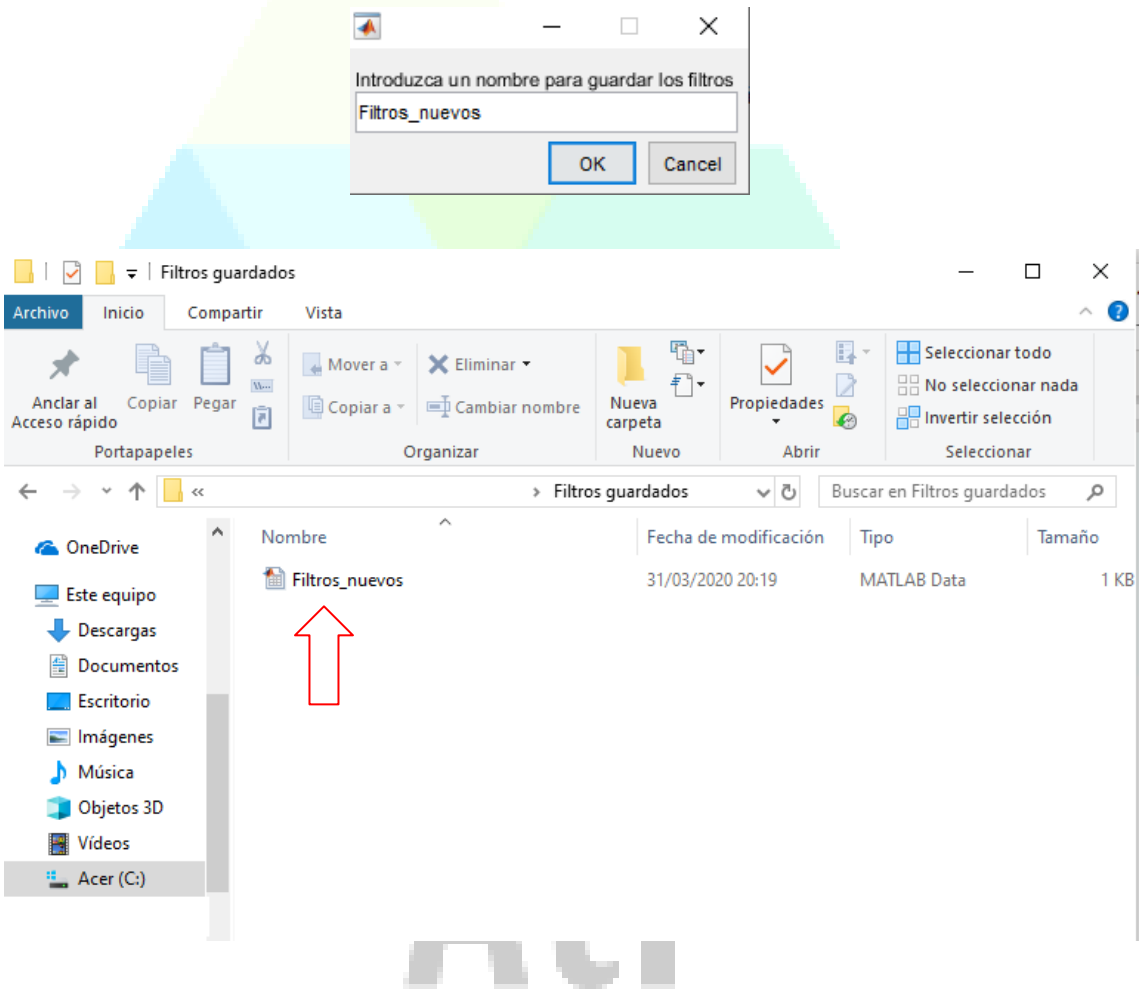
- **Guardado de filtros**

Una vez que el usuario haya introducido los filtros independiente del método utilizado se podrán guardar para poder ser utilizados en futuros proyectos. Para ello se deberá elegir la opción

"Guardar filtros" en la pestaña "Filtros".

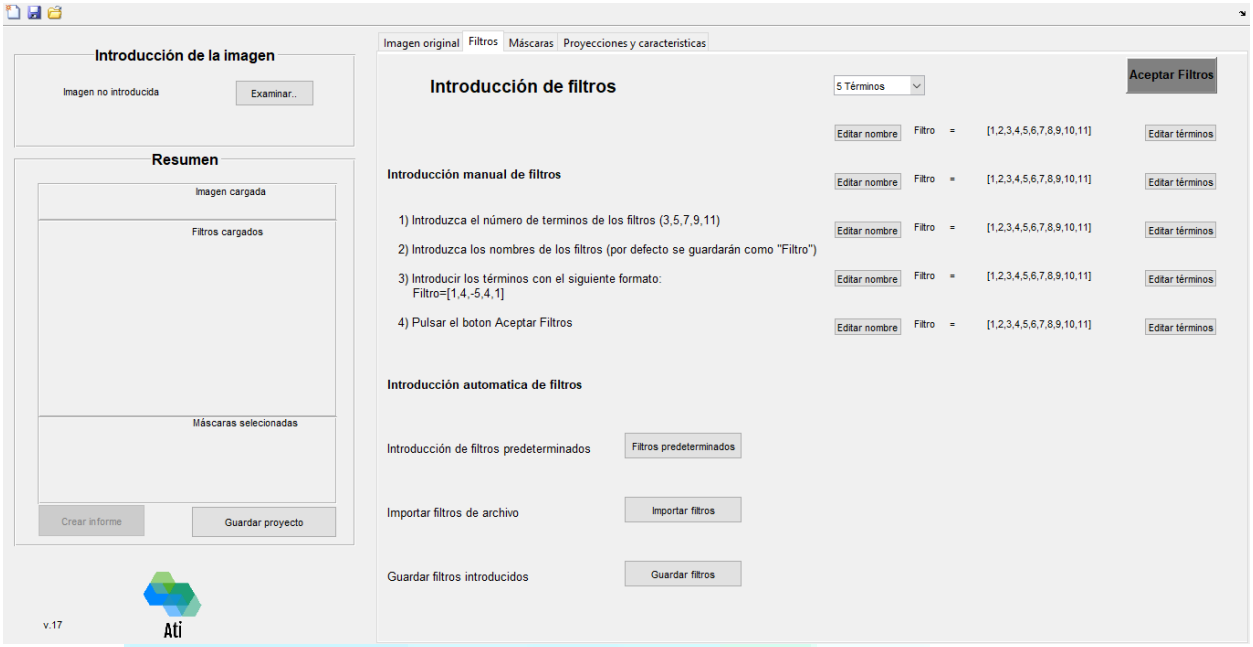


En este punto el usuario deberá introducir un nombre para poder identificar los filtros guardados. El número de filtros, sus nombres y todos sus términos serán guardados en la carpeta "*Filtros guardados*"bajo un archivo ".mat".

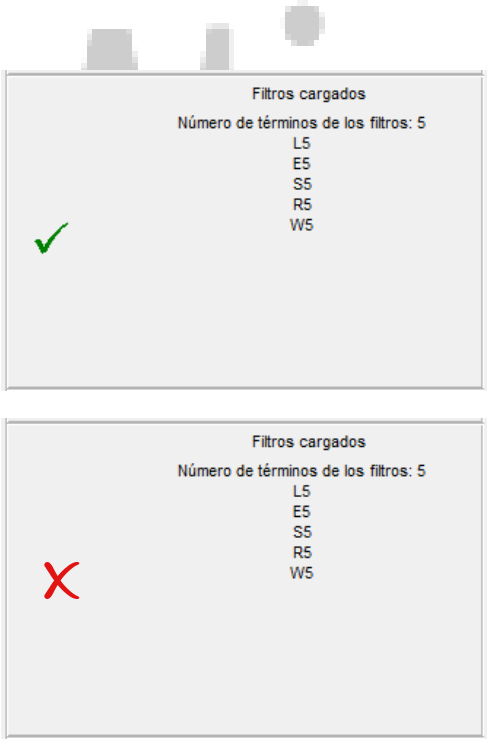


- **Aceptar de filtros**

Una vez introducidos todos los filtros de manera correcta, el usuario deberá aceptar los filtros introducidos guardándolos en el programa (variable Ima). Para ello se elegirá la opción "Aceptar filtros" en la pestaña "Filtros".



La elección de esta opción iniciará el proceso de guardado y validado de los filtros en el programa. Si la introducción de los filtros es correcta se mostrará, en la zona de resúmenes un check positivo junto con los nombres de todos los filtros aceptados. De no ser así, si se detecta algún fallo en la introducción de los filtros se emitirá una señal de error y en la ventana de resúmenes aparecerá un check negativo.



La aceptación correcta de los filtros permitirá al usuario continuar con la definición de los parámetros necesario para el análisis de la imagen introducida. A partir de este punto será posible la definición de las máscaras.

1.1.5 Aplicación de máscaras

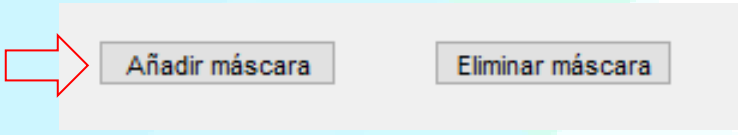
Una vez introducidos y cargados los filtros a utilizar, el siguiente paso en el uso del programa es la definición de las máscaras. Toda esta parte se desarrollará en la pestaña "Filtros" El usuario contará con una pequeña interfaz de ayuda para hacer más cómodo la introducción de las máscaras. Estas podrán ser de dos tipos:

· Máscaras simples: formadas por la combinación de hasta 2 filtros distintos

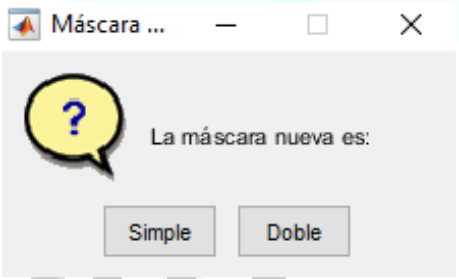
· Máscaras compuestas: formadas por la combinación de 4 filtros distintos

- **Definición de las máscaras**

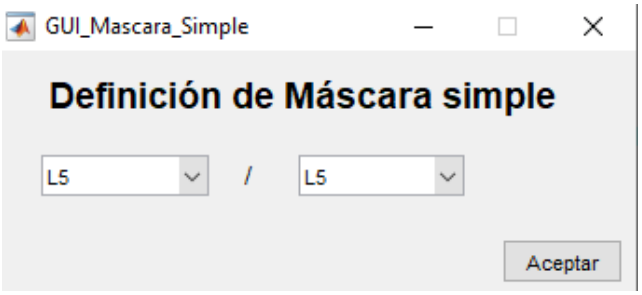
Para introducir de forma manual las máscaras usuario primero deberá elegir la opción "Añadir máscara" dentro de la pestaña "Máscaras".

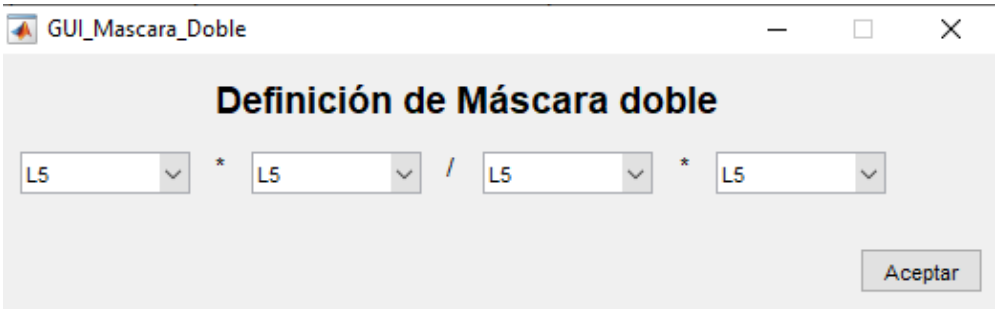


Se despliega entonces una pequeña interfaz inicial donde el usuario deberá determinar si la máscara que va a introducir es simple o compuesta.

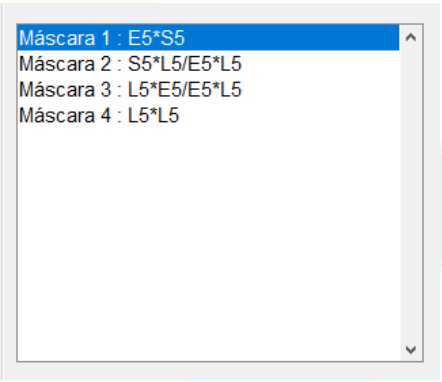


Elegida la opción correspondiente se abrirá una segunda interfaz para determinar qué filtros conforman la máscara (dos filtros para la opción simple, 4 filtros para la opción compuesta).



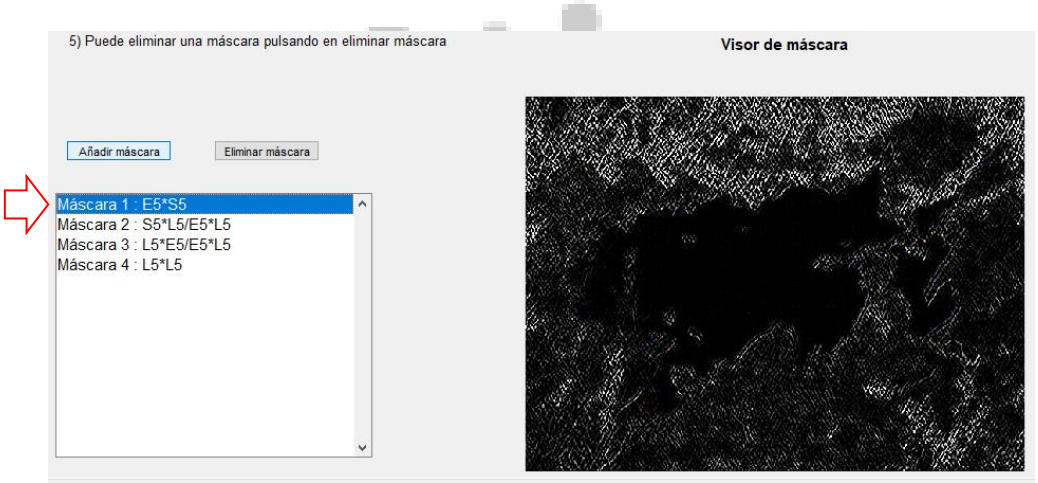


Seleccionando en los desplegables el usuario determinará cómo está compuesta la máscara. Una vez hecho esto se deberá pulsar aceptar. La máscara definida se añadirá al listado de máscaras dentro de la pestaña "Máscaras"



- **Previsualización de máscara**

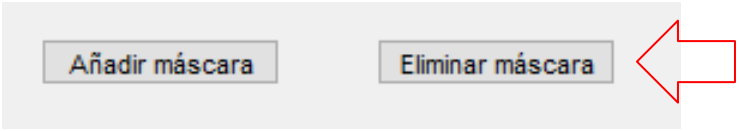
Cada vez que el usuario introduzca una máscara tendrá la posibilidad de poder visualizar su aplicación directa sobre la imagen a analizar. Para ello simplemente se deberá pulsar sobre el nombre de la máscara elegida dentro del listado de máscaras en la pestaña "Máscaras"



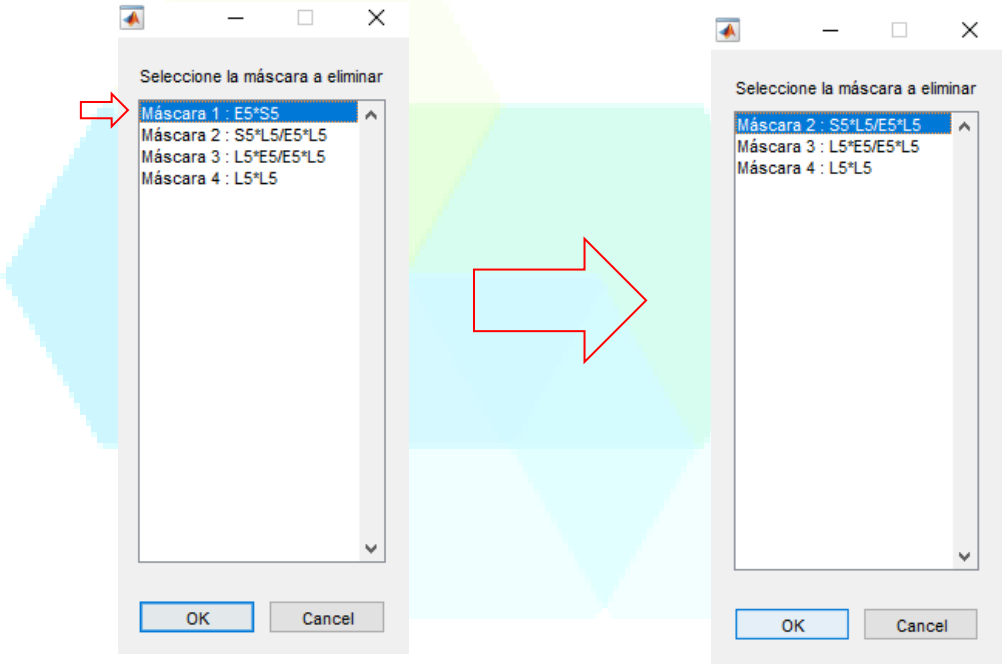
En la esquina inferior derecha de esta pestaña se podrá visualizar la aplicación directa de la máscara elegida en la imagen analizar.

- **Eliminado de máscara**

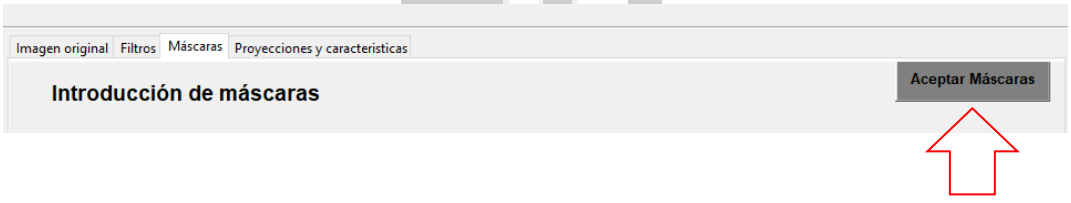
Se presenta la posibilidad al usuario de eliminar una máscara existente dentro de la lista de máscaras definidas. Para ello el usuario deberá pulsar la opción “Eliminar máscara” dentro de la pestaña “Máscaras”



Se despliega una interfaz con el listado de máscaras existentes. Simplemente pulsando sobre el nombre de la máscara que se quiera eliminar y pulsando aceptar esta quedará eliminada de la lista.

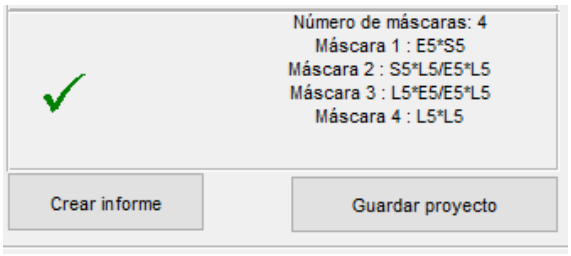


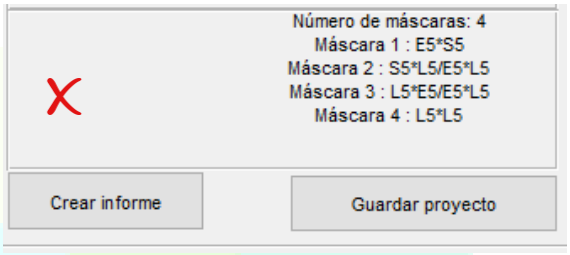
Una vez definidas todos las máscaras de manera correcta, el usuario deberá aceptar las máscaras introducidos guardándolas en el programa (variable Ima). Para ello se elegirá la opción "Aceptar máscaras" en la pestaña "Máscaras".



La elección de esta opción iniciará el proceso de guardado y validado de las máscaras en el programa. Si la introducción de los máscaras es correcta se mostrará, en la zona de resúmenes un check positivo junto con los nombres de todos las máscaras aceptadas. De no ser así, si se detecta algún fallo en la introducción de las máscaras se emitirá una señal de error y en la ventana de resúmenes aparecerá un

check negativo.



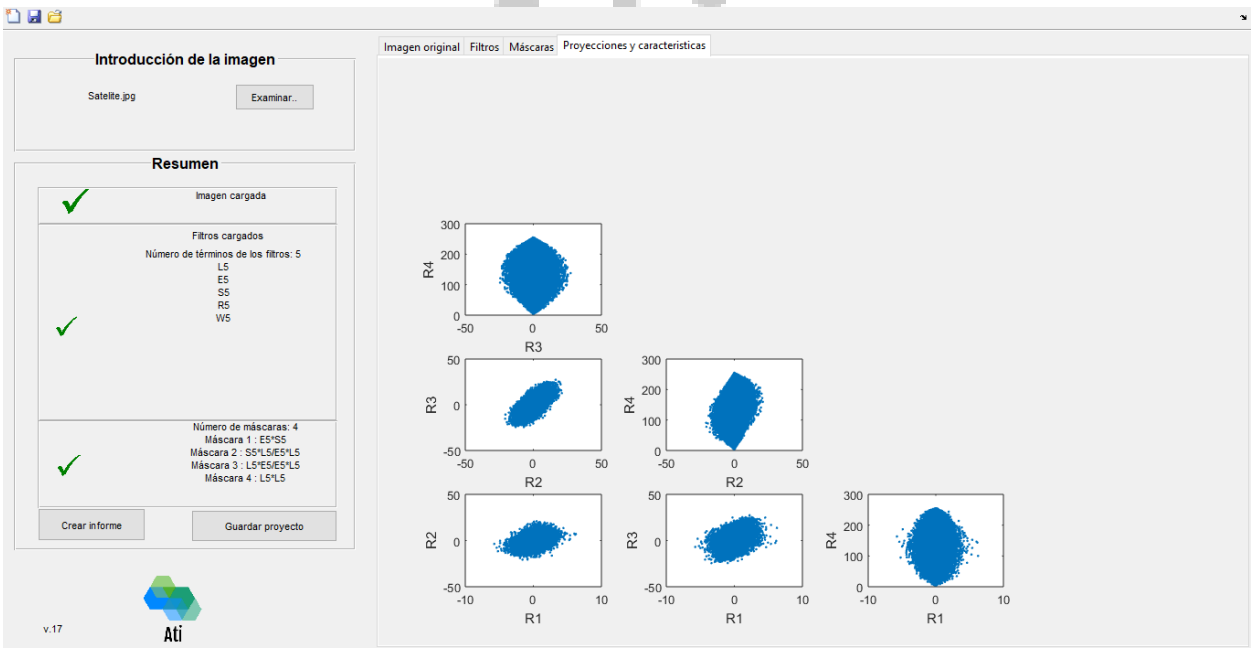


La aceptación correcta de las máscaras permitirá al usuario poder realizar el análisis completo de la imagen mediante el método de Laws.

1.1.6 Generación de resultados e informe

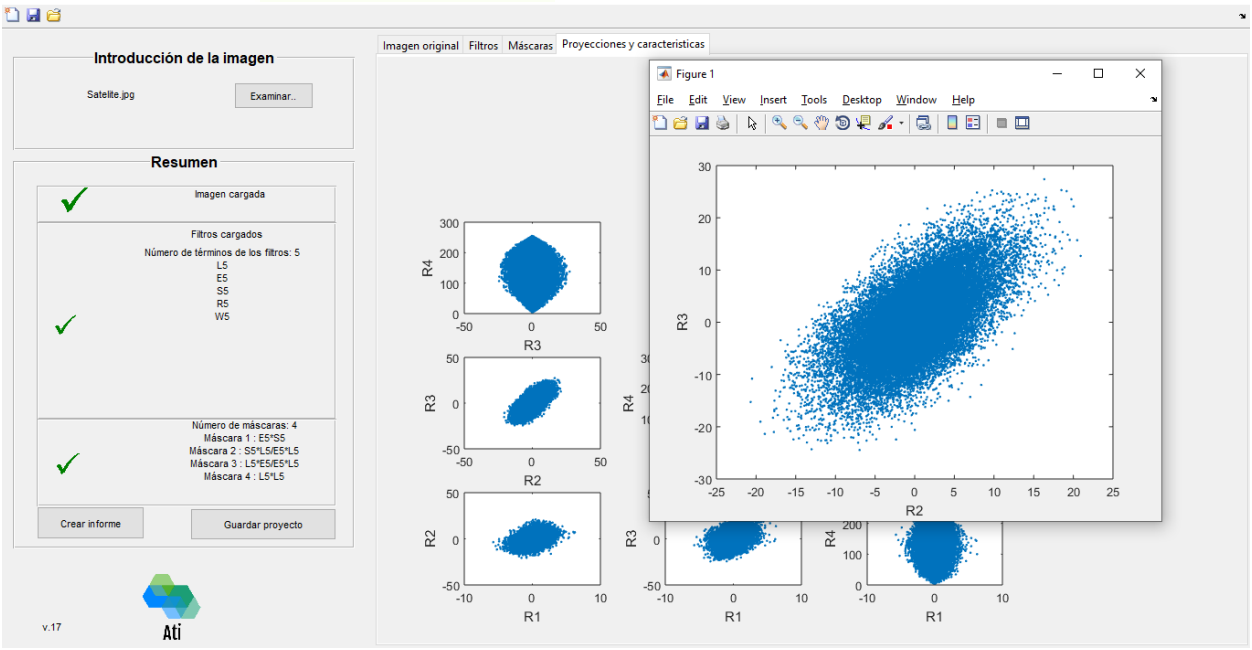
De manera automática, una vez que el usuario pulsa el botón “Aceptar máscaras” estas son cargadas al programa y aplicadas a la imagen a analizar. El método de aplicación es el descrito en la teoría de Laws según su método de análisis de imágenes basado en texturas.

En la interfaz se ha reservado una pestaña para representación de los distintos resultados según las máscaras definidas. Estos resultados se mostrarán en la pestaña *Proyecciones y características*.

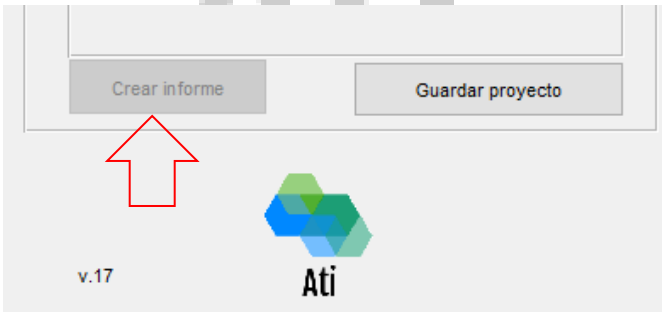


En esta pestaña se mostrará, en una representación piramidal la combinación de todas las máscaras definidas aplicadas a la imagen.

Pulsando en cada gráfica se podrá abrir en un ventana independiente la gráfica seleccionada para poder más en detalle sus resultados



Una vez que los resultados estén calculados se podrá crear un informe automático en PDF que recopile la información de todo el análisis de la imagen utilizado. Para ello se deberá pulsar en el botón "*Generar informe*" en la ventana de resúmenes.



A continuación se creará el informe y será guardado con el nombre del proyecto en la carpeta "*Informes guardados*".

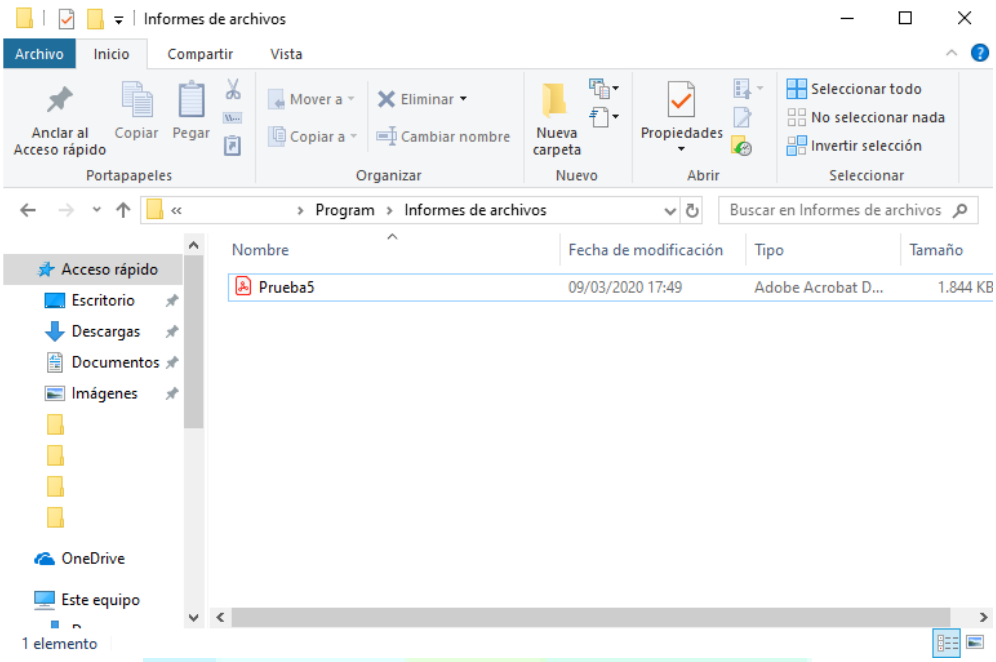

Informes de archivos
Archivo
Inicio
Compartir
Vista
Anclar al Acceso rápido
Copiar
Pegar
Mover a
Copiar a
Eliminar
Cambiar nombre
Nueva carpeta
Propiedades
Seleccionar todo
No seleccionar nada
Invertir selección
Portapapeles
Organizar
Nuevo
Abrir
Seleccionar
Program
Informes de archivos
Buscar en Informes de archivos
Acceso rápido
Escritorio
Descargas
Documentos
Imágenes
OneDrive
Este equipo
Nombre
Fecha de modificación
Tipo
Tamaño
Prueba5
09/03/2020 17:49
Adobe Acrobat D...
1.844 KB
1 elemento

Table of Contents

Informe del análisis de la imagen

Informe Ati

Datos de la imagen

Proyecto: Prueba5 *09-Mar-2020*

1) Imagen
- Nombre de la imagen: Satelite.jpg

Imagen a analizar

Datos de los fitros

2) Filtros
- Número de filtros: 5
- Listado de filtros usados:
· L5 = [1 ,4 ,6 ,4 ,4]
· E5 = [-1 ,-2 ,0 ,2 ,2]
· S5 = [-1 ,0 ,2 ,0 ,0]
· R5 = [-1 ,2 ,0 ,-2 ,-2]
· W5 = [1 ,-4 ,6 ,-4 ,-4]

Datos de las máscaras

3) Máscaras
- Número de máscaras: 4

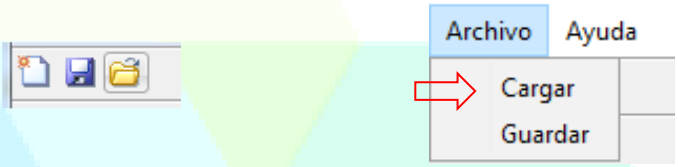
- Listado de máscaras definidas:
*· Máscara 1 : E5*S5*

1.1.7 Guardado y cargado de un proyecto

El programa Ati cuenta con unas herramientas de guardado y cargo de proyectos propias. En cualquier etapa de utilización del programa el usuario podrá pulsar el botón "Guardar" dentro de la barra superior de herramientas para guardar todos los datos introducidos en su proyecto.

Una vez pulsado será necesario introducir un nombre para guardar el proyecto. Se admiten cualquier combinación alfanumérica de caracteres menos el uso del punto "." o cualquier otro carácter que no esté contemplado por Matlab para el proceso de almacenado de variables. Tras guardar el proyecto actual, el usuario podrá continuar con la edición del proyecto y modificar cualquier parámetro del análisis de imágenes.

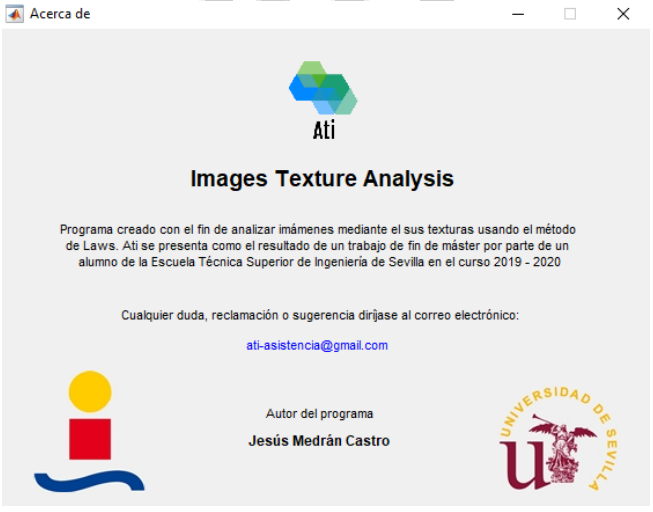
Para poder cargar los datos de un proyecto que previamente ha sido guardado se deberá pulsar el botón "Abrir" dentro de la barra superior de herramientas.



Una vez pulsado el programa redirigirá al usuario a la carpeta "Proyectos guardados" para poder seleccionar el proyecto que desee. El usuario podrá cargar cualquier proyecto almacenado en su ordenador siempre que se trate de un archivo ".mat" y con las características específicas requeridas por el programa Ati. Una vez cargado el proyecto se añadirán a la interfaz todas las variables que este contenga (imagen, filtros, máscaras, etc.). Será necesario de nuevo aplicar las máscaras de forma manual para poder obtener los resultados del programa.

2 CONTACTO

Para resolver cualquier duda, reclamación sobre el programa o solicitar algunas sugerencias para futuras mejoras diríjase a la dirección de correo electrónico: ati-asistencia@gmail.com



Ati

7 BIBLIOGRAFÍA

[1] T. y. A. Departamento de Ingeniería electrónica y Á. d. I. d. S. y. A. , *Detección de bordes en,* Universidad de Jaén.

[2] C. A. Cáceres Flórez, D. Amaya Hurtado y O. L. Ramos Sandoval , «Procesamiento de imágenes para reconocimiento de daños causados por plagas en el cultivo de Begonia semperflorens (flor de azúcar),» *Sanidad Vegetal y Protección de Cultivos / Plant and Crop Protection,* 2014.

[3] G. Cid Espinosa, Programación de interfaz gráfica en App designer para control vectorial de motores de imanes permanentes, 2018.

[4] R. González y P. Wintz, «Tema 3 y 4,» de *Procesamiento digital de imágenes*, 1996, pp. pág 89-269.

[5] J. F. Escribano Molina, Desarrollo de una interfaz gráfica en Matlab para la aplicación de modelos de regresión local polinómica, 2009.

[6] G. Fernández de Córdoba Martos, Creación de interfaces gráficas de usuario (GUI) con Matlab, 2007.

[7] P. Corcuera, *Creación de interfaces de usuario con Matlab,* Dpto. Matemática Aplicada y Ciencias de la Computación. Universidad de Cantabria.

[8] N. Aguirre Dobernack, «Capítulo 3. Procesamiento de imágenes».

[9] J. Gomez Labat, Desarrollo de unaa interfaz gráfica de usuario para el control de analizadores de espectros ópticos mediante Matlab, 2016.

[10] M. E. Borrero Serrano, «Capítulo 3,» de *Diseño de interfaz gráfica en Matlab*, pp. pág 56-74.

[11] J. Buhigas Pérez, Desarrollo de una aplicación en interfaz gráfica de Matlab para la determinación del comportamiento dinámico de un vehículo automóvil, 2011.

[12] M. G. Forrero Vargas y E. A. Arias Cruz, «Estudio del efecto de las máscaras de convolución en imágenes mediante uso de la trasnformada de Fourier,» *Ingeniería e Investigación,* nº 48, 2001.

[13] F. Giménez-Palomares, J. Monsoriu y E. Alemany-Martínez, «Aplicación de la convolución de matrices al filtrado de imágenes,» *Modelling in Science Education and Learning,* 2016.

[14] B. Medrano Garfia, *Procesamiento de imágenes con Matlab,* Dpto. Matemática Aplicada 1.

[15] Y. Alvarez Germade, E. Barbará Morales y O. Rodrigez Ramirez, «Filtrado digital en el procesamiento de imágenes empleando matlab,» de *Conferencia científica de ingeniería y arquitectura*, 2010.

[16] F. Ibarra Picó, Análisis de texturas mediante coeficiente morfológico modelado conexionista aplicado, 1995.

[17] B. Paniagua Paniagua, M. A. Vega Rodríguez y P. Bustos García, Análisis avanzado de texturas para la detección automática de la calidad del corcho.