UNIVERSIDAD DE QUINTANA ROO
DIVISIÓN DE CIENCIAS E INGENIERÍA

PROCESAMIENTO DE IMÁGENES PARA LA CARACTERIZACIÓN DE CELDAS DE COMBUSTIBLE

TRABAJO DE TESIS
PARA OBTENER EL GRADO DE

INGENIERO EN REDES

PRESENTA
JOSÉ ALBERTO TORRES POZOS

DIRECTOR DE TESIS
DR. JAIME S. ORTEGÓN AGUILAR

ASESORES

DR. GLISERIO ROMELI BARBOSA POOL
DR. JAVIER VÁZQUEZ CASTILLO
DR. HOMERO TORAL CRUZ
MSI. RUBÉN ENRIQUE GONZÁLEZ ELIXAVIDE

Universidad de Quintana Roo
DCI DIVISIÓN DE CIENCIAS E INGENIERÍA

CHETUMAL QUINTANA ROO, MÉXICO, NOVIEMBRE DE 2017

UNIVERSIDAD DE QUINTANA ROO
DIVISIÓN DE CIENCIAS E INGENIERÍA

TRABAJO DE TESIS ELABORADO BAJO SUPERVISIÓN DEL COMITÉ DE ASESORÍA Y APROBADO COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE:

INGENIERO EN REDES

COMITÉ DE TRABAJO DE TESIS

DIRECTOR: _____
DR. JAIME S. ORTEGÓN AGUILAR

ASESOR: _____
DR. GLISERIO ROMELI BARBOSA POOL

ASESOR: _____
DR. JAVIER VÁZQUEZ CASTILLO

CHETUMAL QUINTANA ROO, MÉXICO, NOVIEMBRE DE 2017

UNIVERSIDAD DE QUINTANA ROO

División de Ciencias e Ingeniería

Agradecimientos

Para mis padres que con mucho esfuerzo y trabajo han hecho este sueño una realidad. Con ese gran apoyo y amor es como me motivaron y me guiaron por el camino indicado. Todos mis logros se los debo a ellos.

Gracias Madre

Gracias Padre

Un especial agradecimiento al CONACYT por el apoyo económico brindado bajo los proyectos SEP-CB-2013/221988, PDCPN-2013/213373 y LENERSE SENER-CONACYT 254667. Asimismo, se agradece a la Red de energías renovables de la Península de Yucatán y a la División de Ciencias e Ingeniería de la UQROO.

UNIVERSIDAD DE QUINTANA ROO

División de Ciencias e Ingeniería

Dedicatoria

Primeramente quisiera agradecer a la Universidad de Quintana Roo (UQRoo) por aceptarme y porque me dieron la oportunidad de ser parte de esta gran institución. Al igual que a todos los docentes quienes me apoyaron y tomaron de su tiempo para poder enseñarme sus conocimientos.

Agradezco también a mi asesor de Tesis el Dr. Jaime Ortegón porque me dio la oportunidad de ser parte de este proyecto fenomenal. También por tener paciencia en guiarme hasta la conclusión de este trabajo.

También quisiera agradecerle al Mto. Rubén Elixavide quien con sus palabras de motivación me ayudaron y me empujaron en la realización de esta tesis.

Finalmente quisiera agradecer a todos que de una manera u otra fueron parte de mi vida Universitaria. Gracias a todos por el apoyo y la amistad.

Resumen

El procesamiento de imágenes para la caracterización de electrodos de celdas de combustible es un paso importante, ya que nos empieza a mostrar nuevos métodos y soluciones globales para el uso de hidrógeno como un combustible renovable. La tecnología de celdas de combustible empieza a ser fuertemente considerada para implementar un sistema energético, el cual puede ayudar en muchos paradigmas mundiales, como la reducción de la contaminación, el desarrollo sustentable, la resiliencia de las ciudades, entre otras.

De tal forma, este proyecto apoya en el estudio de las celdas de combustible. Para estudiar las imágenes de microscopia de los electrodos de las celdas de combustible, actualmente no existe un software comercial exclusivo, y de tal forma, este proyecto presenta una implementación para poder estudiar a más detalle y cuantificar los cambios de las imágenes obtenidas de los electrodos de las celdas.

UNIVERSIDAD DE QUINTANA ROO

División de Ciencias e Ingeniería

Contenido

CAPÍTULO 1

CAPÍTULO 1

Introducción

El procesamiento de imágenes significa que las señales de imágenes pueden ser transferidas hacia un entorno de datos numéricos, y luego los datos pueden ser analizados por una computadora (Liu Jianjun, 2011). Esto ha desarrollado en las últimas décadas como una nueva técnica o enfoque de desarrollo para un gran número de aplicaciones en una gran cantidad de campos de estudios. Esto se debe a la evolución de la tecnología y a la gran cantidad de instrumentos de visión como la cámara fotográfica digital que al mismo tiempo son normalmente de bajo costo y de gran disponibilidad. Normalmente el procesamiento de imágenes consiste con mapeos de imágenes bi-dimensionales, en donde cada punto consiste de información numérica en términos de intensidad, amplitud, escala de colores de grises o de algún modelo de color. De tal forma el procesamiento de imágenes toma el conocimiento estructural de la imagen como entrada y produce como salida un grupo de parámetros y la función relacionada con el contenido de la imagen. El paso fundamental, para llegar desde la entrada a la salida, está constituido por una serie de operadores, algoritmos o técnicas de procesamiento; los cuales se refieren a la modulación de brillo y contraste de imágenes. Con estas técnicas, es posible calcular la porosidad, distribución de longitud, distribución de tamaño de poros y tosquedad al igual que el grado de conectividad entre poros y correlacionar estos para calcular las propiedades físicas de la dinámica de fluidos. Con el avance en el procesamiento de imágenes, éste se ha adoptado para caracterizar la estructura y las propiedades de volumen (Vincenzo Guarino, 2010).

La presente tesis tiene como objetivo final la implementación de un programa que aplique algoritmos matemáticos sobre imágenes tomadas con SEM (microscopio electrónico de barrido - Scanning Electron Microscope). Estos algoritmos llevarán a dar la caracterización de imágenes, para luego utilizar los resultados para determinar los coeficientes de correlación. Los coeficientes determinados serán de gran utilidad para la caracterización de electrodos de celdas de combustible.

Las imágenes tomadas con SEM muestran imágenes de celdas de combustible con una gran magnificación, en donde se logran ver poros que se encuentran dentro de una celda de combustible. De tal forma, la capa del catalizador tiene una gran influencia en el rendimiento del intercambio de protones de una celda de combustible. Esto se debe a que el componente no solo proporciona sitios reactivos, sino también, su estructura define la eficiencia de muchos de los procesos

que intervienen durante el funcionamiento del intercambio de protones, en particular, el fenómeno de transporte. De esta forma, la eficiencia global de intercambio de protones se puede ver cuando ciertas estructuras alcanzan en sus propiedades la conducción eléctrica e iónica, transporte de líquido y gas, y cuando sus propiedades catalíticas se definen para una condición de funcionamiento dada (R. Barbosa, 2011).

La distribución de tamaño de poros y cantidad de poros por imagen son parámetros críticos para poder determinar la idoneidad de estructuras anódicas alúminas para muchas aplicaciones. La caracterización de distribución de tamaños de poros normalmente se obtiene mediante la dispersión y algunos métodos basados en el análisis de la imagen. Las imágenes de pruebas obtenidas del SEM muestran poros bien definidos, al igual que poros en formaciones imperfectas, los cual complican el procesamiento. Aunque sean fáciles de identificar por el ojo humano, la incertidumbre de la delineación manual hecho por un experto y la pequeña muestra de medición de poros son algunos de los inconvenientes del software de procesamiento de imágenes. La interfaz desarrollada ayudará al procesamiento de las celdas de combustible para poder llevar a cabo un estudio a mayor profundidad.

Justificación

Esta tesis forma parte de un proyecto que propone el desarrollo e implementación de técnicas experimentales de manufactura y técnicas numéricas de simulación, para determinar coeficientes efectivos de transporte en electrodos de celdas de combustible. La técnica actual de manufactura de electrodos porosos para celdas de combustible, será analizada mediante microscopia electrónica de barrido (SEM). Las imágenes obtenidas en el SEM, serán procesadas mediante técnicas matemáticas y reconstruidas mediante técnicas estocásticas para cuantificar científicamente la variación micro estructural y su influencia sobre el desempeño eléctrico global. Consecuentemente este proyecto permitirá la innovación de electrodos de sistemas de energía renovable con mejores propiedades físicas y electroquímicas.

El contar con modelos permitirá evaluar procesos de manufactura y pretende determinar coeficientes de un procesamiento digital. Se implementa una imagen y de sus pixeles se debe poder determinar las fases vacías y sólidas de dicha imagen mediante un procesamiento de la imagen, para poder determinar los coeficientes de correlación.

OBJETIVO GENERAL

Desarrollar una herramienta de cómputo para sementar y caracterizar estadísticamente imágenes de electrodos de celdas de combustible, en un entorno Matlab.

OBJETIVOS PARTICULARES

- Segmentar y caracterizar estocásticamente imágenes de celdas de combustible.
- Desarrollar una herramienta informática en entorno Matlab para llevar a cabo la caracterización.
- Implementación de algoritmos para llevar a cabo métodos que ayuden a el procesamiento y caracterización de imágenes.
- Presentar las técnicas y el código de la herramienta del procesamiento y caracterización de imágenes.

- Presentar evidencia de su funcionamiento con imágenes reales.
- Obtener resultados finales de una imagen binarizada, entrenada y caracterizada para lograr determinar los coeficientes de un procesamiento digital de imágenes.

ALCANCE

Desarrollar una interfaz en Matlab para hacer la caracterización estocástica de celdas de combustible mediante el procesamiento de imágenes.

Algunas restricciones se realizan al no tener una imagen resultado para realizar comparaciones. De tal forma el resultado de las imágenes depende a gran factor del individuo realizando el procesamiento de imágenes y su satisfacción en el resultado.

CAPÍTULO 2

CAPÍTULO 2

MARCO TEÓRICO

Dentro del software elaborado para el procesamiento de imágenes, se encontrarán frases o definiciones que son muy importantes. Por esta razón en las siguientes secciones de detalla un vocabulario básico, que enmarco los fundamentos de esta tesis de licenciatura.

Imagen

Una imagen es una representación visual de algún objeto. En informática una imagen es una figura que se ha creado o copiado y guardado en alguna forma electrónica. Esta imagen se puede describir en términos de vectores y tramas de gráficos. El mapeo de una imagen es un archivo el cual contiene información el cual es asociado a las diferentes localizaciones específicas de la imagen.

De tal forma una imagen es cualquier función real i(x,y) con integral finita. Cada posición (x,y) corresponde a una posición dentro de la imagen conocida como pixel (Azuela, 2006).

Pixel

Un pixel es un punto físico dentro de una imagen normalmente representados como un cuadro o circulo el cual contiene información única de tal posición de la imagen.

Para poder utilizar la información de la imagen a más detalle se realiza un proceso de imagen para poder obtener su resultado binarizado.

Imagen binarizada

Una imagen binaria denotado cono b(x,y) es una imagen digital f(x,y) que ha sido cuantificada a dos niveles de intensidad, 0 y 1 en este caso.

La imagen digital (figura 1), se muestra junto con su resultado binario (figura 2).

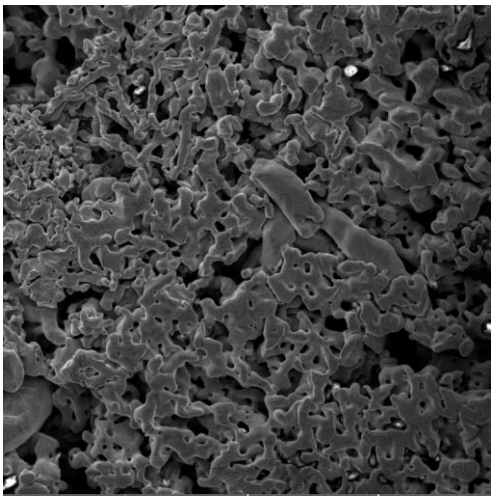


Figura 1. Imagen original del SEM digital

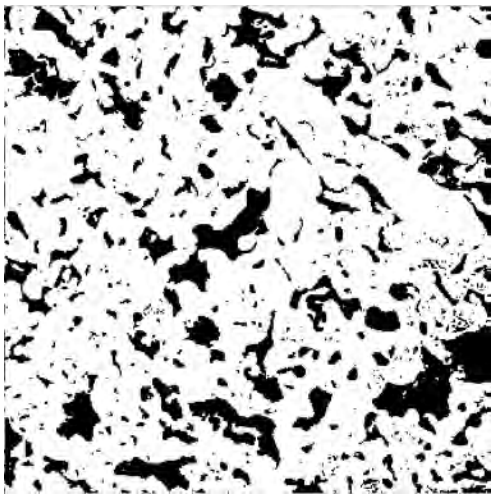


Figura 2. Imagen binarizada resultante

Matlab

Matlab es un lenguaje de programación de alto nivel de cuarta generación, comercial, con un ambiente muy amigable el cual se enfoca más a un entorno analítico y matemático desarrollado por MathWorks. En la figura 3, se presenta la pantalla de inicio de la versión 2015ª.

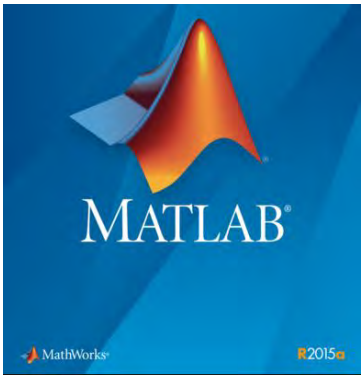


Figura 3: Logotipo del software MATLAB

El lenguaje de programación Matlab es destinado a un uso de computación numérica, pero hoy en día con su popularidad en la comunidad, se ha desplegado a varias ramas como procesamiento de imágenes y señales, comunicaciones, sistemas de control de industrias, diseño de rejillas inteligentes, robótica, computación financiera, entre varias más. Esta herramienta es utilizada mayormente por ingenieros, científicos, estudiantes y maestros.

Matlab se ha vuelto popular entre la comunidad de programación dado por su facilidad de utilizar y su flexibilidad de ser tipo multiplataforma en los sistemas operativos más comunes como Windows, Mac y Linux. Su popularidad creció igual al tener la habilidad de interactuar con otros lenguajes de programación como C, C++, Fortran, Java, entre otros (MATLAB, 2015).

GUI Layout

El diseño de interfaz gráfica por sus siglas en ingles GUI Layout (Graphical User Interface Layout) es un ambiente incluido en Matlab para diseñar gráficamente las interfaces. La figura 4 presenta un ejemplo de una interfaz gráfica para un programa hecho con Matlab.

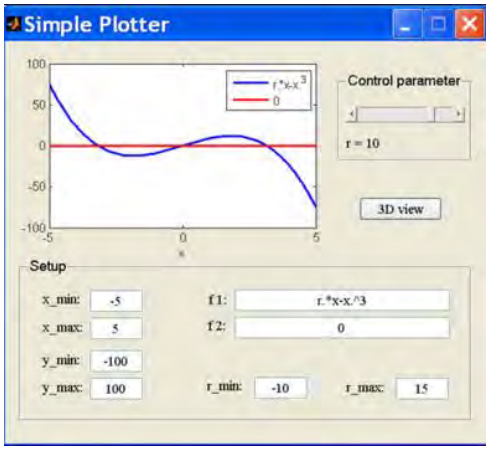


Figura 4: Interfaces mediante manipulación de elementos

Las interfaces gráficas se crean por manipulación directa con los elementos gráficos como los botones, paneles, etc. Matlab permite realizar interfaces gráficas de dos formas, ya sea por manipulación de los elementos o por código puro para crear los elementos de las interfaces. La figura 5 presenta un ejemplo de una interfaz construida visualmente, mientras que la figura 6, el código utilizado para crear una interfaz.

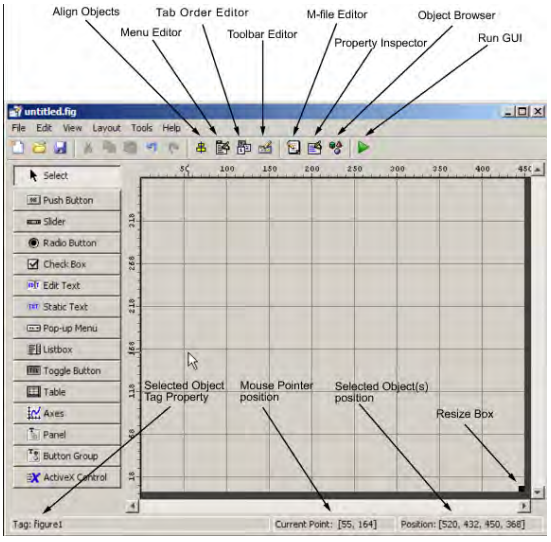


Figura 5: Interfaces mediante manipulación de elementos

```
function ImageProcessor()

% GUI Structures
gui = struct();
data = struct();
% Launch the GUI
data = createData();
gui = createGUI();

    function data = createData()
%
        data.LocalMainDir = pwd;
        data.himage = 0;

    end

    function gui = createGUI()
        gui.rootWindow = figure('Name'              ,'Image Editor' ,...
                                'MenuBar'           ,'none'         ,...
                                'ToolBar'           ,'none'         ,...
                                'NumberTitle'       ,'off'          ,...
                                'Position'          ,[100, 100, 1800, 900]);

        gui.mainvb = uiextras.HBoxFlex('Parent'     ,gui.rootWindow ,...
                                'Spacing'     ,3);

        gui.panelFile = uiextras.BoxPanel('Parent'  ,gui.mainvb     ,...
                                'Title'             ,'Archivos'     ,...
                                'MinimizeFcn'       ,@Minimize);
        gui.vb1 = uiextras.VBoxFlex('Parent'        ,gui.panelFile  ,...
                                'Spacing'           ,4);
        gui.vb2 = uiextras.VBoxFlex('Parent'        ,gui.mainvb     ,...
                                'Spacing'           ,4);
        gui.trainPanel = uiextras.BoxPanel('Parent' ,gui.mainvb     ,...
                                'Title'             ,'Datos'        ,...
                                'MinimizeFcn'       ,@MinimizeA);
        gui.vb3 = uiextras.VBoxFlex('Parent'        ,gui.trainPanel ,...
                                'Spacing'           ,4);
```

Figura 6: Interfaces mediante código puro

Los elementos gráficos utilizados dentro del programa gráfico propuesto fueron:

- Panel. Organiza un único elemento, este puede contener a su vez otros elementos, y puede tener métodos que le permitan desprenderse de una ventana, minimizarse o maximizarse.
- Box. Existen diversos tipos de Box (Cajas) que funcionan como elementos contenedores y no proveen ningún punto de interacción con el usuario.
    - HBox – es un diseño de caja horizontal que organiza a los componentes hijo de forma horizontal.
    - VBox – es un diseño de caja vertical que organiza a los componentes hijo de forma vertical.
    - HBoxFlex – es un diseño de caja horizontal que organiza a los componentes hijo de forma horizontal con divisores arrastrables o flexibles, lo que permite modificar el tamaño de cada uno de los componentes hijo.
    - VBoxFlex – es un diseño de caja vertical que organiza a los componentes de forma vertical con divisores arrastrables o flexibles, lo que permite modificar el tamaño de cada uno de los componentes hijo.

Los objetos o widgets utilizados dentro del programa gráfico fueron:

- Button. Son los botones de la aplicación, existen diversos tipos de botones, pero en esta propuesta solo se usa los pushbutton.
    - Pushbutton – botón que genera algún tipo de acción cuando activado mediante un clic-izquierdo del ratón.
- Popupmenu. Es un tipo de menú que muestra las diferentes opciones al abrirla mediante presionar el botón.
- Checkbox. Elemento que permite activar/desactivar una opción, este genera su acción al seleccionarla y proporciona un número de opciones independientes.
- Axes. Objeto gráfico de ejes en la figura actual, muy útil para dibujar funciones.
- Folder. Crea y regresa la ruta de una carpeta especificada.
- Folderlist. Muestra gráficamente el contenido de las carpetas de la computadora actual.

La figura 7 presenta la aplicación propuesta, indicando los elementos utilizados.

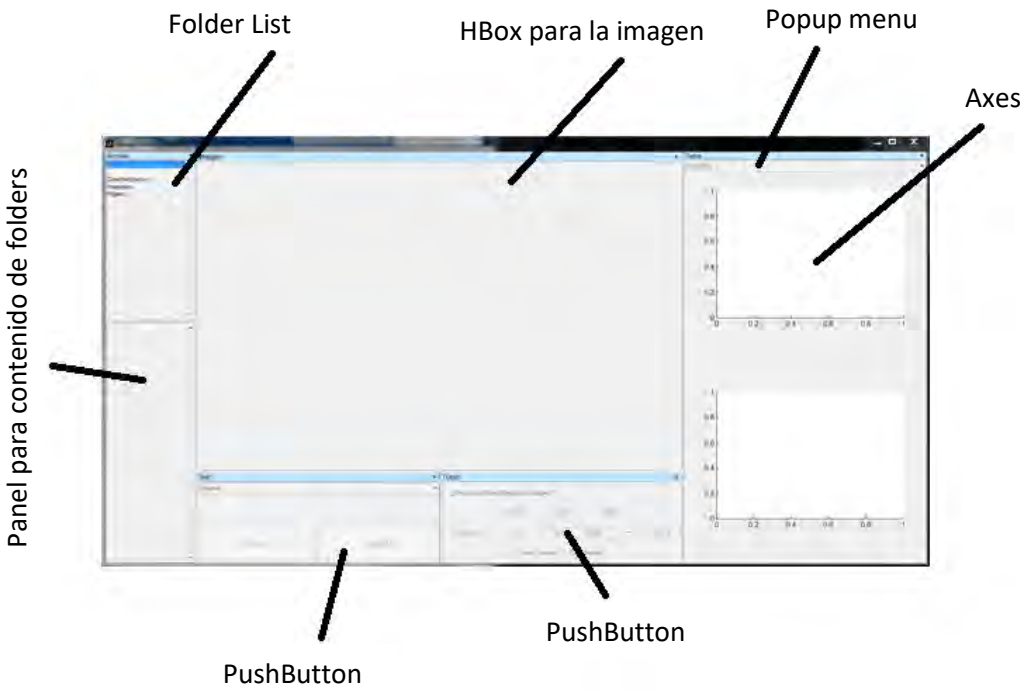


Figura 7: Interfaz gráfica con sus elementos

Histograma

El histograma de una imagen *I* es una gráfica que representa los niveles de intensidad del color de *f* con respecto al número de pixeles presentes en f con cada intensidad de color. La figura 8 presenta un ejemplo de un histograma.

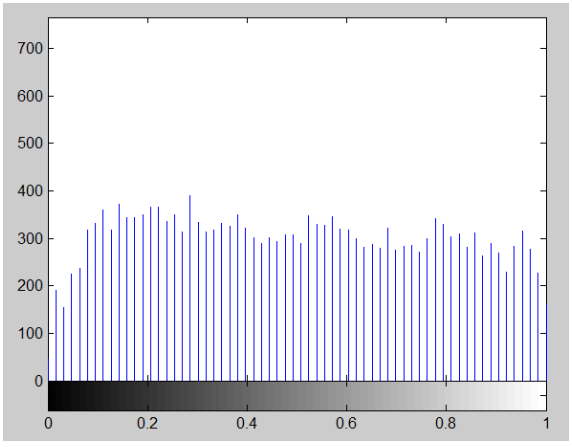


Figura 8: Ejemplo de un histograma

Algunas propiedades del histograma:

- Dos imágenes diferentes pueden tener asociado el mismo histograma

- Los histogramas no contienen información especial sobre la imagen, más allá de la distribución (frecuencia) del color.

Ecualizador

La ecualización es el proceso de ajustar los valores de intensidad y distribuirlos de manera uniforme en la imagen.

Esto también se puede ver como una transformación de una imagen, la cual culmina su histograma con una distribución uniforme. De tal forma se logra una distribución más uniforme entre el número de pixeles asociado a cada nivel de intensidad (BUAP).

La ecualización de un histograma de una imagen se puede obtener mediante la siguiente ecuación.

$$m_k = \frac{1}{tamaño} \sum_{j=1}^{k} n_j$$

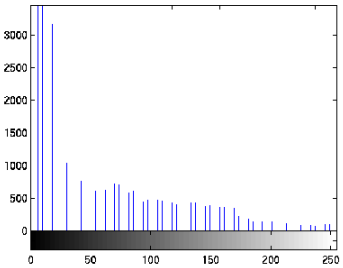
$$donde\ k\ son\ valores\ en\ el\ intervalo\ [0 \ldots . q-1]$$

$$q\ es\ el\ número\ de\ intensidades\ asociados\ a\ la\ imagen$$

$$n\ es\ el\ vector\ de\ frecuencias\ dentro\ de\ la\ imagen$$

En la figura 9 se presenta la imagen original con su histograma, mientras que la figura 10 muestra la imagen ecualizada con su respectivo histograma.

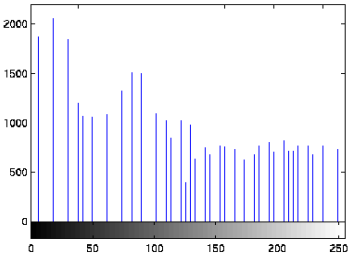
a)



b)

Figura 9: a) Imagen original b) histograma asociado

a)



b)

Figura 10: a) Imagen ecualizada b) histograma asociado

Normalizador

La normalización es el proceso de ajustar los valores ecualizados a un rango especifico. Esto permite que imágenes de diferentes fuentes o formatos puedan ser procesados de manera similar. En esta aplicación se normalizó al rango entre 0 y 1.

Gradiente

El gradiente es una operación vectorial que opera sobre una función escalar, para producir un vector cuya magnitud es la máxima razón de cambio de la función en el punto del gradiente y que apunta en la dirección de ese máximo.

También se define como un vector, en donde sus componentes miden la rapidez en que los valores de los pixeles cambian en la distancia y en las direcciones x e y.

Para esta tesis, se utilizan los gradientes de magnitud y de dirección. La magnitud del gradiente nos dice que tan rápido está cambiando la imagen mientras la dirección del gradiente nos dice la dirección en el cual la imagen está cambiando más rápido.

En la figura 11 se puede apreciar el resultado de imagen al visualizar las gradientes de magnitud y dirección. La imagen a la derecha nos muestra el gradiente en dirección x e y. Para esta tesis, solo se tomó en cuenta una de las direcciones, la dirección x. Estos datos ayudarán a mejorar el resultado de entrenamiento para el procesamiento de imágenes.

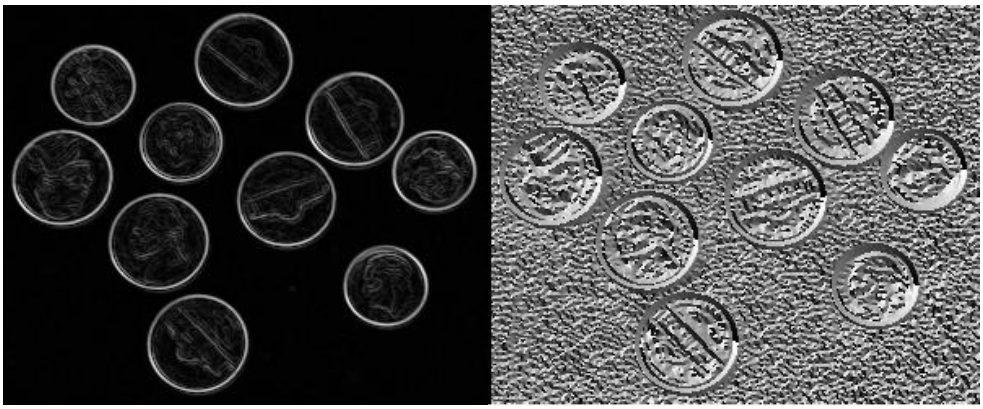


Figura 11. Ejemplo de la visualización de la magnitud y dirección del gradiente

SVM

Las Máquinas de Vectores Soporte (SVM, Support Vector Machines) son algoritmos para el aprendizaje automático. Son grupos de métodos de aprendizaje supervisado que se puede aplicar a la clasificación o regresión.

Las máquinas de vectores de soporte (SVM) poseen la habilidad de ser aproximadores universales de cualquier función multivariada hasta una precisión deseada. Por esto, son de particular interés para el modelado de sistemas complejos desconocidos, parcialmente desconocidos o altamente no lineales. Es importante resaltar que las SVM no se deben utilizar cuando exista un modelo que sea bueno y confiable analíticamente. Las SVM también son llamadas modelos “no paramétricos” debido a que sus parámetros no son predefinidos y su número depende de los datos de entrenamiento utilizados. En otras palabras, los parámetros del modelo se calculan a partir de los datos, como una forma de acercar la capacidad del modelo a la complejidad de los datos (Wang, 2005).

Las SVMs son un grupo de métodos de aprendizaje supervisado que se puede aplicar a la clasificación, regresión y análisis de datos mediante algoritmos de aprendizaje automático. Las SVM se pueden utilizar cuando los datos tienen exactamente dos clases y la clasificación consiste en la búsqueda del mejor (hiper)plano que separa todos los puntos de datos de una clase de los de otra clase.

En los casos de reconocimiento de patrones más simples, las SVM usan un hiperplano para separar linealmente los datos y crear un clasificador con un margen máximo. En los casos donde las clases no son linealmente separables en el espacio de entrada original, la SVM primero transforma el espacio de entrada a un espacio de una mayor dimensión. Esa transformación se puede lograr a través de varios mapeos no lineales: polinomial, sigmoidal, o diferentes funciones spline. En resumen, una máquina de vectores de soporte, toma datos que posiblemente no sean separables linealmente, los lleva a un espacio dimensional superior a través de un mapeo, y en este espacio encuentra un hiperplano para las clases obteniendo el mayor margen posible; esto basándose en los vectores de soporte que son aquellos vectores de entrada que definen el margen (Jaime Ortegon, 2015).

Binarización

La binarización es la conversión de una imagen en dos valores, representados generalmente en blanco y negro.

La binarización de imágenes es una técnica del procesamiento de imágenes que consiste en un proceso de reducción de la información de una imagen digital a dos valores: 0 (negro) y 255 (blanco). Esta técnica consiste en comparar cada pixel de la imagen con un determinado umbral. El umbral es el valor límite que determina sí un pixel será de color blanco o negro. Los valores de la imagen que sean mayores que el umbral toman un valor 255 y el resto de pixeles toman el valor 0. Como se puede ver en la Figura 12.

Para esta tesis el umbral depende del individuo que realiza el procesamiento de imágenes. Todo dependerá de la selección de regiones aplicadas en la imagen.

Figura 12. Imagen Binarizada

Caracterización

En imágenes, la caracterización es el proceso de dividir una imagen en distintos grupos donde cada grupo representa información única de la imagen procesada.

De la caracterización se obtiene la información necesaria para poder estudiar a más profundidad las celdas de combustible, mediante algoritmos que calculan líneas y círculos posibles dentro de una región de pixeles de valor 1 de una imagen binarizada.

Con el propósito de caracterizar estadísticamente la microestructura de un material heterogéneo, como primer paso, definimos funciones de indicador para cada fase j ∈ [0, J - 1], donde J es el número total de fases y j = 0 indica la fase vacía. Por ejemplo, para la fase j, la función del indicador en el punto descrito por x se define como:

$$I_j(x) = \begin{cases} 1 \; si \; x \; se \; encuentra \; en \; la \; fase \; j \\ 0 \; de \; otra \; manera \end{cases}$$

Las funciones de correlación de dos puntos $S_j^{(2)}(x_a.x_b)$ es la probabilidad de que dos puntos finales x_a y x_b de un segmento de línea, con longitud $r = |x_a - x_b|$, caen en la misma fase j:

$$S_j^{(2)}(x_a.x_b) \equiv \langle I_j(x_a), I_j(x_b) \rangle ,$$

Donde ⟨·⟩ representa el valor esperado de su argumento. Para un medio estadísticamente homogéneo e isotrópico, la función de correlación de dos puntos no dependerá de las posiciones específicas de los puntos finales, pero solo en la distancia entre ellos:

$$S_j^{(2)}(x_a.x_b) \rightarrow S_j^{(2)}(r) \equiv \langle I_j(x), I_j(x+r) \rangle ,$$

Donde r es un vector de posición relativa con la magnitud r. La función de correlación de trayecto de línea $L_j(x_a, x_b)$ es la probabilidad de que una línea segmento con puntos finales x_a y x_b con longitud $r = |x_a - x_b|$, se encuentra completamente en fase j:

$$L_j(x_a, x_b) \equiv \langle \left| \frac{1}{r} \int_0^1 I_j(x + \alpha r) d\alpha \right| \rangle ,$$

donde α es una variable de integración. De nuevo, para un medio estadísticamente homogéneo e isotrópico, la función de correlación de trayectoria de línea ya no

depende de las coordenadas específicas de x_a y x_b. Bajo estas circunstancias, esta correlación la función es solo una función de la longitud r del segmento de línea:

$$L_j(x_a, x_b) \rightarrow L_j(r) \equiv \langle \left| \frac{1}{r} \int_0^1 I_j(x + \alpha r) d\alpha \right| \rangle ,$$

Donde r es un vector de posición relativa con magnitud r y una dirección arbitraria.

CAPÍTULO 3

CAPÍTULO 3

IMPLEMENTACIÓN

El proyecto dio inicio con un código de forma no gráfica, pero con el mismo concepto de poder procesar una imagen. La idea era poder establecer los componentes requeridos antes de construir la interfaz gráfica. Se observó su funcionamiento para poder elegir el componente más eficiente y de tal forma funcionó como una prueba de funcionamiento. En esta prueba se establecieron métodos para poder aplicar la ecualización, normalización, SVM, binarización. Del resultado obtenido de la imagen binarizada se realizaban observaciones a la eficiencia del código actual para mejorar o implementar nuevos métodos.

El proceso de binarizar la imagen era de forma manual, lo cual no era de tiempo eficiente. Esto requería seleccionar regiones de poros y sólidos manualmente mediante observaciones a la imagen y tomando nota de la posición inicial y final para su selección. Una vez establecidos los métodos a utilizar para el procesamiento de imágenes, se empezó con el diseño de la interfaz gráfica. Se establecieron posiciones para los objetos necesarios dentro de la interfaz

Inicialmente, se implementa la interfaz gráfica en la herramienta de MATLAB, el cual nos brindará la facilidad y aproximación de resultado de una imagen. Mediante el código, se elabora la interfaz gráfica necesaria para este proyecto. MATLAB nos permite poder crear una interfaz gráfica ejecutable con base en líneas de textos compiladas sobre el mismo software MATLAB. El resultado de GUI final se puede apreciar en la figura 13.

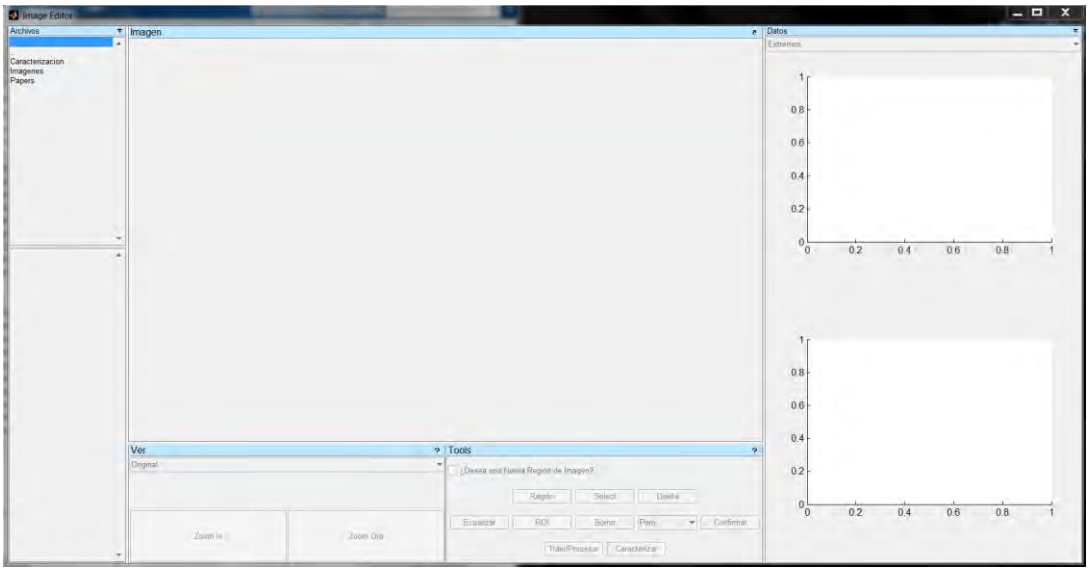


Figura 13: Vista frontal del Software

La interfaz gráfica nos proporciona un entorno fácil de utilizar y con objetos como botones, lista de selección, graficas de resultados, selección de carpetas etc. El GUI es inicializado con:

```
function gui = createGUI()
gui.rootWindow = figure('Name'                ,'Image Editor'
,...
'MenuBar'           ,'none'         ,...
'ToolBar'           ,'none'         ,...
'NumberTitle'       ,'off'          ,...
'Position'          ,[100, 100, 1800, 900])
```

Donde se especifican los atributos generales deseados para la interfaz como los menús que traen incluidos por defecto y su posición. Una vez creada se empiezan a organizar paneles los cual luego contendrán los objetos primarios como botones, paneles, etc. Los paneles se arreglan ya sea de forma vertical o de forma horizontal dependiendo del diseño.

```
gui.mainvb = uiextras.HBoxFlex('Parent'     ,gui.rootWindow ,...
'Spacing'    ,3);

gui.panelFile = uiextras.BoxPanel('Parent'  ,gui.mainvb     ,...
'Title'             ,'Archivos'     ,...
'MinimizeFcn'       ,@Minimize);
gui.vb1 = uiextras.VBoxFlex('Parent'        ,gui.panelFile  ,...
'Spacing'           ,4);
```

Los paneles se establecen como padres (parent) y se le establece a que padre pertenece. Igual los objetos asignados serán solo pertenecientes a tal panel. De tal forma, la creación grafica se puede ver de forma jerárquica. Al tener los paneles se implementan los objetos con los cual el usuario tendrá interacción como los botones y listas.

```
gui.regionBtn = uicontrol('Parent'          ,gui.ImgSelPanel,...
                          'String'          ,'Región'       ,...
                          'Callback'        ,@regionImg);
```

Al igual que los paneles, los botones también se les establece un padre, al cual pertenecerá, su nombre y a la función a la cual se redirigirá como en el ejemplo es "`@regionImg`". Con esto se forma el backbone del programa el cual contendrá todas las funciones y espacios visuales para llevar a cabo la caracterización de imágenes.

El formato GUI se dividió en 6 secciones visuales en donde cada sección lleva a cabo su función específica. En la figura 14, se puede apreciar la representación gráfica dividida en 6 zonas, que se describen a continuación:

Figura 14: Divisiones dentro del programa GUI

1. Archivos
    a. Su función es presentar las carpetas disponibles en el directorio del usuario.
    b. Contiene la función de poder subir un nivel de directorio (representado por "..."). La idea es que pueda detectar los archivos e imágenes desde el dispositivo en uso y de tal forma incluirlas dentro de la pantalla del GUI para su visualización.
    c. Componentes utilizados:
        i. Paneles

2. Archivos
    a. Esta sección se encarga de enlistar cualquier contenido (de imágenes) del directorio actual, escogido en la sección 1.
    b. Componentes utilizados:
        i. Paneles

3. Imagen
    a. Su función es de exhibir imágenes seleccionadas de la zona 2.
    b. Al igual también exhibe imágenes que pasan por el proceso de:
        i. Ecualización
        ii. Caracterización
        iii. Comparación de ya sea i, ii con la imagen original.

c. Componentes utilizados:
    i. Paneles
    ii. Axes

4. Ver
    a. Esta zona tiene las funciones de poder realizar operaciones de zoom + o – en una imagen.
    b. Se pueden elegir varias opciones de la imagen elegida para caracterizar. Cuando el proceso es completo se pueden elegir todas las siguientes funciones las cual se ilustrarán en la zona 3:
        i. Original
        ii. Ecualización
        iii. Caracterización
        iv. Comparación
    c. Componentes utilizados:
        i. Paneles
        ii. List Box
        iii. Botones

5. Tools
    a. Esta zona tiene múltiples funciones:
        i. La habilidad de poder escoger/borrar una nueva región de una imagen ya elegida.
        ii. La función de ecualizar la imagen.
        iii. La función de poder escoger las distintas características (poro/sólido) dentro de la región deseada.
        iv. La función de entrenar el software con la información elegida.
        v. La función de poder caracterizar la imagen con lo que se utilizó para entrenar al software.
    b. Componentes utilizados:
        i. Paneles
        ii. List Box
        iii. Check Box
        iv. Botones

6. Datos
    a. Esta zona se encarga de ilustrar los datos obtenidos de la caracterización y representarlas en dos gráficas, una para sólidos y otra para vacíos:
        i. Cantidad de extremos iguales de la imagen

ii. Cantidad de líneas completas dentro de la imagen de un solo valor
iii. Círculos completos dentro de la imagen

b. Componentes utilizados:
   i. Paneles
   ii. Axes

Ejemplo

Ejemplo de función del sistema propuesto basado en SVM para la binarización:

A continuación, se puede apreciar la caracterización de una imagen mediante el Image Editor.

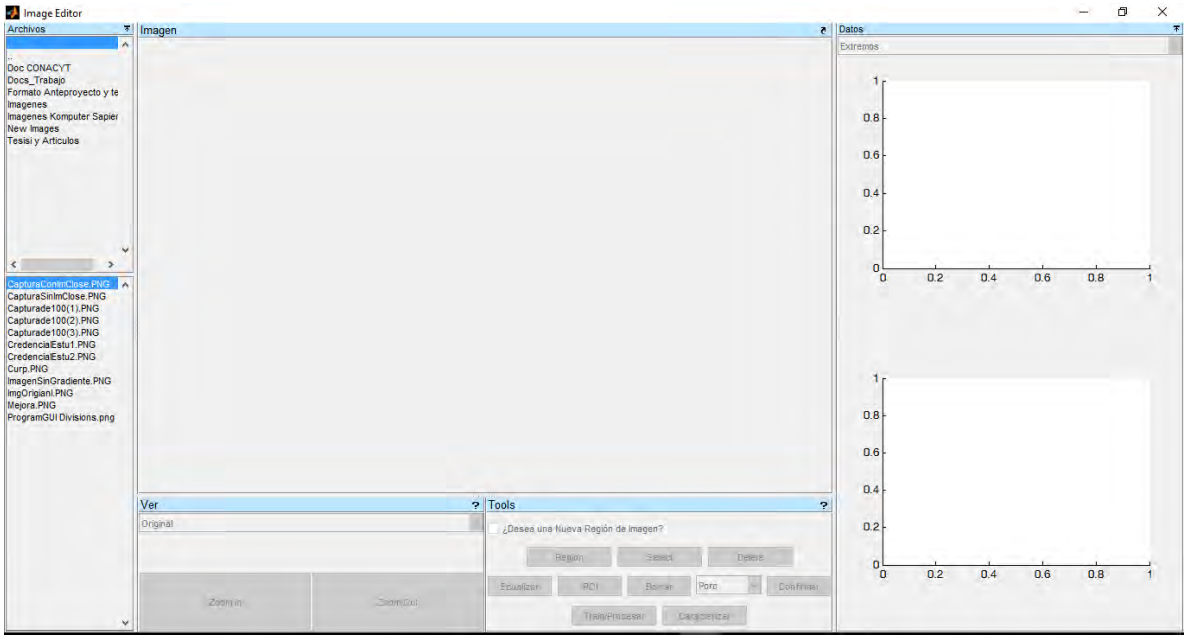


Figura 15: Ilustración Final del GUI

1. En la figura 15, se presenta la pantalla por defecto que aparece cuando se ejecuta el programa.
2. A la izquierda se puede apreciar en “Archivos”, el listado de carpetas en el directorio en el cual se encuentra la aplicación. La segunda fila debajo de “Archivos” nos muestra el contenido de la carpeta seleccionada al momento.
3. Del listado de contenido, utilizaremos la imagen “ImgOrigianl.PNG” para este ejercicio, tal como se muestra en la figura 16.

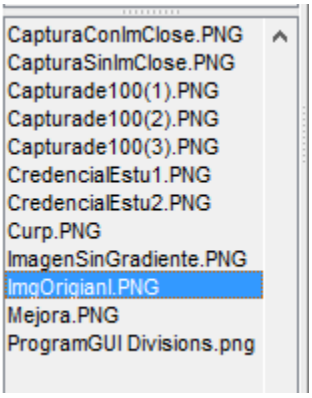


Figura 16: Zona 1 de Listado de archivos

4. Al seleccionar la imagen, automáticamente podemos visualizarla dentro del panel “Imagen”, esto se puede ver en la figura 17.

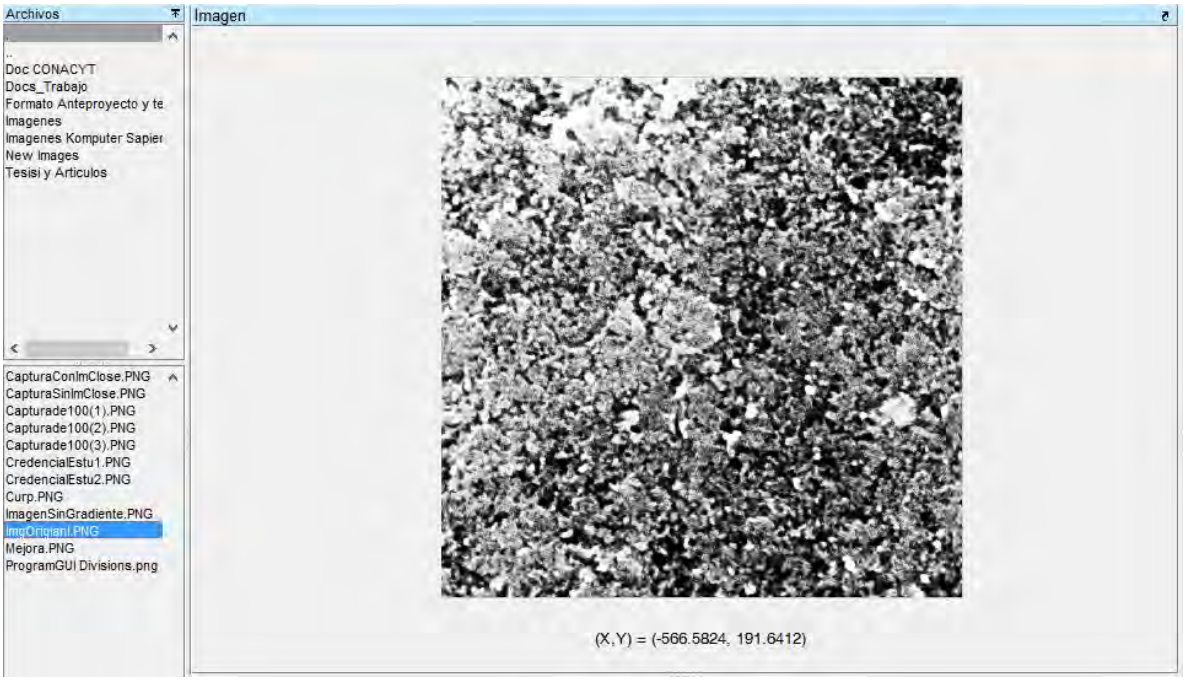


Figura 17: Zona 3 el cual ilustra la imagen seleccionada

5. Con la imagen ya cargada, podemos ahora utilizar el panel de “Ver”. Bajo este panel tendremos a disponibilidad la opción de poder realizar zooms (+ ó –) en la imagen. También tenemos disponible un listbox que nos permitirá escoger entre las diferentes opciones de la imagen, como se muestra en la figura 18.

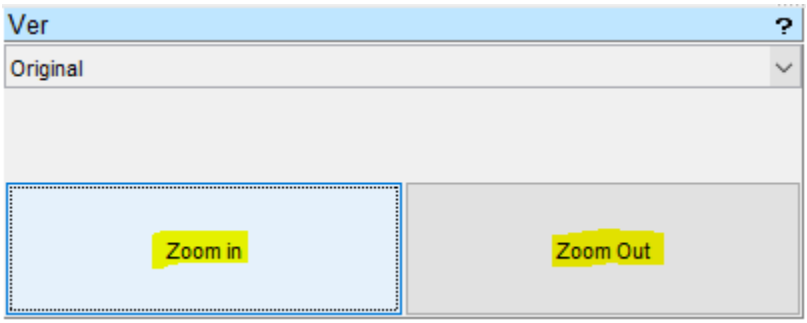


Figura 18: Zona con opciones de selección y zoom

Como apenas estamos iniciando el proceso, entonces solo tenemos disponible para ver la imagen Original. Ya que se realicen los otros procesos, podremos seleccionar y ver estas otras imágenes mediante el listbox.

Seleccionando la opción de “Zoom in” nos permite realizar acercamientos de la imagen. Cuando se desee retroceder el “Zoom in” entonces seleccionamos la opción de “Zoom Out”. En la figura 19, se puede apreciar diferentes zoom.

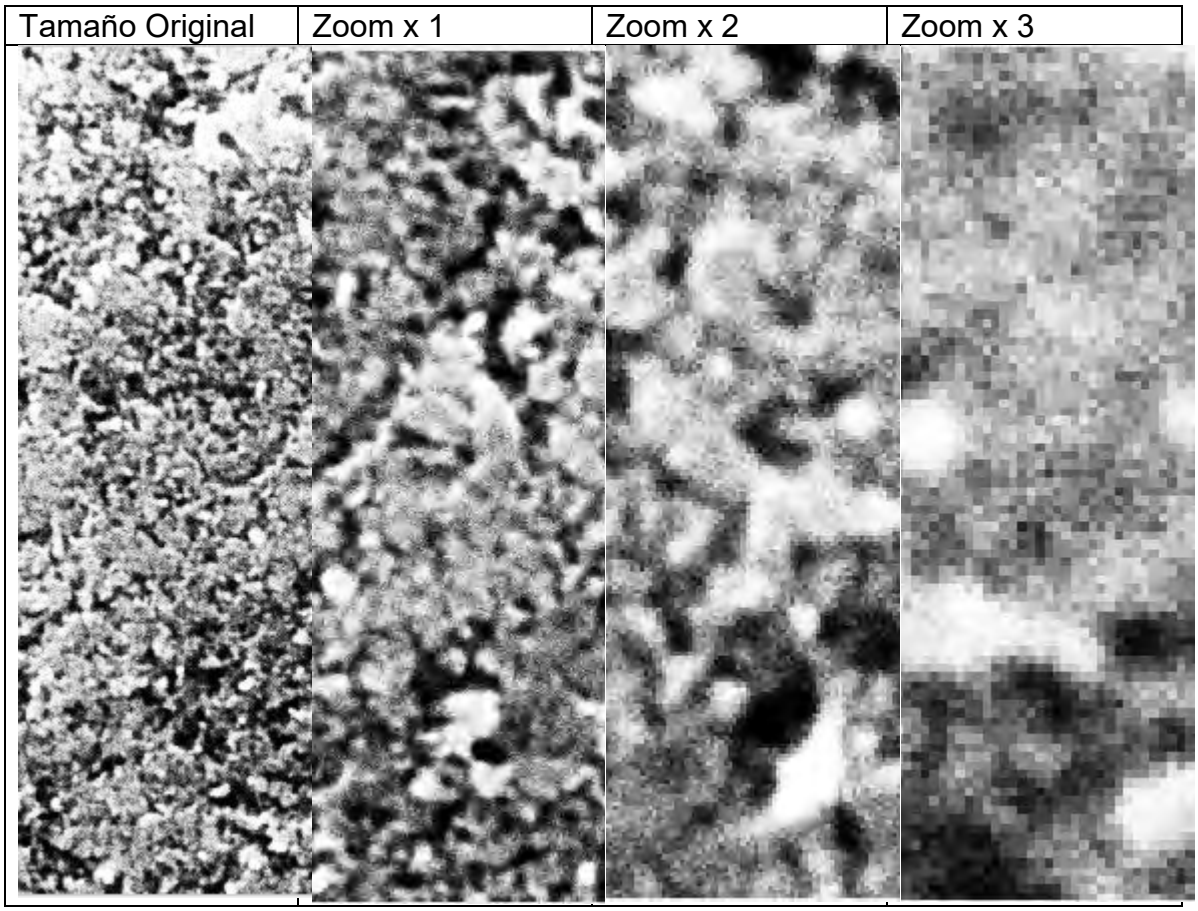


Figura 19: Ejemplo de Zoom en una imagen

Para obtener la figura 19, se selecciona el botón de “Zoom in” e instantáneamente se puede ver que el botón cambia de color el cual nos muestra que la opción esta seleccionada.

Luego con el cursor realizamos clics en la imagen de donde queremos llevar a cabo esta acción (+):

Para poder realizar la acción inversa realizamos clic en el botón de “Zoom Out”. Cada clic realizara un (-) a la imagen. Ay que tomar en cuenta que el botón de “Zoom in” sigue seleccionado y de tal forma ay que realizar un clic para desactivarlo.

6. Para dar inicio al proceso de caracterización, podemos utilizar la imagen original completa o podemos seleccionar una sección de la imagen. En este caso estamos utilizando una región específica. Para esto nos dirigimos al panel de “Tools” y seleccionamos el único checkbox disponible, como se muestra en la figura 20.

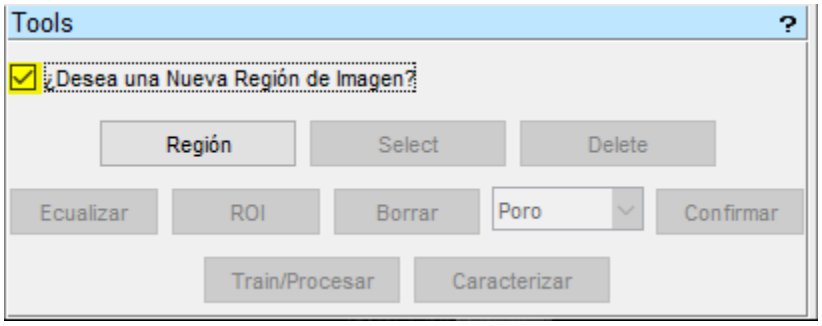


Figura 20: Imagen ilustrando el checkbox para selección de región

Luego seleccionamos el botón “Región” para poder iniciar con la selección de la región requerida. Con el cursor seleccionamos el punto inicial sobre la imagen, realizamos un clic sin soltarlo y arrastramos hasta un punto final requerido. Esto nos mostrará una caja con la región que escogimos como se puede ver en la figura 21.

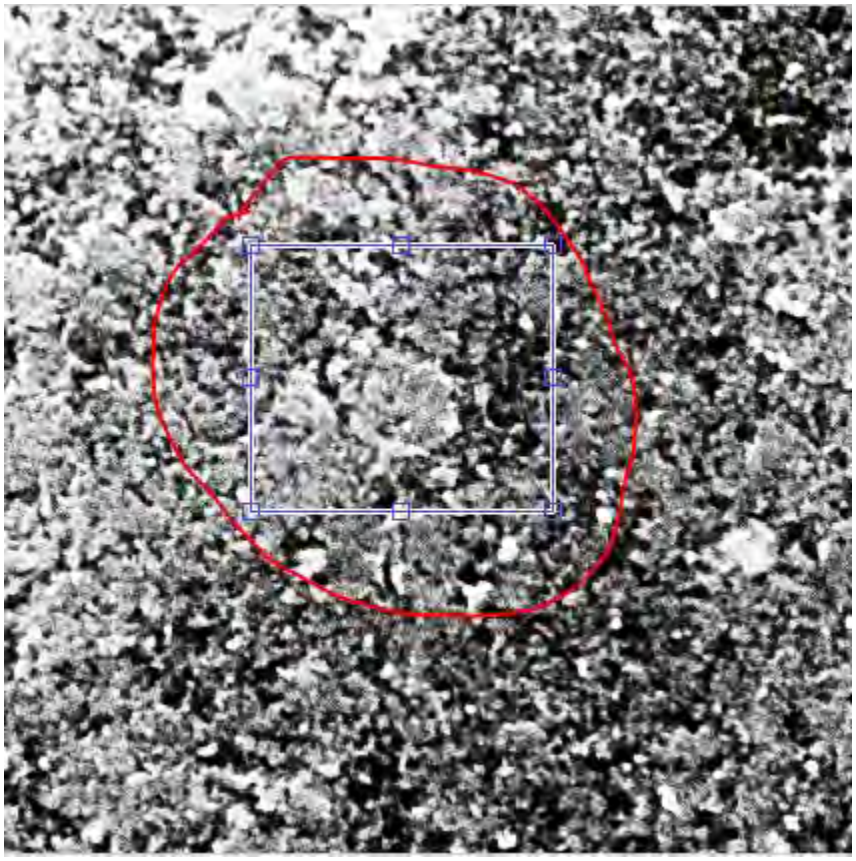


Figura 21: Imagen ilustrando la selección del ROI

Si se cometió algún error entonces se puede seleccionar el botón “Delete” para borrar la selección recién hecha. Si la selección de la región es la deseada entonces se selecciona el botón “Select” para confirmar y guardar nuestra selección. Estas opciones se muestran en la figura 22.

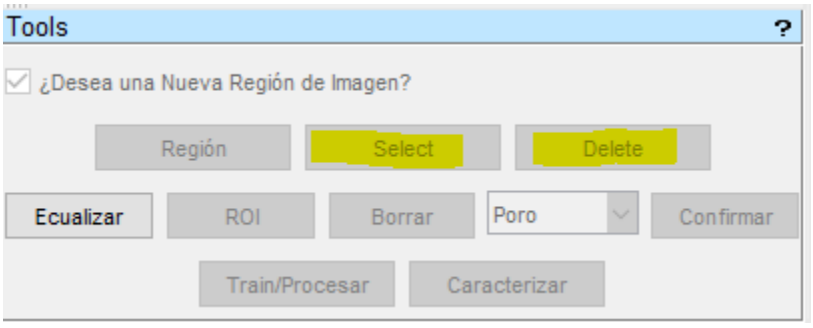


Figura 22: Imagen ilustrando los botones de Select y Delete

Una vez confirmada la selección, la imagen se exhibirá en “Image” con el nuevo tamaño, tal como se muestra en la figura 23.

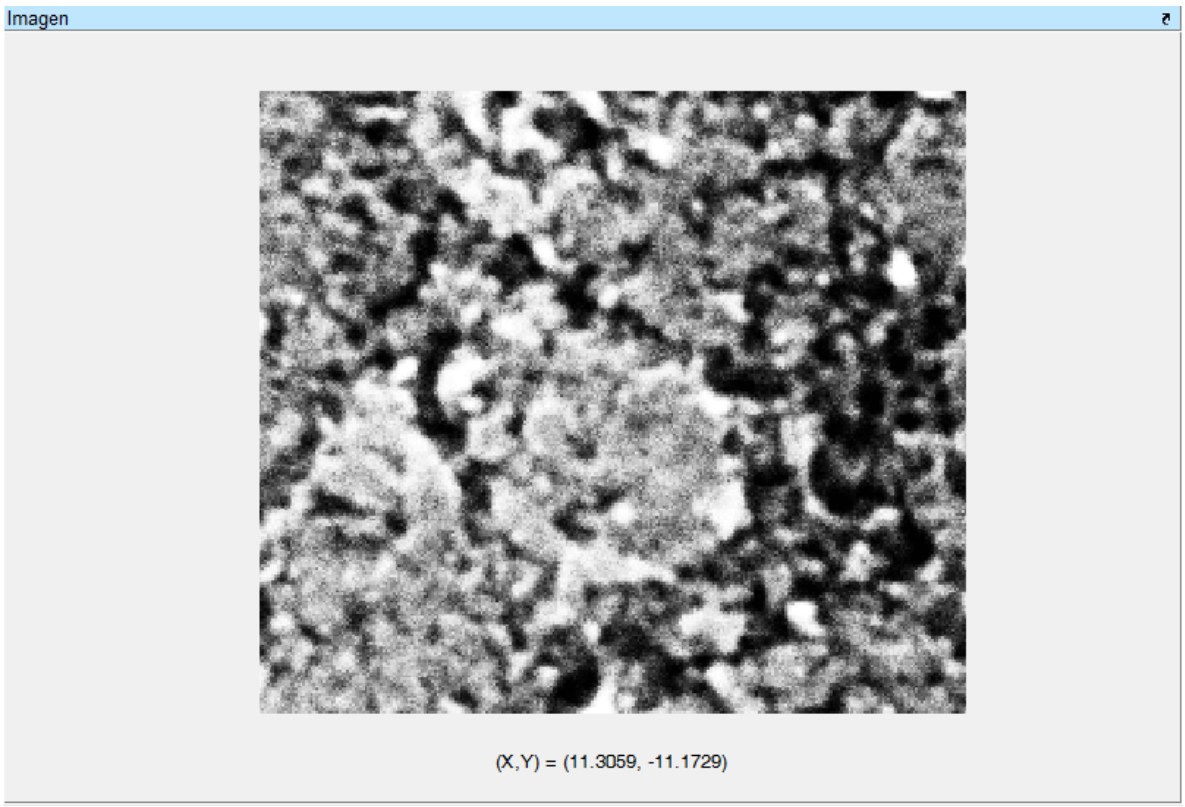


Figura 23: Imagen ilustrando el recorte a la imagen mediante ROI

7. Ya teniendo nuestra imagen, podemos seguir con el proceso, en donde ahora, Ecualizamos la imagen, con el botón subrayado en amarillo de la figura 24. Se selecciona la imagen y podemos ver en la figura 25 el cambio en el ajuste de la intensidad del color.

Figura 24: Imagen ilustrando el botón de Ecualizar

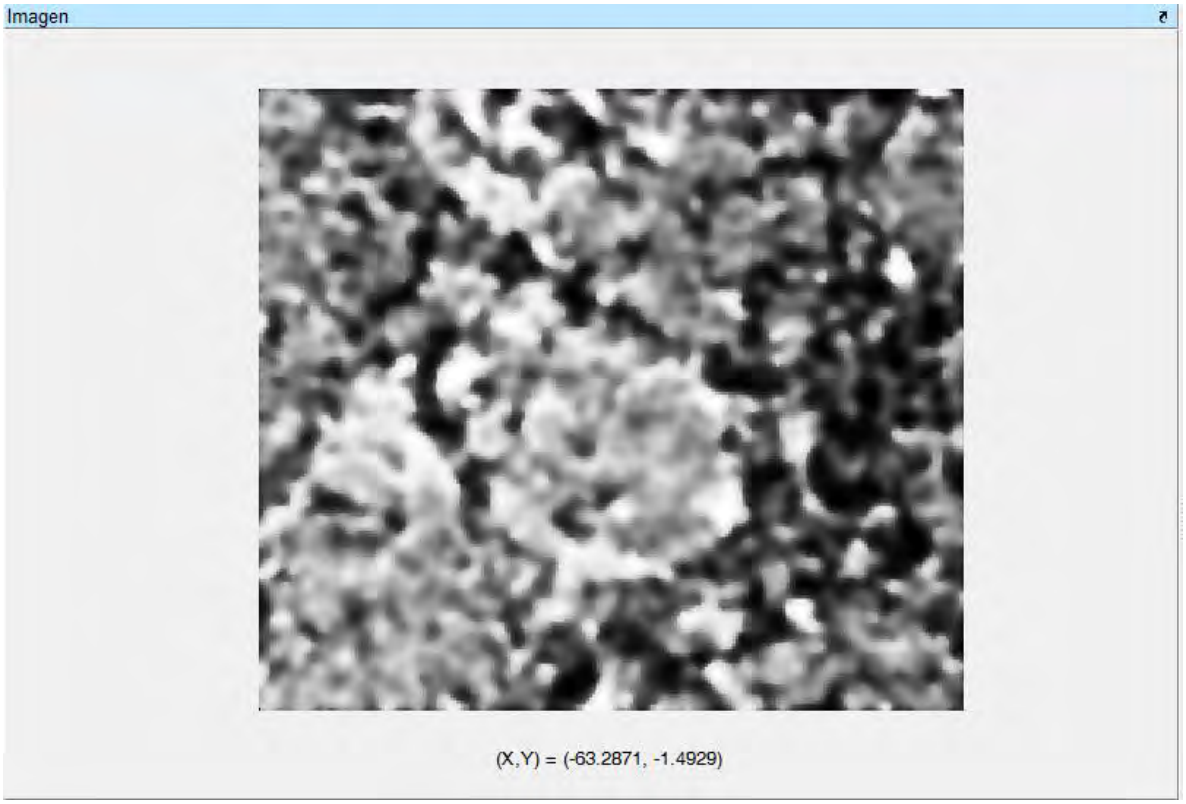


Figura 25: Imagen ilustrando imagen ecualizada

8. Con la imagen ya ecualizada, podemos seleccionar las regiones “poros” y “sólidos”. Para esto se selecciona el botón “ROI” y con el cursor seleccionamos la región requerida. Una vez seleccionada escogemos si tal región es un poro o un sólido y damos Confirmar para enviar la información, esto se hace con los controles subrayados en la figura 26.

Figura 26: Imagen ilustrando los botones de selección de estado sólido/poro

Al final tendremos un resultado como el mostrado en la figura 27, donde podemos apreciar cajas azules que fueron seleccionadas como sólidos y cajas rojas para representar poros.

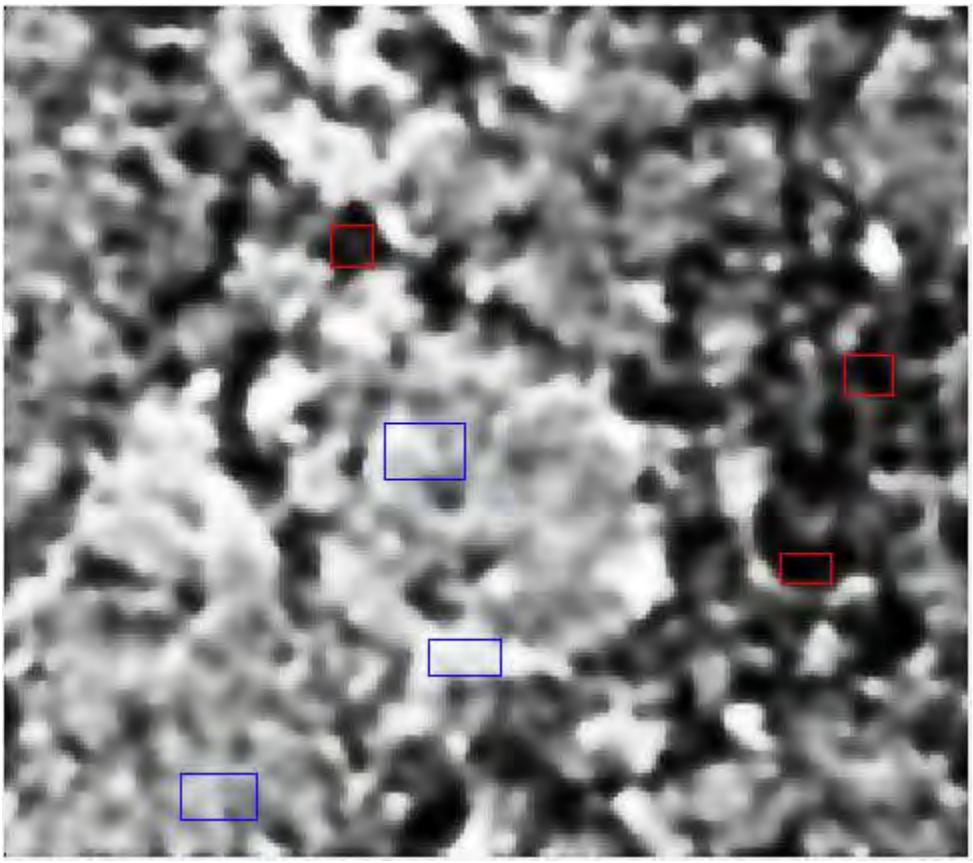


Figura 27: Imagen ilustrando selección de poros y sólidos en imagen

9. Una vez teniendo nuestras regiones seleccionadas podemos proceder con el paso de entrenar el SVM. Iniciamos este proceso con el botón “Train/Procesar”, mostrado en a figura 28.

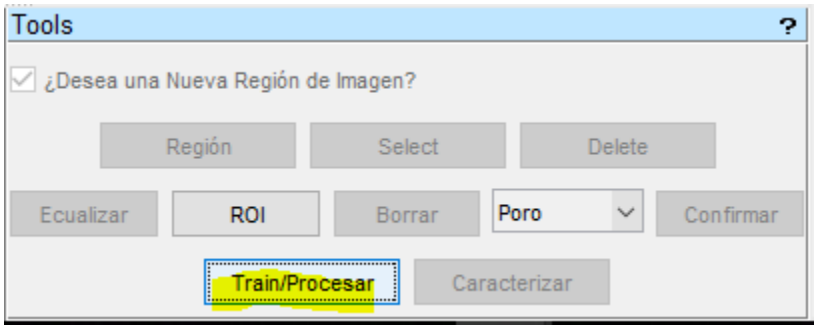


Figura 28: Imagen ilustrando el botón de train/procesar

Esto tomará toda la información seleccionada desde el paso 6 y se entrenará para poder producir la versión binaria de la imagen original (o región seleccionada). Con esto obtendremos la imagen presentada en la figura 29.

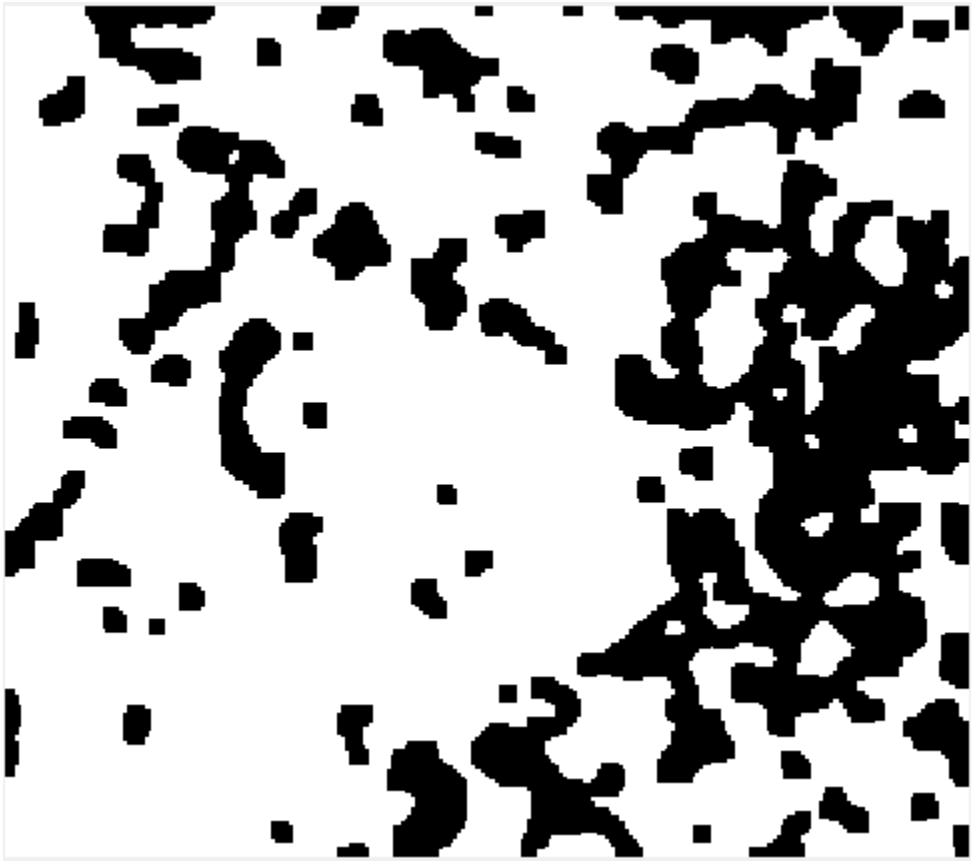


Figura 29: Imagen resultado binarizada

El resultado dependerá de las selecciones creadas de poros/sólidos. Se pueden escoger cualquier cantidad de regiones para poder detallar aún más el entrenamiento.

10. Una vez obtenida nuestra imagen binarizada, podemos caracterizarla. Realizando un clic en el botón “Caracterizar” nos permite a realizar esta función, tal como se presenta en la figura 30.

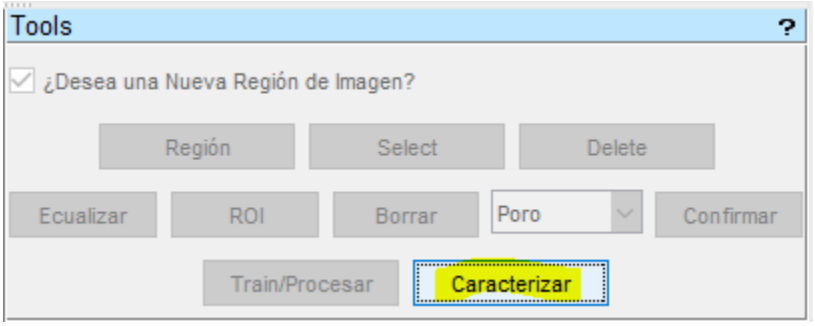


Figura 30: Imagen ilustrando el botón caracterizar

11. La caracterización ahora nos permite utilizar los paneles de Datos y la misma vez podemos ver gráficas con los resultados de la caracterización, figura 31

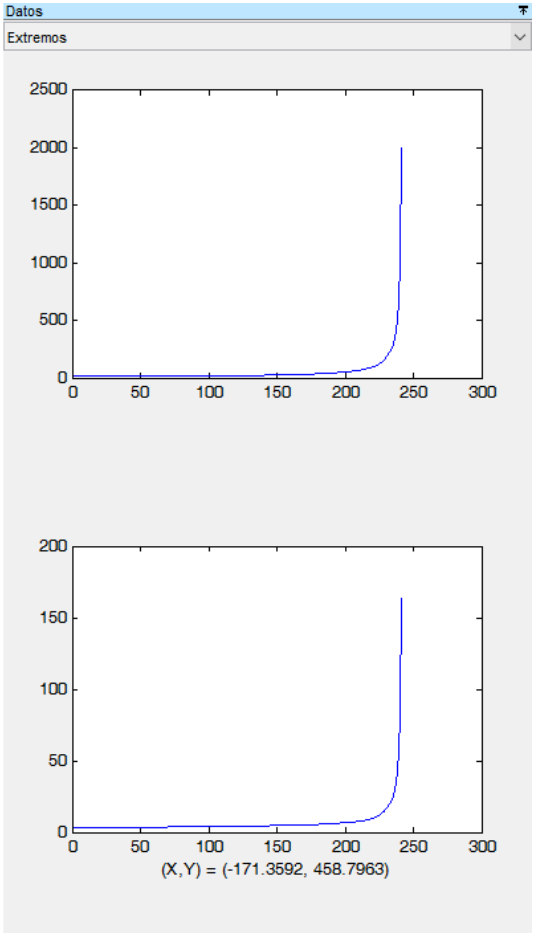


Figura 31: Zona ilustrando resultado de graficas tipo caracterización

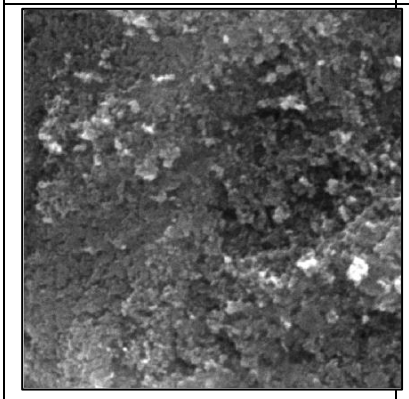
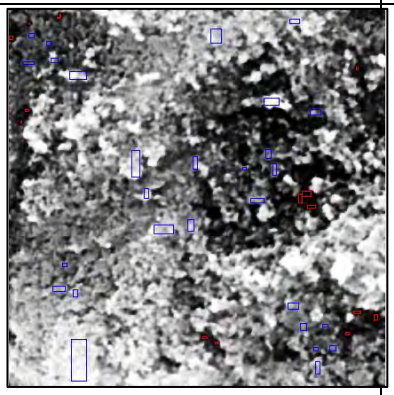
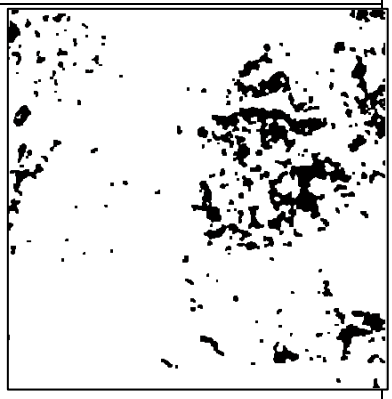
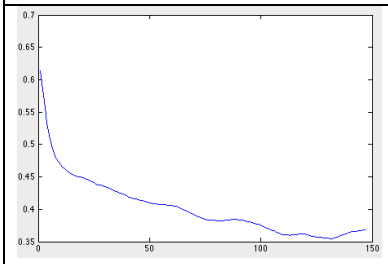
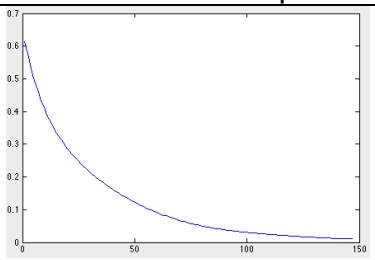
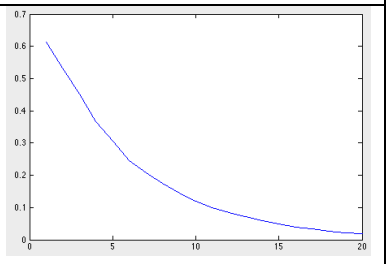
CAPÍTULO 4

CAPÍTULO 4

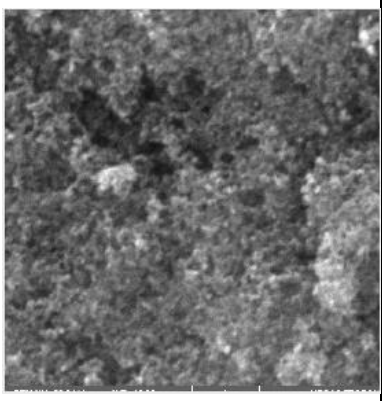
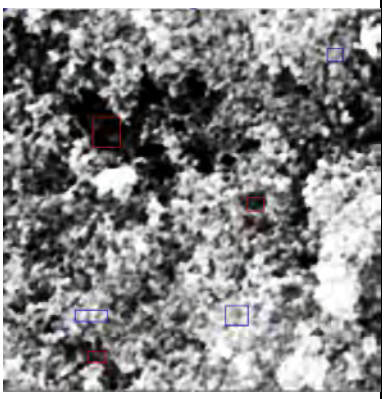
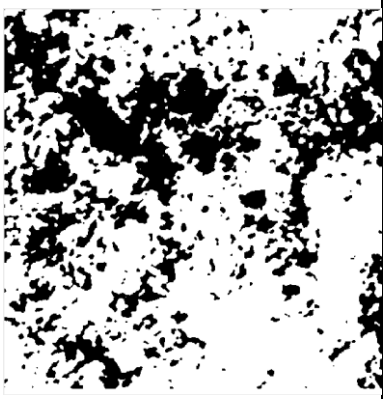
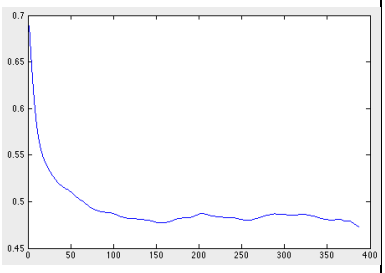
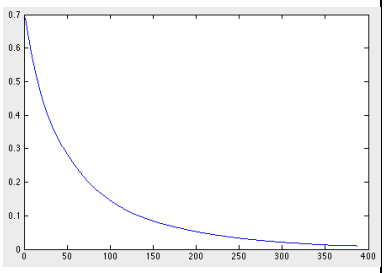
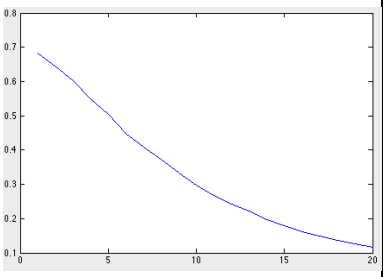
RESULTADOS EXPERIMENTALES

La fase de pruebas se considera que se realizó con éxito. Existen varias herramientas que nos brindan el procesamiento de imágenes, pero no existe alguna imagen a la cual se le pueda decir que es exactamente y precisamente procesada, y de tal forma la aceptación de los resultados dependerá del usuario y su satisfacción.

A continuación, se puede apreciar la imagen original con los resultados obtenidos mediante la herramienta al lado de resultados de caracterización.

Resultado 1		
Imagen Original	Regiones Seleccionadas	Binarizada
		
Gráfica extremos	Gráfica líneas completas	Gráfica de círculos
		
La gráfica nos demuestra que entre mas nos acercamos al punto maximo del eje x, existen menos ocurencias de estados solidos extremos	La gráfica nos demuestra que entre mas nos acercamos al punto maximo del eje x, existen menos ocurencias de lineas solidas completas	La gráfica nos demuestra que entre mas nos acercamos al punto maximo del eje x, existen menos ocurencias de círculos, los cuales

lo cual nos indica la normalización de esta.	lo cual nos indica la normalización de esta.	cubren una area solida entera.

Resultado 2		
Imagen Original	Regiones Seleccionadas	Binarizada
		
Grafica Extremos	Grafica Líneas Completas	Círculos
		
La gráfica nos demuestra que entre mas nos acercamos al punto maximo del eje x, existen menos ocurencias de estados solidos extremos	La gráfica nos demuestra que entre mas nos acercamos al punto maximo del eje x, existen menos ocurencias de lineas solidas completas	La gráfica nos demuestra que entre mas nos acercamos al punto maximo del eje x, existen menos ocurencias de círculos, los cuales

lo cual nos indica la normalización de esta.	lo cual nos indica la normalización de esta.	cubren una area solida entera.

CAPÍTULO 5

CAPÍTULO 5

CONCLUSIONES

El estudio de celdas de combustible empieza a ser un tema de mucho interés, el cual involucra a muchos campos de estudio en el avance de soluciones y propuestas tecnológicas. De tal forma este proyecto nos proporcionó el aporte del campo de informática en las áreas de programación e inteligencia artificial. Utilizando programación, se creó el editor de imágenes “The Image Editor”, el cual nos proporciona una técnica automatizada de aprendizaje para poder caracterizar los valores de pixeles dentro de las imágenes SEM.

La caracterización nos ayuda en el estudio a detalle la efectividad del transporte de energía dentro de las estructuras de las celdas. Esto lo lleva a cabo mediante un número de métodos como: Ecualización, binarización, normalización, aprendizaje automatizado y caracterización.

Cabe mencionar, una parte de los resultados fueron publicados en la revista Komputer Sapiens en 2016 con el título: Aprendizaje SVM en Tecnología Sustentable. La revista Komputer Sapiens, la cual se encuentra indizada en el CONACYT. La figura 32, presenta la primera hoja del artículo.


www.komputersapiens.org

KOMPUTER SAPIENS

Revista de Divulgación de la Sociedad Mexicana de Inteligencia Artificial

Año VIII, Vol. I. Enero - Mayo 2016 Artículo Komputer Sapiens 16 / 32

ARTÍCULO ACEPTADO

Caracterización de Electrodos de Celdas de Combustible Aplicando SVM

Jaime Ortegón Aguilar, José Torres Pozos, Romeli Barbosa Pool y Javier Vázquez Castillo

Figura 32: Publicación resultado de la tesis Komputer Sapiens Año VIII, Vol. I

En el artículo se presenta una aplicación de las máquinas de vectores de soporte o SVM para la clasificación de los píxeles de una imagen del electrodo de una celda de combustible obtenida a través de SEM. Considerando que la celda está constituida por dos fases es posible determinar el coeficiente efectivo de transporte y realizar simulaciones sobre su comportamiento. Se puede apreciar la utilidad de una técnica de aprendizaje automático, las SVM, las cuales ayudan a clasificar los píxeles dentro de imágenes de tipo SEM. Esta técnica nos ayuda a determinar la efectividad electrónica y iónica conductiva de la capa catalítica dentro de celdas de combustible de hidrógeno. Sirve como antecedente, en un paso crucial para determinar la efectividad de transporte de energía dentro de la estructura del electrodo con la información estadística. Se aplican varios métodos, como ecualización, normalización, morfología, entrenamiento y aprendizaje automático y caracterización. Con un pre procesamiento es posible seleccionar las regiones que servirán para el entrenamiento de la SVM, ésta se entrena para aprender a clasificar la información de la imagen actual. A partir de los píxeles clasificados, se puede estudiar más a detalle las PEMFC, apoyando a que en un futuro cercano puedan ser la fuente de energía que aporte un porcentaje importante del consumo global, principalmente en sistemas portátiles.

Actualmente, la caracterización se realiza usando un solo factor de magnificación (acercamiento o zoom) del SEM. Como trabajo futuro, para una misma muestra, se debe procesar imágenes de diferentes factores de magnificación y que todos ellos sean caracterizados automáticamente.

REFERENCIAS BIBLIOGRÁFICAS

Barbosa R., Andaverde J., et al. (2011) “Stochastic reconstruction and a scaling method to determine effective transport coefficients of a proton exchange membrane fuel cell catalyst layer”. Journal of Power Sources. Vol. 196, No. 3, pp. 1248–1257.

Barbosa R., Escobar B., Cano U., et al. (2011). “Stochastic Reconstruction at Two Scales and Experimental Validation to Determine the Effective Electrical Resistivity of a PEMFC Catalyst Layer”. ECS Transactions. Vol. 41 No. 1, pp. 2061–2071.

Das P.K., Li X. y LiuJianjun Z.-S. (2010). “Effective transport coefficients in PEM fuel cell catalyst and gas diffusion layers: Beyond Bruggeman approximation”. Applied Energy. Vol 87, pp. 2785–2796.

Erdinc O. y Uzunoglu M. (2010). “Recent trends in PEM fuel cell-powered hybrid systems: Investigation of application areas, design architectures and energy management approaches”. Renewable and Sustainable Energy Reviews. Vol. 14, No. 9, pp. 2874–2884.

Gonzalez R y Woods R.(2008). “Digital image processing”. 3a. edición. Pearson/Prentice Hall. ISBN 978-0-13-168728-8. 6. Guarino V., Guaccio A., et al. (2010). “Image processing and fractal box counting: user-assisted method for multi-scale porous scaffold characterization”. Journal of Materials Science: Materials in Medicine. Vol. 21, No. 12. pp. 3109–3118.

Jianjun L., Lijun L. y Youjun J. (2011). “Using Rock SEMImage to Create Pore-scale Finite Element Calculation Mesh”. Physics Procedia. Vol. 22, pp. 227–232.

Mathworks (2015). “MATLAB”. Recuperado el 18 de octubre de 2015 de http://www.mathworks.com/products/matlab/

Wang L. (2005) “Support vector machines: theory and applications”. Springer. ISBN 978-3-540-24388-5.

ANEXO A

```
% Procesamiento de imágenes para la caracterización de celdas de combustible (c)
% Universidad de Quintana Roo
% Jose A. Torres Pozos

function ImageProcessor()

% GUI Structures
gui = struct();
data = struct();
% Launch the GUI
data = createData();
gui = createGUI();

    function data = createData()
%
        data.LocalMainDir = pwd;
        data.himage = 0;

    end

    function gui = createGUI()
        gui.rootWindow = figure('Name'              ,'Image Editor' ,...
                                'MenuBar'           ,'none'         ,...
                                'ToolBar'           ,'none'         ,...
                                'NumberTitle'       ,'off'          ,...
                                'Position'          ,[100, 100, 1800, 900]);

        gui.mainvb = uiextras.HBoxFlex('Parent'     ,gui.rootWindow ,...
                                'Spacing'    ,3);

        gui.panelFile = uiextras.BoxPanel('Parent'  ,gui.mainvb     ,...
                                'Title'             ,'Archivos'     ,...
                                'MinimizeFcn'       ,@Minimize);
        gui.vb1 = uiextras.VBoxFlex('Parent'        ,gui.panelFile  ,...
                                'Spacing'           ,4);
        gui.vb2 = uiextras.VBoxFlex('Parent'        ,gui.mainvb     ,...
                                'Spacing'           ,4);
        gui.trainPanel = uiextras.BoxPanel('Parent' ,gui.mainvb     ,...
                                'Title'             ,'Datos'        ,...
                                'MinimizeFcn'       ,@MinimizeA);
        gui.vb3 = uiextras.VBoxFlex('Parent'        ,gui.trainPanel ,...
                                'Spacing'           ,4);

        gui.files = dir('.');
        gui.listFolders = uicontrol('Style'         ,'ListBox'      ,...
                                'Parent'            ,gui.vb1        ,...
                                'String'
,{gui.files([gui.files.isdir]).name},...
                                'Callback'          ,@onClickFolder);

        data.images =           [dir('*.jpg');       dir('*.png')   ;...
                                dir('*.tif')];
        data.listImages = uicontrol('Style'         ,'ListBox'      ,...
```

```
                                'Parent'            ,gui.vb1        ,...
                                'String'            ,{data.images.name},...
                                'Callback'          ,@onClickFiles);

        gui.panelImg = uiextras.BoxPanel('Parent'   ,gui.vb2        ,...
                                'Title'             ,'Imagen'       ,...
                                'FontSize'          ,10             ,...
                                'DockFcn'           ,@separate      ,...
                                'IsDocked'          ,true);
        gui.axesFigure = axes('Parent'              ,gui.panelImg   ,...
                                'Visible'           ,'off'          ,...
                                'ActivePositionProperty','OuterPosition');

        gui.toolspanel   = uiextras.HBox('Parent'   ,gui.vb2);
        gui.processpanel = uiextras.VBox('Parent'   ,gui.vb3);


        gui.caracPanel   = uiextras.HBox('Parent'   ,gui.processpanel);
        gui.caracPopup   = uicontrol('Parent'       ,gui.caracPanel ,...
                                'Style'             ,'popup'        ,...
                                'String',{'Extremos','Lineas
Completas','Circulos'},...
                                'Callback'          ,@VerGrafica);


        gui.axesFigure1 = axes('Parent'             ,gui.processpanel,...
                                'Visible'           ,'on'           ,...
                                'ActivePositionProperty','OuterPosition');
        gui.axesFigure2 = axes('Parent'             ,gui.processpanel,...
                                'Visible'           ,'on'           ,...
                                'ActivePositionProperty','OuterPosition');


     %*****************Paneles y Botones de Panel Ver

        gui.panelVer = uiextras.BoxPanel('Parent'   ,gui.toolspanel ,...
                                'Title'             ,'Ver'          ,...
                                'FontSize'          ,10             ,...
                                'HelpFcn'           ,@Help);
        gui.vb4 = uiextras.VBox('Parent'            ,gui.panelVer);
        gui.popup =   uicontrol('Parent'            ,gui.vb4        ,...
                                'Style'             ,'popup'        ,...
                                'String'
,{'Original','Ecualizada','Binaria','Comparación'},...
                                'Callback'          ,@SelectImg);
        gui.zoomBtnPanel = uiextras.HBox('Parent'   ,gui.vb4);

        gui.zoomButton = uicontrol('Style'          ,'togglebutton' ,...
                                'String'            ,'Zoom in'      ,...
                                'Parent'            ,gui.zoomBtnPanel,...
                                'Callback'          ,@Zoom);
        gui.zoomOutButton = uicontrol('Style'       ,'togglebutton' ,...
                                'String'            ,'Zoom Out'     ,...
                                'Parent'            ,gui.zoomBtnPanel,...
                                'Callback'          ,@ZoomOut);


    %*****************Paneles, Botones de Panel Tools

        gui.panelButton = uiextras.BoxPanel('Parent',gui.toolspanel ,...
                                'Title'             ,'Tools'        ,...
                                'FontSize'          ,10             ,...
                                'HelpFcn'           ,@Help);
```

```
        gui.vb5 = uiextras.VBox('Parent'            ,gui.panelButton);
        gui.checkBoxPanel = uiextras.HBox('Parent'      ,gui.vb5);
        gui.ImgSelPanel = uiextras.HButtonBox('Parent'  ,gui.vb5);
        gui.EstSelPanel = uiextras.HButtonBox('Parent'  ,gui.vb5);
        gui.procesarPanel = uiextras.HButtonBox('Parent',gui.vb5);

        gui.regionChkBox = uicontrol('Parent'       ,gui.checkBoxPanel,...
                                'String'            ,'¿Desea una Nueva Región de
Imagen?',...
                                'Style'             ,'checkbox'     ,...
                                'Callback'          ,@regionCheck);
        gui.regionBtn = uicontrol('Parent'          ,gui.ImgSelPanel,...
                                'String'            ,'Región'       ,...
                                'Callback'          ,@regionImg);
        gui.SelImgBtn = uicontrol('Parent'          ,gui.ImgSelPanel,...
                                'String'            ,'Select'       ,...
                                'Callback'          ,@SelectImrectPos);
        gui.DelregionBtn = uicontrol('Parent'       ,gui.ImgSelPanel,...
                                'String'            ,'Delete'       ,...
                                'Callback'          ,@Delregion);

        gui.EcualizarBtn = uicontrol('Parent'       ,gui.EstSelPanel,...
                                'String'            ,'Ecualizar'    ,...
                                'Callback'          ,@Ecualizar);
        gui.ROIBtn = uicontrol('String'             ,'ROI'          ,...
                                'Parent'            ,gui.EstSelPanel,...
                                'Callback'          ,@onROIClick);
        gui.BorrarBtn = uicontrol('String'          ,'Borrar'       ,...
                                'Parent'            ,gui.EstSelPanel,...
                                'Callback'          ,@Delete2);
        gui.EstadoPopPanel = uicontrol('Parent'     ,gui.EstSelPanel,...
                                'Style'             ,'popup'        ,...
                                'String'            ,{'Poro','Solido'});
        gui.ConfirmBtn = uicontrol('Parent'         ,gui.EstSelPanel,...
                                'String'            ,'Confirmar'    ,...
                                'Callback'          ,@Confirm);


    %****************Botones de Procesar y caracterizar

        gui.ProcessBtn = uicontrol('Parent'         ,gui.procesarPanel,...
                                'String'            ,'Train/Procesar',...
                                'Callback'          ,@Train);
        gui.CaractBtn = uicontrol('Parent'          ,gui.procesarPanel,...
                                'String'            ,'Caracterizar' ,...
                                'Callback'          ,@Caracterizar);



    %*************Estableciendo el estado inicial de los botones y popups
        set(gui.popup         ,'Enable'       ,'off');
        set(gui.zoomButton    ,'Enable'       ,'off');
        set(gui.zoomOutButton ,'Enable'       ,'off');

        set(gui.regionChkBox  ,'Enable'       ,'off');

        set(gui.regionBtn     ,'Enable'       ,'off');
        set(gui.SelImgBtn     ,'Enable'       ,'off');
        set(gui.DelregionBtn  ,'Enable'       ,'off');

        set(gui.EcualizarBtn  ,'Enable'       ,'off');
        set(gui.ROIBtn        ,'Enable'       ,'off');
        set(gui.EstadoPopPanel,'Enable'       ,'off');
```

```
        set(gui.ConfirmBtn    ,'Enable'       ,'off');
        set(gui.BorrarBtn     ,'Enable'       ,'off');

        set(gui.ProcessBtn    ,'Enable'       ,'off');
        set(gui.CaractBtn     ,'Enable'       ,'off');
        set(gui.caracPopup    ,'Enable'       ,'off');


    %*************Estableciendo los tamaños dentro del GUI
        gui.vb1.Sizes       =   [-2 -3];
        gui.vb2.Sizes       =   [-7 -2];
        gui.mainvb.Sizes    =   [-1.5 -8 -4];
        gui.processpanel.Sizes = [1 -5 -5];
        gui.oldSizesM       =   gui.mainvb.Sizes;
        gui.oldSizesD       =   gui.mainvb.Sizes;

    end

    %*************Funciones
    function onClickFiles(hObj,eData)
        data.wait = waitbar(0,'Cargando...');
        index = get(hObj,'Value');
        if isempty(index) == 0
            waitbar(1/3);
            data.CurrentImageName = data.img_files(index).name;
            data.RGBImage = imread(data.CurrentImageName);
            waitbar(2/3);
            set(gui.axesFigure  ,'Visible'  ,'on');
            axes(gui.axesFigure);
            data.imgOriginal=data.RGBImage(1:end,1:end,1);

            waitbar(3/3);
            data.himage = imshow(data.RGBImage);

            set(gui.popup       ,'Enable'   ,'on');
            set(gui.zoomButton  ,'Enable'   ,'on');
            set(gui.zoomOutButton,'Enable'  ,'on');
            set(gui.regionChkBox,'Enable'   ,'on');
            set(gui.EcualizarBtn,'Enable'   ,'on');

            set(gui.popup       ,'value'    ,1);
            set(gcf,'WindowButtonMotionFcn' ,@mouseMove);
%             gcf
            data.clicked    = 0;
            data.equalized  = 0;
            data.binary     = 0;
            data.compare    = 0;
            data.PoroArray  = [];
            data.SolidoArray = [];
            data.GradM      = [];
            data.GradX      = [];
            data.GradMS     = [];
            data.GradXS     = [];
            close(data.wait);
        end

    end

    function mouseMove(object,eventdata)
        C = get(gca,'CurrentPoint');
%         gui.axesArea = get(gui.axesFigure,'Position')

%         if (C(1) < ...
```

```
%            (gui.axesArea(1) + gui.axesArea(3))) & ...
%            (C(1) > ...
%            (gui.axesArea(1))) & ...
%            (C(2) < ...
%                 gui.axesArea(2) + gui.axesArea(4)) & ...
%                 C(2)

             xlabel(gca, ['(X,Y) = (', num2str(C(1,1)), ', ',num2str(C(1,2)),
')'])
%         end
    end

    function onClickFolder(hObj,eData)
        index_selected = get(hObj,'Value');
        if strcmp(get(gui.rootWindow,'SelectionType'),'open')
        else
            folder_list = get(hObj,'String');
            %Item selected in list box
            folder_name = folder_list{index_selected};
            cd(folder_name);
            data.files = dir('.');
            set(gui.listFolders,'Value',1);
            set(gui.listFolders,'String',{data.files([data.files.isdir]).name});

            images = [dir('*.png'); dir('*.jpg'); dir('*.tif')];
            data.img_files = images;
            set(gui.listFolders,'Value',1);
            set(data.listImages,'String',{data.img_files.name});

         end
    end

    function separate(~,~)
         if gui.panelFile.IsDocked
             gui.oldSizesD = gui.mainvb.Sizes;
            gui.mainvb.Sizes(1) = 0;
            gui.mainvb.Sizes(3) = 0;
            gui.panelFile.Visible = 'off';
            gui.panelButton.Visible = 'off';
            gui.panelVer.Visible = 'off';
        else
            gui.mainvb.Sizes = gui.oldSizesD;
            gui.panelFile.Visible = 'on';
            gui.panelButton.Visible = 'on';
            gui.panelVer.Visible = 'on';
        end
        gui.panelFile.IsDocked = ~gui.panelFile.IsDocked;
    end

    function Help(~,~)
    end

    function Minimize(~,~)
        if gui.panelFile.IsMinimized
           gui.mainvb.Sizes = gui.oldSizesM;

        else
            gui.oldSizesM = gui.mainvb.Sizes;
            gui.mainvb.Sizes(1) = 30;

        end
        gui.panelFile.IsMinimized = ~gui.panelFile.IsMinimized;
    end
```

```
function MinimizeA(~,~)
        if gui.panelFile.IsMinimized
           gui.mainvb.Sizes = gui.oldSizesM;

        else
            gui.oldSizesM = gui.mainvb.Sizes;
            gui.mainvb.Sizes(3) = 30;

        end
        gui.panelFile.IsMinimized = ~gui.panelFile.IsMinimized;
    end

    function regionImg(~,~)
        data.rect = imrect;
        set(gui.SelImgBtn,'Enable','on');
        set(gui.DelregionBtn,'Enable','on');
%         (gca,[10 10 100 100])
    end

    function SelectImrectPos(~,~)
        data.newPos = getPosition(data.rect);
        data.Round = round(data.newPos);
        data.cols = data.Round(1) + data.Round(3);
        data.rows = data.Round(2) + data.Round(4);

        data.NewImg = data.RGBImage(data.Round(2):data.rows,...
            data.Round(1):data.cols);
        data.himage = imshow(data.NewImg);
        set(gui.DelregionBtn,'Enable','off');
        set(gui.SelImgBtn,'Enable','off');
        set(gui.EcualizarBtn,'Enable','on');
        set(gui.regionBtn,'Enable','off');
        set(gui.regionChkBox,'Enable','off');

    end

    function regionCheck(hObject, eventdata, handles)
        if (get(hObject,'Value') == get(hObject,'Max'))
            set(gui.regionBtn   ,'Enable','on');
            set(gui.EcualizarBtn,'Enable','off');
            set(gui.ROIBtn      ,'Enable','off');
            set(gui.EstadoPopPanel,'Enable','off');
            set(gui.ConfirmBtn  ,'Enable','off');
            set(gui.ProcessBtn  ,'Enable','off');
            set(gui.CaractBtn   ,'Enable','off');
            set(gui.ROIBtn      ,'Enable','off');
            set(gui.BorrarBtn   ,'Enable','off');
            data.clicked = 1;
        else
            set(gui.EcualizarBtn,'Enable','on');
            set(gui.regionBtn   ,'Enable','off');
            set(gui.SelImgBtn   ,'Enable','off');
            set(gui.DelregionBtn,'Enable','off');
            data.clicked = 0;
        end

    end

    function Ecualizar(~,~)
        if data.clicked == 1
            data.Thisimage = data.NewImg;
        else
```

```
        data.Thisimage = data.imgOriginal;
    end
    data.J=histeq(data.Thisimage);
    data.J = medfilt2(data.J,[3 3]);
    data.J=wiener2(data.J,[5 5]);

    data.l=double(data.J(:,:,1));
    data.i=(data.l-min(data.l(:)))./(max(data.l(:)-min(data.l(:))));

    data.himage = imshow(data.i);

    set(gui.popup,'value',2);
    set(gui.ROIBtn,'Enable','on');
    set(gui.EstadoPopPanel,'Enable','on');
    set(gui.EcualizarBtn,'Enable','off ');
    set(gui.regionChkBox,'Enable','off');
    data.equalized=1;
end

function Delregion(~,~)
    delete(data.rect);
end

function onROIClick(~,~)
    data.region = imrect;
    data.EstadoPos = getPosition(data.region);
    data.Round2 = round(data.EstadoPos);
    data.Ecols = data.Round2(1) + data.Round2(3);
    data.Erows = data.Round2(2) + data.Round2(4);

    data.ERegion = data.i(data.Round2(2):data.Erows,...
        data.Round2(1):data.Ecols);


    set(gui.ConfirmBtn,'Enable','on');
    set(gui.BorrarBtn,'Enable','on');
 end

function Delete2(~,~)
    delete(data.region);
    set(gui.BorrarBtn,'Enable','off');
end


function SelectImg(hObj,eventdata,handles)
    val = get(hObj,'Value');
    str = get(hObj,'String');

    switch str{val}
        case 'Original'
            axis 'auto';
            data.himage = imshow(data.Thisimage);

        case 'Ecualizada'
            axis 'auto';
            if data.equalized==0
                data.msg = msgbox('Falta Ecualizar','Aviso','warn');
                set(gui.popup,'value',1);
            else
                data.himage = imshow(data.i);
            end

        case 'Binaria'
```

```
            axis 'auto';
            if data.binary==0
                data.msg2 = msgbox('Falta Entrenar','Aviso','warn');
                set(gui.popup,'value',1);
            else
                data.himage = imshow(data.resultado);
            end

        case'Comparación'
            axis 'auto';
            if data.compare==0
                data.msg3 = msgbox('Falta Entrenar','Aviso','warn');
                set(gui.popup,'value',1);
            else
                data.himage = imshowpair(data.i,data.resultado,'montage');
            end
    end

end

function Confirm(~,~)
    val1 = get(gui.EstadoPopPanel,'Value');
    str1 = get(gui.EstadoPopPanel,'String');

    hold on

    [gx,gy]=imgradientxy(data.ERegion);
    [gm,gd]=imgradient(data.ERegion);

    delete(data.region);


    switch str1{val1}
        case 'Poro'
            data.PoroRegion = reshape(data.ERegion,[],1);
            data.PoroArray = [data.PoroArray; data.PoroRegion];

            data.GradMRegion = reshape(gm,[],1);
            data.GradM = [data.GradM; data.GradMRegion];

            data.GradXRegion = reshape(gx,[],1);
            data.GradX = [data.GradX; data.GradXRegion];

            rectangle('Position', [data.Round2(1),data.Round2(2),...
            data.Round2(3),data.Round2(4)],'EdgeColor','r');

        case 'Solido'
            data.SolidoRegion = reshape(data.ERegion,[],1);
            data.SolidoArray = [data.SolidoArray; data.SolidoRegion];

            data.GradMSRegion = reshape(gm,[],1);
            data.GradMS = [data.GradMS; data.GradMSRegion];

            data.GradXSRegion = reshape(gx,[],1);
            data.GradXS = [data.GradXS; data.GradXSRegion];

            rectangle('Position', [data.Round2(1),data.Round2(2),...
            data.Round2(3),data.Round2(4)],'EdgeColor','b');
    end



    set(gui.ProcessBtn,'Enable','on');
```

```
        set(gui.BorrarBtn,'Enable','off');
        set(gui.ConfirmBtn,'Enable','off');
    end

    function Train(~,~)

        data.trainarray = [data.PoroArray data.GradM data.GradX;...
            data.SolidoArray data.GradMS data.GradXS];

        data.amtZeros = size(data.PoroArray);
        data.zeros = zeros(data.amtZeros);

        data.amtOnes = size(data.SolidoArray);
        data.ones = ones(data.amtOnes);

        data.OnesZerosArray = [data.zeros; data.ones];


SVMStruct=svmtrain(double(data.trainarray(:,1)),double(data.OnesZerosArray(:,1)))
;

        [row,col] = size(data.i);
        data.imagenResh = reshape(data.i,[],1);
        data.imgDouble = double(data.imagenResh);
        data.out = svmclassify(SVMStruct,data.imgDouble);

        data.resultado = reshape(data.out,row,[]);
        se=strel('disk',3);
        close=imclose(data.resultado,se);
        data.himage = imshow(close);

% [centers, radii, metric] = imfindcircles(data.resultado,[15 30])

        set(gui.popup,'value',3);
        set(gui.CaractBtn,'Enable','on');
        set(gui.ROIBtn,'Enable','off');
        set(gui.EstadoPopPanel,'Enable','off');
        set(gui.ProcessBtn,'Enable','off');
        data.binary=1;
        data.compare=1;
    end

    function Caracterizar(~,~)
        wait = waitbar(0,'Cargando...');
        set(gui.caracPopup,'Enable','on');

        data.w = size(data.resultado,1);
        data.h = size(data.resultado,2);
        fileID = fopen('infile','w');
        fprintf(fileID,'%d ', data.w);
        fprintf(fileID,'%d ', data.h);
        fprintf(fileID,'%d ', data.resultado);
        fclose(fileID);
        waitbar(1/3);
        program = strcat(data.LocalMainDir, '\contador2.exe l infile outfile');
        system(program);
        if data.w > data.h
            largo = data.w
        else
            largo = data.h
        end
        fileID = fopen('outfile','r');
        xb2=fscanf(fileID,'%d ', [largo 4]);
```

```
        fclose(fileID);

        size(data.resultado)
        total(1:data.w,1) = (data.w-1:-1:0)*data.w;
        total(1:data.h,2) = (data.h-1:-1:0)*data.h;
        total(:,3) = total(:,1);
        total(:,4) = total(:,2);

        waitbar(2/3);

        size(xb2)
        size(total)

        lin = xb2./total;
        data.graph1 = lin(:,1);
        data.graph2 = lin(:,2);
        data.graph3 = lin(:,3);
        data.graph4 = lin(:,4);

        waitbar(3/3);
        close(wait);

        program = strcat(data.LocalMainDir, '\contador2.exe c infile outfile2');
        system(program);
        fileID = fopen('outfile2','r');
        xb3=fscanf(fileID,'%d ');
        fclose(fileID);

        data.graph5 = xb3(:,1);

        set(gui.CaractBtn,'Enable','off');
        set(gui.caracPopup,'Value',1);

        set(gui.axesFigure1,'Visible','on');
                axes(gui.axesFigure1);
                plot(data.graph1);

                set(gui.axesFigure2,'Visible','on');
                axes(gui.axesFigure2);
                plot(data.graph3);

    end

    function VerGrafica(~,~)

        val2 = get(gui.caracPopup,'Value');
        str2 = get(gui.caracPopup,'String');

        switch str2{val2}
            case 'Extremos'
                set(gui.axesFigure1,'Visible','on');
                axes(gui.axesFigure1);
                plot(data.graph1);

                set(gui.axesFigure2,'Visible','on');
                axes(gui.axesFigure2);
                plot(data.graph3);

            case 'Lineas Completas'
                set(gui.axesFigure1,'Visible','on');
                axes(gui.axesFigure1);
                plot(data.graph2);
```

```
                set(gui.axesFigure2,'Visible','on');
                axes(gui.axesFigure2);
                plot(data.graph4);

            case 'Circulos'
                set(gui.axesFigure1,'Visible','on');
                axes(gui.axesFigure1);
                plot(data.graph5);

                set(gui.axesFigure2,'Visible','off');

        end
    end

    function Zoom(hObject,eventdata,handles)
        zoom;
        data.zoomOut = 0;
        data.zoomIn = 1;
        if data.zoomIn == 1
            set(gui.zoomOutButton,'value',0);
        else
            set(gui.zoomOutButton,'value',1);
        end

    end

    function ZoomOut(hObject,eventdata,handles)
        zoom(0.5);
        data.zoomOut = 1;
        data.zoomIn = 0;
        if data.zoomOut == 1
            set(gui.zoomOutButton,'value',0);

        else
            set(gui.zoomOutButton,'value',1);
        end
    end
set(gui.listFolders,'Value',1);

end
```