UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE INGENIERÍA
UNIDAD ENSENADA

PROGRAMA DE POSGRADO
EN CIENCIAS E INGENIERÍA

TESIS

Desarrollo de una aplicación de correlación digital que haga uso de filtros lineales y no lineales compuestos

---

Presenta:

Ariel Alfredo Padilla Ramírez

Director
Dr. Josué Álvarez Borrego

Ensenada, Baja California, México, Julio del 2009

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE INGENIERÍA
UNIDAD ENSENADA

"Desarrollo de un aplicación de correlación digital que haga uso de filtros lineales y no lineales compuestos"

TESIS

Que para obtener el grado de maestría en ingeniería presenta:

Ariel Alfredo Padilla Ramírez

Aprobada por:

Dr. Josué Álvarez Borrego
Director de tesis

M. I. Juan Pablo Torres Herrera
Miembro del comité

Dr. José de Jesús Zamarripa Topete
Miembro del comité

Ensenada Baja California, México. Junio 2009

ASUNTO: Voto aprobatorio sobre trabajo
De tesis de grado de Maestro en ingeniería

Dr. Juan Ivan Nieto Hipólito
Coordinador de Posgrado
Facultad de Ingeniería-Ensenada
P R E S E N T E

Después de haber efectuado una revisión minuciosa sobre el trabajo de tesis presentado por el C. ARIEL ALFREDO PADILLA RAMÍREZ para poder presentar la defensa de su examen y obtener el grado de Maestro en Ingeniería, me permito comunicarle que he dado mi voto aprobatorio, sobre el trabajo titulado:

"Desarrollo de un aplicación de correlación digital que haga uso de filtros lineales y no lineales compuestos"

Esperando reciba la presente de conformidad, quedo de Usted.

Ensenada, B.C. a día 3 del mes de Junio del 2009

M. I. Juan Pablo Torres Herrera

c.c.p expediente

ASUNTO: Voto aprobatorio sobre trabajo
De tesis de grado de Maestro en ingeniería

Dr. Juan Ivan Nieto Hipólito
Coordinador de Posgrado
Facultad de Ingeniería-Ensenada
P R E S E N T E

Después de haber efectuado una revisión minuciosa sobre el trabajo de tesis presentado por el C. ARIEL ALFREDO PADILLA RAMÍREZ para poder presentar la defensa de su examen y obtener el grado de Maestro en Ingeniería, me permito comunicarle que he dado mi voto aprobatorio, sobre el trabajo titulado:

"Desarrollo de un aplicación de correlación digital que haga uso de filtros lineales y no lineales compuestos"

Esperando reciba la presente de conformidad, quedo de Usted.

Ensenada, B.C. a día 3 del mes de Junio del 2009

Dr. José de Jesús Zamarripa Topete

c.c.p expediente

RESUMEN

Los avances tecnológicos en el mundo actual nos inclinan al desarrollo de tecnologías automatizadas, capaces de realizar trabajos manuales en poco tiempo y permitiendo ocuparlo en otro tipo de labores o investigaciones. En este trabajo se presenta el desarrollo de una aplicación de reconocimiento automático de imágenes de organismos planctónicos (SISREC) que permitirá la agilización de investigaciones en diversas áreas biológicas, ocupando para ello diversos tipos de filtros de correlación. Se utilizó el poder matemático de Matlab y la robustez de .Net, interconectados a través de una librería (dll) generando una aplicación distribuible. Se muestran los resultados en forma grafica por medio de imágenes así como en formato numérico listo para ser importado a una aplicación para su manejo estadístico. Se trabajó con 3 filtros lineales (CMF, POF e IF) y un nuevo filtro no lineal compuesto, generado este ultimo en tiempo de ejecución. Se delimitó un umbral de confiabilidad para determinar los límites de muestreo gráfico de resultados superiores al 90%.

Se utilizaron 420 imágenes de copépodos correspondientes a 7 especies distintas tanto de hembras y machos, encontrando resultados del 100% de confiabilidad. También se realizaron pruebas con diferentes tipos de imágenes de letras, rotíferos y diatomeas, abarcando en todas ellas resultados satisfactorios. Cuenta con 3 pantallas, la principal donde se cargan las imágenes de las muestras a utilizar así como la selección del filtro deseado para la tarea, resultados, donde aparecen los picos de correlación obtenidos del proceso y, finalmente una pantalla auxiliar que permite guardar los resultados en formatos gráficos (jpg o bmp) o mandarlos directo a una impresión.

SISREC es una aplicación computacional basada en sistemas x86 de 32 y 64 bits tal cuales son Xp y Vista respectivamente. Es capaz de lograr reconocimientos de organismos por medio de sistemas de correlación de forma automática, logrando minimizar el tiempo ocupado en reconocimientos manuales en más de un 200% y con una confiabilidad superior al 90%.

ABSTRACT

The technological advances in the world permit us the development of new automated technologies, of this way, we have the capacity of developing manual work in short time and to use the rest of the time in another type of labors or investigations. In this thesis the development of a plankton image automatic recognition system is presented. This system will permit a fast streamlining in diverse fields of biological sciences using different correlation filter types. The Matlab power joint with .Net through the dll library is used. The results are shown graphically using images and numerical format which are used in statistical analysis. Three different linear filters were used (CMF, POF and IF) and a new composite nonlinear filter. This last one is generated in executing time. A threshold was determined in order the results have a 90% of confidence.

420 copepods images were used corresponding to seven different species with females and males included in the samples. The results showed a 100% of confidence level. Also, several tests were realized with different types of images of letters, butterflies, rotifs and diatoms, all the results were satisfactory. The platform has 3 screens, one screen has the information of the filters, another the problem image and the last one show the results of the identification. Finally, it exists an auxiliary screen to keep the results in graphic format (jpg or bmp) or send the results to the printer. To use this kind of system give us the automatization of an as laborious process as it is it the identification of organisms direct to microscope.

SISREC is a computational application based in systems x86 of 32 and 64 bits like Xp and Vista respectively. It can to identify different kind of microorganisms using the correlation in automatic way, thus, the time consuming is reduced more than 200% when is compared with manual work and it has a level of confidence at least of 90%.

A mis padres:
Patricia Ramírez Aroche y Alfredo Padilla Anguiano
A mis hermanos:
Paulina, Aldo y Sheyla
A mis familias adoptivas:
Mares Padilla y Padilla Vega

Agradecimientos

Agradezco sinceramente:

- A mis padres por el apoyo brindado a lo largo de mi vida y más aun estos últimos años, esto es de ustedes.
- A mis hermanos y sobrinos, Pau, Atun, Itzela, Cachepunk y Chimpankey, por soportar la distancia y la gran incomunicación durante este tiempo.
- A la familia Padilla Vega. Mario, Monina, Mayeyo, Beto e Israel por recibirme con los brazos abiertos, ustedes son parte importante de este logro.
- Al Dr. Josué Álvarez Borrego, por ser un gran maestro y guía, mil gracias por el apoyo durante estos años nada fáciles para mí.
- Al Dr. Víctor Zavala Hamz, por empujarme y devolverme al camino, gracias por ser un apoyo siempre presente.
- A mis compañeros el M. C. Jesús Lerma y el M. C. Ángel Coronel, por estar ahí siempre dispuestos a ayudarme y guiarme en todo momento.
- A mis amigos de Ciencias Marinas, Ramón, Vla, Salamanca y demás Guarapos KonKenKos, por esas noches de despeje necesarias en estos años.
- A mi roomie, Tenoch, gracias por la paciencia y sobre todo por el apoyo en todas esas situaciones pasadas.
- A mis amigos Tavo, Rul y Bto (Softtek).
- Al Consejo Nacional de Ciencia y Tecnología (CONACyT), por el apoyo económico brindado.

CONTENIDO

CONTENIDO (continuación)

LISTADO DE FIGURAS

LISTADO DE FIGURAS (continuación)

LISTADO DE FIGURAS (continuación)

LISTADO DE FIGURAS (continuación)

LISTADO DE FIGURAS (continuación)

LISTADO DE TABLAS

CAPÍTULO 1

INTRODUCCIÓN

1.1 Antecedentes

Las Ciencias Computacionales (CC) se han venido expandiendo a un gran número de áreas en los últimos 50 años, desde la aparición de la primera generación de computadoras, donde su construcción se basaba en bulbos. Los dispositivos de almacenamiento eran sumamente limitados, la programación se realizaba mediante tarjetas perforadas en lenguajes de bajo nivel, las aplicaciones primordiales se limitaban a cálculos matemáticos simples y se hacían esfuerzos en el almacenamiento de información. Conforme fue evolucionando la tecnología, los computadores fueron capaces de almacenar más información, ejecutar programas más eficientes, poderosos y sofisticados en menor tiempo y a un costo mucho menor, por lo tanto hubo una proliferación masiva de estas tecnologías en todos los sectores que atañen a nuestra sociedad (Mandell, 1979), siendo esta importante en el vivir de cada uno de nosotros, ya sea por: comunicaciones, seguridad, educación, entre otros. Conforme ha crecido esta necesidad de tecnología dentro de nuestra sociedad requerimos de otras alternativas, algunas de estas ya están presentes y su estudio no es nuevo, pero tal vez su explotación si, el reconocimiento de patrones o imágenes es una de ellas.

Actualmente el reconocimiento de patrones o imágenes ha tenido gran demanda en diversos campos de la sociedad, desde pequeñas empresas hasta instituciones gubernamentales; esto no es nuevo ya que la tecnología de reconocimiento de huella digital (identificación biométrica), el reconocimiento de rostros, entre otros, usan esta tecnología. Desafortunadamente su reconocimiento se basa en hardware mas que en software y por tanto el costo de su desarrollo es mas alto; algunas aplicaciones adicionales es el reconocimiento de objetos microscópicos y macroscópicos.

El concepto de utilizar computadoras digitales para procesar imágenes tiene aproximadamente tres décadas, por lo que el Procesamiento Digital de Imágenes (DIP,

Digital Image Processing) ha emergido como disciplina por sí misma (Bueno-Ibarra, 2005), fundamentada en la formulación de una sólida base matemática completa (Pratt, 1995), (Russ, 1992), (Watkins *et al.*, 1993), (Cho *et al.*, 1993).

Ha emergido una industria completa que involucra el desarrollo de sistemas, periféricos y software basados en computadoras que están especialmente diseñados para accesar, procesar y desplegar información contenida en imágenes. Por lo tanto, todo este soporte ha traído como consecuencia una rápida expansión y evolución de esta disciplina (Green, 1983).

Bueno-Ibarra (2005) en "Desarrollo de una tecnología sistematizada para la adquisición y análisis de partículas biogénicas" menciona que el DIP tiene dos funciones principales: 1) El mejoramiento de la información pictórica para el análisis e interpretación humana y 2) El procesamiento de los datos de la escena digital para la percepción automática por máquinas (MP, *Machine Perception*).

Dentro de las primeras encontramos aquellos procesos y metodologías capaces de hacer mejoras a la imagen para su correcta interpretación o análisis, por ejemplo, en el área geográfica hay un procedimiento de procesamiento de imágenes para convertir los niveles de grises de una imagen y definir ciertos niveles del suelo ya sea por erosión o la presencia de alguna colina o pendiente; en el área médica pasa algo similar con las imágenes de rayos X. Como estos, muchos casos hay dentro de diversas áreas astronomía, física, medicina, etc. (Bueno-Ibarra, 2005).

Dentro de la segunda función entran las MP, donde los procesos y metodologías están enfocadas a extraer información de la imagen por medio de cálculos y poder ser procesados por una computadora, por mencionar algunos: coeficientes de Fourier, momentos estadísticos, entre otros (Bueno-Ibarra, 2005). Problemas típicos en MP son reconocimiento automático de caracteres, visión en máquinas industriales, procesamiento automático de huellas digitales, análisis de muestras biológicas y análisis de imágenes de satélite para predicción del tiempo (González y Woods, 2008).

En el transcurso de los años el interés por automatizar los procesos de reconocimiento de imágenes fue fluyendo de una manera paulatina, donde primero como se mencionó anteriormente, su soporte era en esencia el hardware: microscopios y sistemas ópticos, pero al pasar de los años la influencia digital cobró más interés, tal es el caso de Zavala-Hamz y Álvarez-Borrego (1997) que reconocieron organismos planctónicos del genero *Acartia* y *Calanus*, a nivel sexo no solo con filtros armónicos circulares sino también con el uso de imágenes digitales. Pech-Pacheco y Álvarez-Borrego (1998) en "Procesos óptico-digitales aplicados al reconocimiento de cinco especies de fitoplancton" donde, utilizando sistemas óptico-digitales hicieron identificaciones de 5 especies de fitoplancton del genero *Ceratium* con una certeza del 90%, en este caso se hace notar el uso de algunos elementos más como el scanner digital para obtener la matriz de datos de las fotografías.

Ya adentrada la era digital es de mencionar una gran cantidad de documentos referentes al procesado de imágenes, donde el uso de elementos ópticos no ha sido erradicado, pero tal vez si sustituido por herramientas de esta generación, todo con la finalidad de hacer mas rápido y preciso el trabajo de un investigador, tal es el caso de la identificación de parásitos en el pez Botete (Fájer y Álvarez-Borrego, 2002), donde usando correlación invariante digital logran identificar los parásitos, ahorrando tiempo de observación y aumentando un nivel de confiabilidad, o el de Rodríguez (2003) que identifica una especie de parásitos de peces llamado Tricodina. Como este, hay muchos artículos y libros por citar algunos, por ejemplo en el análisis de imágenes de cromosomas en abulón (Gallardo-Escalante *et al*, 2004), donde se muestra los beneficios en tiempo y calidad en la identificación, mostrando los antecedentes previos que se han desarrollado, permitiéndonos fundamentar más este trabajo de tesis.

Siempre se ha buscado la forma de optimizar tanto los procesos de cómputo como la calidad de identificación, tal es el caso de la identificación de la bacteria del cólera en "identificación de *Vibrio Cholera* 01 mediante correlación invariante", donde el objetivo fue evaluar la efectividad de sistemas de correlación con imágenes de color para el reconocimiento de la bacteria, y también desarrollar filtros invariantes comparándo los filtros clásicos con los filtros sólo de fase (Álvarez-Borrego, *et al,* 2000, 2002). La

tuberculosis ha sido también parte de este avance en identificación, en “técnicas automáticas de identificación de la bacteria de la tuberculosis” (Forero *et al*, 2001 y Forero *et al*, 2002), donde presenta una nueva forma de detectar la bacteria basados en su contorno y fluorescencia, eliminando así la identificación errónea de algunos otros objetos, redujo el tiempo de procesado al reducir las partículas de los candidatos a buscar.

Álvarez-Borrego y Chávez-Sánchez (2001) detectaron el virus IHHN en tejido de camarón por medio de sistemas de correlación digital, utilizando filtros de fase en los tres campos de color sin considerar escala y rotación, permitiendo la identificación de éstos en pocos segundos.

Encontrar objetos ocultos en imágenes también ha sido analizado, tal es el caso de González-Fraga (2005) en "Reconocimiento de objetos parcialmente ocultos usando correlación de filtros con entrenamiento", donde utilizando varios tipos de filtro logra separar la imagen a buscar del fondo de la imagen problema, para así identificarla claramente.

El reconocimiento de imágenes ha demostrado su impacto e importancia en múltiples de áreas como anteriormente se mencionó, dentro del campo de la biología se han dado muchos importantes avances, tal es el caso de un método para discriminación entre cuatro especies de copépodos con la posibilidad de discriminar entre sexos, independientemente de la posición, rotación o escala, todo esto con filtros de fase. Lo que falta ahora es un catalogo de todas las especies y sus regiones (Castro-Longoria *et al*, 2001). En *“*Discriminación entre especies *Acartia* (Copepoda: Calanoida) utilizando sus patrones de difracción en un sistema de correlación digital invariante a posición y rotación” (Álvarez-Borrego y Castro-Longoria, 2003) se logra discriminar entre especies del genero *Acartia*, genero con mayor índice de especies dentro de la base de la cadena alimenticia.

El análisis citogenético en abulones por parte de Gallardo-Escárate (2005), donde se utilizó el procesado de imágenes para el análisis de cromosomas de cuatro especies de abulones, optimizando y automatizando procesos, así como la identificación de brazos largos y cortos

de algunos pares de cromosomas. De la misma manera realizó un análisis para la localización cromosomal en peces, de varios tipos de DNA ribosomal. Se ha logrado la estimación del tamaño del genoma de *Argopecten purpuratis*, usando análisis de imágenes de fluorescencia. Basados en esto, recientemente se ha publicado una nueva metodología para estimar el tamaño del genoma (Álvarez-Borrego *et al*, 2007) basados en el análisis de la intensidad de fluorescencia en las imágenes.

Muchos investigadores han desarrollado gran cantidad de algoritmos para el reconocimiento de imágenes, algoritmos para quitar ruido en imágenes de color, donde primero se detectan incongruencias en la imagen base buscando relaciones entre los componentes de color y después se separa la información de color e intensidad (Kober, V. *et al*, 2007); filtros de correlación para el reconocimiento de patrones de objetos, donde usando términos de correlación de variables enfocadas a objetos, se hacen las comparaciones de fragmentos de imágenes (González-Fraga *et al*, 2006).

En procesos industriales se han utilizado filtros de fase para verificar por medio de imágenes, la ubicación correcta del palillo de la paleta, ha ayudado al proceso de producción, permitiendo la automatización de procesos y a su vez un aumento en la calidad del producto (Hernández-Constante, *et al*, 2007).

Pocos casos existen sobre el desarrollo de un sistema amigable al usuario para este tipo de trabajos, el mas notorio y uno de los mas importantes es el desarrollado por el grupo de investigación GVA de la Escuela Universitaria Técnica Industrial de la Universidad Politécnica de Madrid, donde se realizan investigaciones en el ámbito de la visión artificial, el proyecto denominado “Análisis de imágenes biomédicas para la cuantificación del nivel de fragmentación de los espermatozoides humanos y de la carga vírica de la hepatitis C”, produjo el desarrollo de dos Software actualmente usados en los hospitales españoles HaloCount y VirusCount (Santos, 2005). Otro tipo de plataforma, es el desarrollado por la compañía SolexVision pero que es un sistema que no está enfocado solamente a la identificación de objetos sino también al control de un microscopio.

México ha mantenido de una u otra forma el interés en el área, un claro ejemplo de esto es el mostrado por Bueno-Ibarra (2005) en “Desarrollo de una tecnología sistematizada para la adquisición y análisis de partículas biogenéticas”, donde muestra un sistema donde existe una automatización de procesos en el análisis de partículas, obtención automática de imágenes, dejando también la posibilidad de uso de una manera distribuida (tal una red LAN).

En esta tesis se desarrollará una aplicación de correlación para identificación de objetos, que esté soportado por una plataforma computacional amigable al usuario escrita en C#, con la cual podamos acceder a la imagen o conjunto de imágenes que tienen la información que deseamos reconocer, esto con el uso de un nuevo filtro no lineal.

1.2 Planteamiento del problema

Como ya se ha mencionado el reconocimiento de patrones o imágenes ha tenido un gran auge en los últimos años en diversidad de campos de estudio y aplicaciones. Dentro del área biológica en el reconocimiento de plancton tenemos muchos ejemplos de ello desde hace algunos años, como el de Zavala-Hamz y Álvarez-Borrego (1996 y 1997) que usan patrones de difracción como herramienta para reconocer copépodos y hacen reconocimiento de microorganismos marinos mediante filtros CH, respectivamente; o el uso de procesos óptico-digitales para el reconocimiento de cinco especies de fitoplancton por parte de Pech-Pacheco y Álvarez-Borrego (1998) esto, por mencionar solo algunos. Estos estudios esta apoyados de sistemas ópticos en su mayoría (hardware) y con el paso del tiempo ha sido notable la necesidad de hacer mas automáticos los procesos y ahorrar tiempo sin perder la calidad de identificación por el contrario hacerla mucho mas efectiva.

El aumento en la calidad de dispositivos computacionales tanto en almacenamiento como en velocidad, han permitido el incremento de este tipo de investigaciones y de alguna u otra manera su automatización, tal es el caso de Pech-Pacheco (1998) en “Analizador automatizado de partículas biogénicas” donde realizó identificación de especies de

dinoflagelados de una manera hibrida (óptico-digital) automatizada; mas tarde Bueno-Ibarra (2005) mejora esta técnica notablemente en “desarrollo de una tecnología sistematizada para la adquisición y análisis de partículas biogénicas", ahorrando tiempo de ejecución, procesos computacionales, espacio de almacenamiento y sobre todo la obtención de imágenes para su análisis de una manera automática.

Es notable el avance en el campo, no solo en la búsqueda de la automatización sino también en el desarrollo de nuevos filtros para identificación, mucho más precisos y por lo tanto más complejos, dando una claridad mayor a esta. Los filtros están divididos en dos tipos en base a sus propiedades: filtros lineales y no lineales; cada uno tiene sus ventajas y desventajas dependiendo su aplicación; para nuestro tema de estudio los dos son eficientes, pero la calidad de identificación de los no lineales es mucho mayor.

El desarrollo de nuevas herramientas y nuevos filtros obliga a los investigadores a conocer los medios de una manera mas específica, a adentrarse más en la tecnología (González y Woods, 2008); desafortunadamente esto no siempre pasa y el investigador prefiere seguir con las herramientas comunes de identificación. En base a esto es predecible decir que la falta de uso de los filtros de reconocimiento de los últimos 5 años no se ven utilizados debido a la falta de una herramienta que haga un uso de ellos de manera sencilla (Álvarez-Borrego, 2008).

En esta tesis se pretende abordar ese problema desarrollando un nuevo filtro no lineal apoyado de un lenguaje de programación de alto nivel que nos permita hacer una interfaz amigable al usuario.

1.3 Preguntas de investigación

1. ¿Existen nuevos filtros no lineales para el reconocimiento de imágenes en los últimos 5 años?
2. ¿Por qué se mantiene el uso de filtros lineales para el reconocimiento de imágenes?
3. ¿Existen sistemas de correlación que hagan uso de filtros no lineales?
4. ¿Por qué es menor el uso de filtros no lineales en reconocimiento de imágenes?
5. ¿Qué es un filtro no lineal en procesamiento de imágenes?

Las preguntas principales de este estudio son:

1. ¿Qué es un filtro no lineal en procesamiento de imágenes?
2. ¿Existen sistemas de correlación que hagan uso de nuevos filtros no lineales?
3. ¿Cuál es la razón del aun uso de los viejos filtros lineales en reconocimiento de imágenes?

Variables:

1. Filtro lineal.
2. Filtro no lineal.
3. Imagen.
4. Correlación.

Pregunta de investigación:

"¿Existen aplicaciones de correlación que hagan uso de nuevos filtros no lineales para el reconocimiento de imágenes?"

1.4 Hipótesis:

- El uso de nuevos filtros no lineales para el reconocimiento de imágenes es posible con el desarrollo de una aplicación de correlación que haga uso de ellos.

1.5 Objetivo General

"Desarrollar una aplicación de correlación digital amigable que haga uso de un filtro no lineal compuesto para el reconocimiento de objetos, soportado por una plataforma computacional."

1.6 Objetivos Específicos

1) Reconocer los beneficios de los filtros no lineales de los últimos años con respecto de los filtros lineales.

2) Desarrollar un nuevo filtro no lineal.

3) Desarrollo del algoritmo del nuevo filtro no lineal.

4) Desarrollo de una aplicación de correlación con el nuevo algoritmo.

5) Realización de una aplicación computacional para el sistema de correlación digital.

1.7 Alcances y Limitaciones

El presente proyecto está dirigido a personal sin conocimientos matemáticos en reconocimiento de imágenes, se pretende mostrar la facilidad que es el reconocimiento de éstas, independientemente del campo de estudio de la persona que lo utilice.

Se realizará un nuevo filtro no lineal y su plataforma de trabajo, la cual será capaz de ejecutarse en cualquier computador que cumpla los requerimientos marcados en infraestructura, esto debido al alto costo computacional que requieren los procesos para el reconocimiento.

Se espera que en un futuro cercano más áreas de estudio estén interesadas en el desarrollo de aplicaciones similares pero con enfoques más específicos como reconocimiento de alguna partícula en especial, algún tipo de organismo, objeto en una imagen satelital, por mencionar algo. En estudios anteriores de reconocimiento de imágenes se trabaja con filtros no robustos y en ocasiones el tiempo de detección del objeto y el costo computacional es mucho más alto.

Las características que centran este trabajo de tesis son:

Técnicas:

- Los filtros en los que se basa este estudio son en filtros lineales y no lineales para reconocimiento de imágenes.
- La resolución de las imágenes a tratar será de 256x256.
- El formato a utilizar será el BMP, ideal para la presentación de fotografías de gran calidad, trabajos de arte o imágenes con escalas de grises y sin pérdida de información.
- PC de la familia x86 (32 bits y 64 bits).
- Sistema Operativo Windows XP o Windows Vista.
- Trabajara en una sola estación (no red).

Usuarios:

- Investigadores que utilicen procesamiento de imágenes.

- Laboratoristas con necesidad de identificación de imágenes.
- Ingenieros y profesionistas que requieran reconocer objetos en imágenes.

1.8 Infraestructura

Hardware:

- Windows Xp/Vista.
- 400 Mb de espacio en disco.
- 712 Mb Ram (recomendado>1024).
- T. Video 32 Mb (preferencia no integrada).
- Procesador mínimo 2.5 Ghz o 1.8 en Dual.

Software:

- Net Framework 2.0.
- MCR (Matlab Component Runtime).
- Windows Installer 3.1.

1.9 Justificación

Las técnicas de procesamiento de imágenes y los diversos algoritmos matemáticos utilizados en la física-óptica, pueden ser utilizadas en combinación con otras técnicas tales como la citogenética (análisis del número cromosómico) y marcadores moleculares para identificar especies, analizar poblaciones, reconocer y caracterizar los progenitores de un híbrido, estudiar la organización del genoma y la arquitectura nuclear, etc. Se han realizado estudios para determinar mediante imágenes fluorescentes e hibridación *in situ* el tamaño genómico y el contenido de ADN cromosómico de varias poblaciones de abulones. Con la utilización de éstas técnicas se discuten las relaciones cromosómicas entre las diferentes

especies estudiadas, la distribución, la existencia de híbridos, las relaciones filogenéticos con otros abulones del mundo (Gallardo, *et al*, 2004).

Por otro lado, la elevada complejidad taxonómica del plancton en general, requiere de microscopios inteligentes que puedan contar con bases de datos que tengan la información existente para dar respuestas rápidas a las preguntas que se hacen día a día. El plancton es importante tanto en ecosistemas continentales como oceánicos ya sea por el simple conocimiento científico, así como porque son la base de la cadena alimenticia, por el uso que se le puede dar para conocer las corrientes marinas, la contaminación, los efectos de los procesos antropogénicos, la importancia en la transferencia de energía, la importancia del plancton en las poblaciones comercialmente importantes, los efectos en cambios climáticos, la productividad, regiones biogeográficas, marea roja etc., Sin embargo, la identificación de microorganismos del fitoplancton y zooplancton requiere de profesionistas altamente especializados en los diferentes grupos zoológicos que lo componen tales como larvas, huevos de peces, crustáceos, moluscos, numerosos microorganismos como los protozoarios, rotíferos, foramíniferos, sifonóforos, medusas etc., y en las diversas familias de microalgas. La estructura de estos microorganismos es compleja y difícil por lo tanto la labor de los diversos trabajos que requieren de su conocimiento es ardua, toma tiempo pero sobre todo en muchas ocasiones por las altas cantidades de muestras a revisar no es muy precisa. En varios trabajos se presentan los avances realizados para el estudio de fito y zooplancton los cuales servirán de base para que en un futuro cercano sea posible el desarrollo de un sistema automatizado de identificación de estos organismos. Con un catálogo de imágenes digitales elaborada por un experto y la aplicación de técnicas de correlación invariante se podrían identificar rápidamente diferentes especies ahorrando así tiempo y esfuerzo (Pech-Pacheco y Álvarez Borrego, 1998).

Otro grupo de organismos importantes y en muchas ocasiones difíciles de identificar son las bacterias. En el caso particular de la bacteria *Vibrio cholerae* 01, responsable de la enfermedad del cólera cuenta con métodos de identificación no muy efectivos (medios de cultivo y pruebas bioquímicas). Para solucionar este problema, se han desarrollado técnicas moleculares (PCR) que son altamente específicas. Sin embargo, para evaluar numerosas muestras ambientales, estas pruebas resultan ser sumamente caras. El sistema de correlación a color multicanal utilizado por Pérez-Mouriño y Álvarez-Borrego, 2006, en lo referente a la eficiencia del sistema para el

reconocimiento de *Vibrio cholerae* O1 fue casi del 100%. El sistema es una herramienta útil independiente de su forma, tamaño y posición en muestras de laboratorio y ambientales, que permite contar el número de organismos presentes en cada imagen problema con mayor eficiencia.

Otra bacteria de importancia en la salud pública es la bacteria *Mycobacterium tuberculosis* causante de la tuberculosis pulmonar y principal causa de muerte por enfermedad infecciosa en el mundo. El diagnóstico de la micobacteriosis debe ser hecho a través del cultivo de muestras de esputo, usando técnicas que carecen de la sensibilidad necesaria, por lo cual los especialistas deben esperar los resultados de los cultivos hasta 2 meses. El estudio automático puede conllevar varias ventajas, tales como una reducción sustancial en el trabajo de los especialistas, mejorar la sensibilidad de la prueba, mayor precisión en el diagnóstico y un aumento en el número de imágenes que pueden ser analizadas. Los resultados obtenidos por (Forero, et al, 2002) indican que la técnica de segmentación desarrollada permite extraer la mayor parte de bacilos presentes en una imagen, eliminando la mayoría de detritus. Los resultados obtenidos muestran que la técnica desarrollada proporciona un buen funcionamiento en términos de especificidad y sensibilidad, y por lo tanto es susceptible de ser utilizada en un sistema automatizado de análisis de muestras.

El diagnóstico de enfermedades por histopatología es la técnica que se lleva a cabo para el reconocimiento de cambios en las células, tejidos y órganos, así como para la identificación de patógenos presentes en los mismos. En el caso de la actividad acuícola son utilizadas ampliamente para determinar la presencia de patógenos (virus, bacterias, hongos, endoparásitos) o cambios patológicos derivados de deficiencias nutricionales, factores ambientales o genéticos en poblaciones cultivadas de peces, crustáceos o moluscos. Para ello hay que revisar a veces cientos de muestras, por consiguiente el procesamiento y análisis de las muestras de los organismos lleva días, por lo que los productores demandan rapidez y precisión en la entrega de los diagnósticos. Si se logra hacer una lectura de las laminillas en forma automática, rápida y precisa, los especialistas serán relevados de la identificación de patógenos conocidos o de daños muy característicos y se dedicarán solamente a la búsqueda de nuevas patologías o nuevos patógenos, esto redundará en una mayor eficiencia para esta tarea. Con el fin de probar si la metodología de correlación a color y procesamiento de imágenes podía ser usada para identificar

cambios histopatológicos, se llevó a cabo un estudio previo en el que se usaron diapositivas con imágenes de tejidos de camarón con cuerpos de inclusión del virus conocido como IHHN (Álvarez-Borrego y Chávez-Sánchez, 2001). Con este primer trabajo se demostró el potencial de la técnica de diagnóstico y que la metodología puede aplicarse para otros campos de la histopatología o en estudios epidemiológicos de cualquier organismo, incluyendo el humano. Se mostró también que los tiempos de identificación se pueden reducir a segundos pero que es necesaria mayor investigación para considerar aspectos como rotación, escala, color, morfología de los virus y el uso de diferentes clases de filtros dependiendo de la complejidad de los cuerpos de inclusión a reconocer.

Las técnicas óptico-digitales no son solamente útiles para microorganismos como virus, bacterias o el plancton, al igual que estos grupos de micro-organismos, la identificación específica de parásitos usando datos morfológicos es lenta y laboriosa, requiere de preparaciones de calidad que serán utilizadas para medir y analizar con claves de identificación, comparar con las descripciones originales y consultar con expertos del grupo taxonómico de que se trate. Con el fin de identificar parásitos en dos especies de peces, se tomaron imágenes con una cámara de alta resolución conectada a un microscopio y utilizando un programa instalado en una computadora. Se aplicaron las técnicas de filtros solo de fase con invariancias a posición, rotación y escala. Los resultados indican que este sistema es sencillo y que el sistema de filtros es una herramienta útil para la identificación de parásitos independientemente de su forma, tamaño y posición (Álvarez-Borrego y Fájer-Ávila, 2006). Este trabajo es el primer paso para desarrollar un sistema automático que facilite la identificación de parásitos a investigadores y biólogos que no sean especialistas en taxonomía que permita procesar un gran número de muestras en corto tiempo. El periodo de tiempo requerido para identificar una imagen en este sistema digital es de pocos segundos.

Así que en esta tesis se desea generar una plataforma computacional que sirva de apoyo a un sistema de identificación de objetos que hará posible que cualquier persona que no tenga conocimiento matemático en este tipo de investigación, pueda utilizar esta teoría para facilitar su tarea de campo.

CAPÍTULO 2

MARCO TEÓRICO

Los métodos del DIP son de gran interés en dos principales áreas de aplicación: basada en la interpretación de imágenes por medio de la percepción humana; y el procesamiento de datos de la imagen, transmisión y representación por la percepción automática de una máquina (González y Woods, 2008).

Dentro de la primera se encuentran sus orígenes desde los años 20's, pero con la invención del transistor en 1948 por Laboratorios Bell, la invención del circuito integrado por Texas Instruments en 1959, los lenguajes de alto nivel (COBOL y FORTRAN), el desarrollo de los primeros sistemas operativos (OS, *Operative System*) cerca de los años 60's, el primer microprocesador por parte de Intel en 1970 y finalmente la introducción de IBM con la primer computadora personal; fueron parte importante para su exponencial desarrollo, claros ejemplos de esto son el primer análisis de imágenes de la luna transmitidas por el *Ranger 7* al Jet Propulsion Laboratory en Pasadera California y el surgimiento de la tomografía computarizada en 1970. Al día de hoy el trabajo con imágenes para la interpretación humana se da en muchos campos como la medicina en el estudio de radiografías, la geografía en el análisis de imágenes satelitales, la arqueología en la reconstrucción de objetos a partir de imágenes; como estos muchos más en diversas áreas como la industria, leyes, criminología, entre otros. Todo esto con ayuda del DIP basándose en su contorno, brillo, color, forma, contraste, entre otros.

La segunda área de aplicación de las técnicas del DIP basada en la percepción de máquinas, consiste en obtener información de la imagen para procesarla por medios computacionales, ejemplos de este tipo de información son momentos estadísticos y coeficientes de la trasformada de Fourier por mencionar algunos. Problemas comúnmente tratados de esta manera automática son el reconocimiento de caracteres, visión-máquina en la industria al momento de ensamblar productos o inspeccionarlos, el área militar, reconocimiento de huellas digitales (biometría), reconocimiento de imágenes de satélite para determinación de

climas, etc. (González y Woods, 2008). Al paso de los años el avance tecnológico ha permitido el incremento de esta área al surgir más medios de almacenamiento, velocidades de procesado, sistemas de información, medios de publicación y divulgación, etc.

2.1 El Procesado Digital de Imágenes (DPI)

Una imagen puede ser definida como una función en dos dimensiones de la forma *f*(*x, y*) figura 1, donde *x* y *y* representan coordenadas espaciales, la amplitud de *f* en cualquier par de coordenadas (*x, y*) es denominada la intensidad o nivel de gris de la imagen en ese punto especifico.

$$f(x, y) =$$

Figura 1.- Imagen vista como función en dos dimensiones.

Cuando los valores de *x*, *y*, y los valores de la intensidad de *f* son todos finitos, de cantidades discretas, una imagen es llamada imagen digital (DI, *Digital Image*). En base a esto, el termino DIP proviene del trabajo con estas en computadores digitales. Cada imagen está conformada por cierto número de elementos, cada uno con una localización en particular, estos elementos son llamados de distintas formas: elementos de la imagen, elementos de la película (debido a su término en ingles, Picture), pels y el más común *píxel,* figura 2.

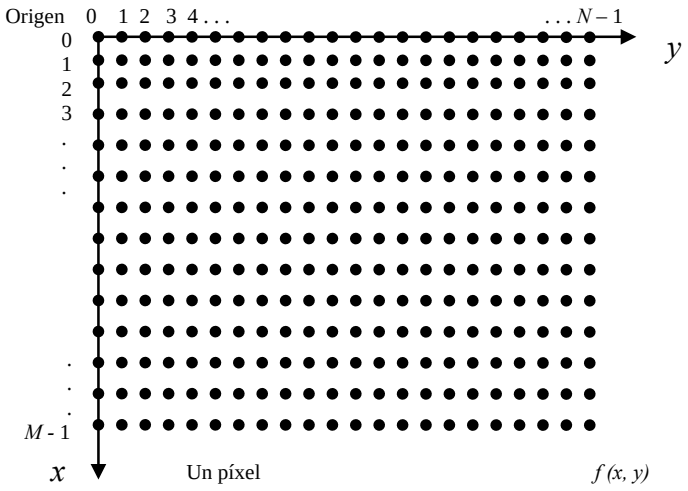


Figura 2.- Imagen vista en términos de píxel.

Por lo tanto se puede denotar como una matriz de MxN como se muestra en la figura 3.

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,0) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Figura 3.- Notación en forma de matriz de una imagen *f(x, y).*

La percepción visual del ser humano es limitada a ciertos campos debido a la banda en que se encuentran, limitándonos por el espectro electromagnético, como es el caso de los rayos gamma; justo aquí es donde el procesado de imágenes se detiene pero inicia el campo de la visión por computador (CV, *Computer Vision*). Una forma de definir o determinar el DIP como disciplina es cuando la salida y entrada de los procesos son imágenes.

Es difícil determinar con claridad el inicio y fin del DIP junto con la CV debido a la interacción mutua de ellas, es por eso que se utilizan tres niveles de procesos computarizados para aclarar un poco este paradigma:

- Nivel de procesamiento bajo, envuelven operaciones primitivas con imágenes pre-procesadas para reducir ruido, bajar o subir contraste, aumento de resolución, etc. Este nivel es caracterizado porque los datos de entrada y salida son imágenes.
- Nivel de procesamiento medio, envuelve tareas de segmentación, clasificación de estos segmentos para futuros procesamientos, clasificación (reconocimientos) de objetos individuales, etc. Es caracterizado porque los datos de entrada son imágenes pero los resultados son atributos generalmente extraídos de estas, el contorno y reflejo son algunos ejemplos.
- Nivel de procesamiento alto, envuelve el reconocimiento de objetos analizando la imagen usando funciones cognoscitivas normalmente asociadas a la visión.

En este trabajo de tesis los datos de entrada son imágenes al igual que los resultados, es por ello que se sitúa en el campo del DIP, más específicamente el reconocimiento de objetos o patrones dentro de estas.

2.2 Transformada de Fourier

Debido a que cualquier señal puede ser descompuesta en sus componentes frecuenciales la transformada de Fourier (FT, *Fourier Transform*) es de suma importancia en el DIP, ya que si se considera una DI, cada píxel de ella en el dominio frecuencial representa una frecuencia en particular dentro de esta en su dominio espacial. La FT tiene muchas aplicaciones dentro del DIP tales como filtrado, reconstrucción, suavizado, entro otros.

El trabajo en el campo de frecuencias o campo de Fourier es mas frecuente debido a que la manipulación de la imagen es mas eficiente desde este plano y también debido a que muchas operaciones son mas sencillas, particularmente algunos tipos de filtrado. Una característica muy importante de la transformada de Fourier es la capacidad de ser reconstruida o recobrada por un proceso inverso sin perdida de información, razón importante para trabajar en el “dominio de Fourier” y poder regresar al dominio espacial con la información integra.

Si la FT, *F(u)*, de una función continua, *f(x)*, de una sola variable, está definida como:

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-j2\pi ux} \, dx \,, \tag{1}$$

donde $j = \sqrt{-1}$. Consecuentemente, la transformada de Fourier inversa (IFT, *Inverse Fourier Transform*) de *f(u)*, resulta en la función original *f(x)*:

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{+j2\pi ux} \, du \,. \tag{2}$$

Para el caso de ecuaciones discretas *f(x), x* = 1, 2, 3,…M-1, las ecuaciones serian la transformada de Fourier discreta (DFT, *Discrete Fourier Transform*) en la ecuación (3) y la transformada de Fourier discreta inversa (IDFT, *Inverse Discrete Fourier Transform*) en la ecuación (4), donde se nota cierta similaridad con el par anterior ecs. (1 y 2), salvo la presencia de sumatorias en lugar de integrales y la existencia del múltiplo 1/*M*, múltiplo que dependiendo el autor será ubicado en la ecuación (3) ó (4), en casos muy extraños es posible encontrarlo en las dos ecuaciones pero de la forma $1/\sqrt{M}$.

$$F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) \, e^{-j2\pi ux / M} \,, \quad \text{para } u = 1, 2, 3,\ldots, \text{M-1} \tag{3}$$

$$f(x) = \sum_{x=0}^{M-1} F(u) \, e^{+j2\pi ux / M} \,, \quad \text{para } x = 1, 2, 3,\ldots, \text{M-1} \tag{4}$$

De forma general es notable observar que los componentes de la FT en la ecuación (1) son valores complejos, por consiguiente es común al trabajar con números complejos expresarlos en coordenadas polares, por lo tanto *F(u)* quedaría de la forma:

$$F(u) = |F(u)|e^{-j\phi(u)}, \tag{5}$$

donde

$$|F(u)| = \sqrt{R^2(u) + I^2(u)} \tag{6}$$

es la magnitud de la transformada de Fourier, y

$$\phi(u) = \tan^{-1}\left|\frac{I(u)}{R(u)}\right| \tag{7}$$

es la fase del ángulo de las transformada. *R(u)* y *I(u)* son la parte real e imaginaria respectivamente (González y Woods, 2008).

2.2.1 Transformada de Fourier Bidimensional

Para el manejo de imágenes debido a su naturaleza bidimensional es necesario acoplar la FT a esa forma, la que es del interés en este caso es la DFT, donde si se considera una imagen *f(x, y)* su transformada discreta bidimensional (BDFT, *Bidimensional Discrete Fourier Transform*) seria:

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\, e^{-j2\pi\left(ux/M + vy/N\right)} \tag{8}$$

para valores de u = 1, 2, 3,…,M-1 y v = 1,2,3,…,N-1

En donde $f(x,y)$ es la imagen digital de *MxN* y $F(u,v)$ la transformada discreta de Fourier en dos dimensiones, esto debido a que trabajamos con una función de 2 variables (imagen). Teniendo la trasformada $F(u,v)$, se puede obtener $f(x,y)$ es decir, la imagen original, a partir de la inversa de la trasformada discreta de Fourier:

$$f(x,y)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}F(u,v)\,e^{+j2\pi\left(ux/M+vy/N\right)} \tag{9}$$

para valores de x = 1, 2, 3,…,M-1 y y = 1,2,3,…,N-1

Al igual que en el caso unidimensional la ubicación del múltiplo, en este caso 1/*MN*, es indiferente y al estar presente en las dos seria de la forma $1/\sqrt{MN}$. Para el caso bidimensional el espectro de Fourier en coordenadas polares, magnitud y fase son:

$$F(u,v)=\left|F(u,v)\right|e^{-j\phi(u,v)}, \tag{10}$$

$$\left|F(u,v)\right|=\sqrt{R^2(u,v)+I^2(u,v)}, \tag{11}$$

$$\phi(u,v)=\tan^{-1}\left|\frac{I(u,v)}{R(u,v)}\right| \tag{12}$$

donde *R(u, v)* y *I(u, v)* son la parte real e imaginaria respectivamente.

2.2.2 Teorema de Convolución

La convolución de dos funciones *f(x, y)* y *g(x, y)* es otra función *h(x, y)*, definida por la doble integral:

$$h(x,y) = f(x,y) * g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x',y')g(x-x',y-y')dx'dy', \qquad (13)$$

donde * denota convolución.

Si *f(x, y)* y *g(x, y)* tienen una FT *F(u, v)* y *G(u, v)* respectivamente, entonces *f(x, y)*g(x, y)* tiene la transformada de Fourier *F(u, v)G(u, v)* (Bracewell, 2000), esto es, la convolución de dos funciones en el dominio espacial es la multiplicación de las transformadas de las dos funciones en el dominio frecuencial.

En base al teorema de convolución y considerando que $\Im\{.\}$ denota una FT, se definen las siguientes ecuaciones:

$$\Im\{f(x,y) * g(x,y)\} = F(u,v)G(u,v), \qquad (14)$$

$$\Im\{f(x,y)g(x,y)\} = F(u,v) * G(u,v). \qquad (15)$$

Por lo tanto es posible calcular la convolución de dos señales haciendo uso de la FT:

$$f(x,y) * g(x,y) = \Im^{-1}\{F(u,v)G(u,v)\} \qquad (16)$$

2.2.3 Teorema de Correlación

La correlación de dos funciones *f(x, y)* y *g(x, y)* es otra función *h(x, y)*, definida por la doble integral:

$$h(x,y) = f(x,y) \otimes g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x',y')g(x+x',y+y')dx'dy', \qquad (17)$$

donde ⊗ denota correlación.

Cuando las funciones son complejas es necesario usar el complejo conjugado de una de las funciones:

$$h(x,y) = f(x,y) \otimes g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x',y')g^*(x+x',y+y')dx'dy', \tag{18}$$

donde * indica el complejo conjugado de la función.
Basandose en lo anterior y considerando que $\Im\{.\}$ denota una FT, se puede resumir el teorema de correlación a las siguientes ecuaciones:

$$\Im\{f(x,y) \otimes g(x,y)\} = F(u,v)G^*(u,v), \tag{19}$$

$$\Im\{f(x,y)g^*(x,y)\} = F(u,v) \otimes G(u,v). \tag{20}$$

Por lo tanto es posible calcular la correlación de dos funciones haciendo uso de la FT:

$$f(x,y) \otimes g(x,y) = \Im^{-1}\left\{F(u,v)G^*(u,v)\right\} \tag{21}$$

2.3 Filtros Digitales

La definición de filtro puede llegar a ser algo burda e inconsistente en algunos términos dependiendo del campo en el que se este trabajando, en el caso de señales Winder (2002) los define: "*los filtros electrónicos permiten a algunas señales pasar pero detiene a otras*".

Los filtros digitales tienen su base teórica en los filtros análogos usados en la electrónica, la invención del computador personal en los años 60's fue ayuda importante para el desarrollo de estos, permitiendo cálculos mas precisos y simulaciones de ciertos procesos electrónicos sin necesidad de armar los componentes de manera física. El trabajo con señales dio pauta

para su utilización en el DIP y, aunque hay ciertas variantes en cuanto a la forma de trabajar del filtro en base al dominio en el que se este trabajando, espacial o de Fourier, la base matemática es la misma.

En el DIP el uso de filtros abarca un campo muy amplio desde segmentación o localización de bordes hasta reconocimiento de patrones, los principales filtros de correlación son derivados directamente del filtro clásico de acoplamiento, introducido por VanderLugt (1964), cada uno optimiza un criterio en particular, ya sea luz, señal ruido, entre o otros.

Se ha visto que una imagen esta conformada por un par de coordenadas *(x, y)* para cada píxel de la misma; en términos de frecuencia es exactamente lo mismo pero sus coordenadas son dadas por (1/*x,* 1/*y*) o como es mas común (*u, v*) (Watkins, *et al*, 1993).

El filtrado en el dominio de la frecuencia consiste en modificaciones a la transformada de Fourier de una imagen y entonces aplicar la transformada inversa obteniendo un resultado. Si se tiene una imagen digital $f(x,y)$, de *MxN*, una ecuación básica de filtrado estaría dada por la siguiente ecuación:

$$g(x,y) = \Im^{-1}\left[H(u,v)F(u,v)\right] \tag{22}$$

Donde $\Im^{-1}$ es la ITDF, $F(u,v)$ es la DFT de una imagen de entrada, $f(x,y)$, $H(u,v)$ es una *función filtro* o simplemente filtro, y $g(x,y)$ es la imagen filtrada; un diagrama del proceso se puede ver en la figura 6.

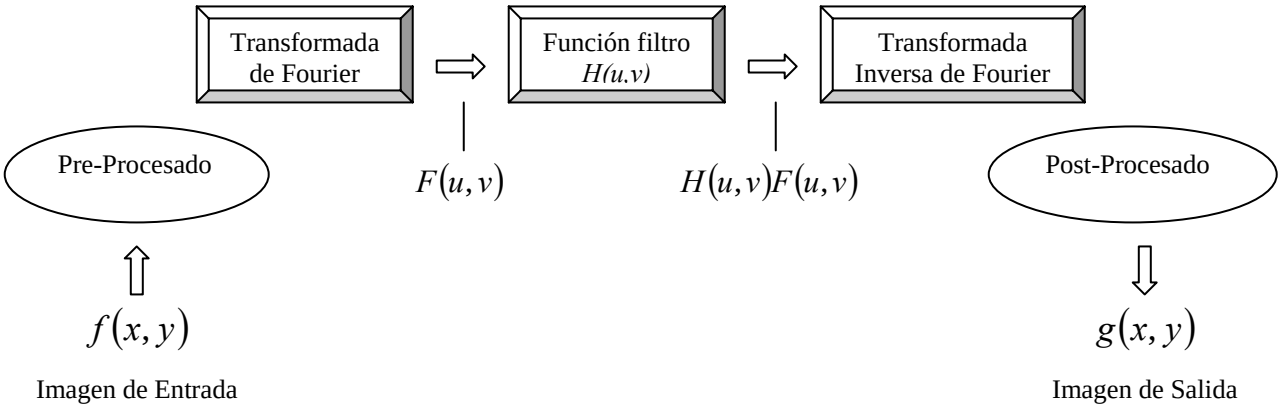
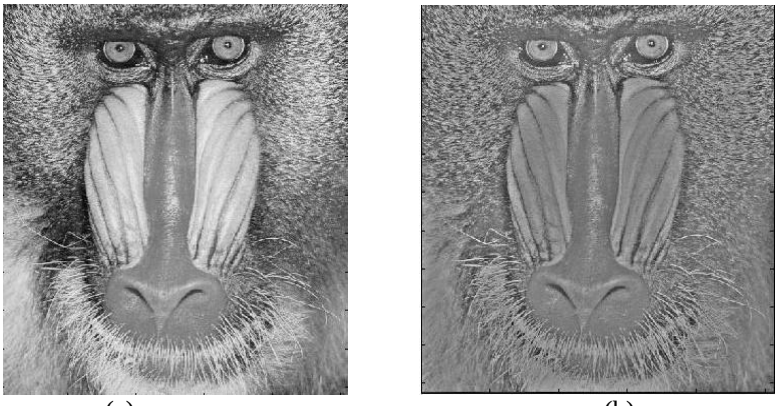


Figura 4.- Pasos básicos para el filtrado en el dominio de frecuencias.

El producto de $H(u,v)F(u,v)$ es simplemente una multiplicación de matrices, un detalle importante al trabajar con filtros en el campo de frecuencias es que al construir un filtro este debe ser del mismo tamaño que la imagen; un ejemplo de cómo trabaja un filtrado en el campo de frecuencias se ve en la figura 5, utilizando como ecuación filtro una campana de Gauss dada por la formula (23):

$$H(u,v) = e^{-\left[(u-((M-1)/2))^2+(v-((N-1)/2))^2\right]/k} \tag{23}$$

Donde la imagen es de tamaño *MxN, k* es el factor de amplitud que afectara la cantidad de frecuencias a dejar pasar, y (*u, v)* las variables de frecuencias acordes a la imagen de entrada.



(a) (b)

Figura 5.- Imagen de Baboon original (a) y después de aplicar la ecuación 23 como filtro (b).

2.3.1 Filtros de correlación lineales

Al trabajar en el procesado de imágenes es importante la clasificación de sus métodos, si es lineal o no lineal, esto entra dentro de los filtros tanto en el campo espacial como en el frecuencial.

En el caso de la linealidad, si consideramos un operador, *H,* este produce una imagen de salida, $g(x,y)$, a partir de una imagen, $f(x,y)$:

$$H[F(x,y)] = g(x,y) \tag{24}$$

H será considerado un operador lineal si

$$\begin{aligned} H\left[a_i f_i(x,y) + a_j f_j(x,y)\right] &= a_i H[f_i(x,y)] + a_j H\left[f_j(x,y)\right] \\ &= a_i g_i(x,y) + a_j g_j(x,y) \end{aligned} \tag{25}$$

Donde a_j, a_i, $f_i(x,y)$ y $f_j(x,y)$ son constantes arbitrarias e imágenes del mismo tamaño respectivamente. Esto es, cumplen la propiedad de la aditividad donde la suma de las entradas de una operación lineal es reciproco a la suma de los resultados de estas si se efectúan separadas; y la propiedad de la homogeneidad, donde la multiplicación por una constante a cada una de las entradas es lo mismo que la multiplicación de la salida por esa misma constante.

Los filtros de correlación lineal pueden ser implementados en un sistema correlador tanto óptico como digital. Desde el desarrollo e introducción del filtro clásico de acoplamiento se han propuesto distintos filtros de correlación optimizados para un mejor desempeño en ciertas áreas del reconocimiento de patrones.

Se mencionarán los principales filtros lineales basados en la correlación, sin embargo, existen muchos filtros más, optimizados para diferentes tareas que no discutiremos en este trabajo.

2.3.1.1 Filtro de acoplamiento clásico (CMF, *Classical Matched Filter*)

El CMF tiene la ventaja de que optimiza la relación señal a ruido (SNR) a la salida de éste cuando la señal de entrada (imagen problema) se encuentra degradada por ruido blanco aditivo. Las desventajas del CMF son que produce picos de correlación bastante anchos, lóbulos laterales grandes, relativamente baja discriminación y en su implementación óptica tiene poca eficiencia de luz.

La ecuación (26) es conocida como CMF, donde $T^*(u,v)$ y β son el complejo conjugado de la transformada de Fourier del objeto a reconocer y una constante arbitraria, respectivamente

$$H(u,v) = \beta \frac{T^*(u,v)}{P_n(u,v)} \equiv H_{CMF}(u,v) \tag{26}$$

Para el caso en que $P_n(u, v)$ y β son igual a uno, tenemos que $H_{CMF}(u, v)$ se puede expresarse simplemente como

$$H_{CMF}(u,v) = T^*(u,v) \tag{27}$$

2.3.1.2 Filtro Solo de Fase (POF, *Phase Only Filter*)

Este tipo de filtro es conocido por maximizar la eficiencia de la luz, una de sus desventajas radica en la poca capacidad de discriminación cuando un objeto de bajo contraste se encuentra en una escena con un fondo complicado (Javidi, 1989).

Tomando la ecuación (26), es posible obtener una función de transferencia para un POF. Debido a que $H_{CMF}(u, v)$ es una función compleja, su forma polar seria:

$$H_{CMF}(u,v) = |H(u,v)|e^{-j\phi(u,v)}, \tag{28}$$

Donde $H(u, v)$ es la magnitud y $-j\phi(u, v)$ es la fase del filtro.

Si se considera que no hay pérdidas en la eficiencia de la luz a través del sistema, es decir la magnitud es constante, solo habrá cambios de fase, por lo tanto es posible normalizar $|H(u,v)| = 1$ y obtenemos que la función de un POF convencional esta dada por:

$$H_{POF}(u,v) = \frac{T^*(u,v)}{|T(u,v)|} = e^{-j\phi_r(u,v)}, \tag{29}$$

donde $T(u,v)$, $T^*(u,v)$, y $\phi_r(u,v)$ son la transformada de Fourier, su complejo conjugado y la fase de la imagen de referencia, respectivamente.

2.3.1.3 Filtro Inverso (IF, *Inverse Filter*)

El IF minimiza el criterio PCE (*Peek to Correlation Energy*), que es una medida de la angostura del pico. Este filtro produce, en el plano de correlación, un pico más estrecho que los filtros anteriores. Si la imagen de referencia y la imagen de entrada son iguales, se produce un pico tipo función Delta, figura 6. Una desventaja es que produce un alto nivel de ruido de fondo en el plano de salida.

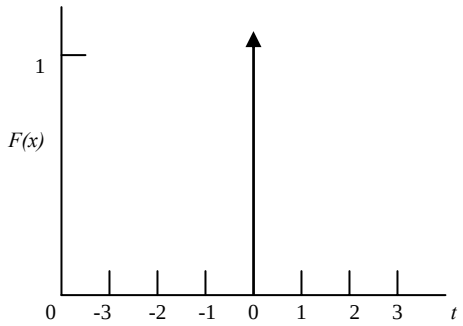


Figura 6.- Representación de la función Delta en t=0.

La ecuación del filtro inverso esta dada por:

$$H(u,v) = \beta \frac{T^*(u,v)}{\left|T(u,v)\right|^2} \equiv H_{IF(u,v)}, \qquad (30)$$

donde β es una constante arbitraria, T (u,v) y $T^*(u,v)$ son las FT de la imagen y su complejo conjugado respectivamente.

2.3.2 Filtros de correlación no lineal

De una manera general se puede expresar un filtro no lineal (NF, *Nonlinear Filter*) de la siguiente manera:

$$NF = \left|F(u,v)\right|^k e^{-j\phi(u,v)}, \qquad 0<k<1 \qquad (31)$$

donde *F(u, v)* es la FT del kernel, *|F(u, v)|* su magnitud y *k* es su nivel de no linealidad (Flores Núñez, *et al,* 2008). Es necesario que el operador no lineal modifique la FT tanto del kernel como de la escena para poder considerar una operación no lineal (Javidi, 1989), los valores intermedios ayudan a variar características como capacidad de discriminación, iluminación.

2.3.2.1 Filtros de correlación no lineales Compuestos

Los filtros compuestos son generados a partir de una cierta cantidad de imágenes del objeto a reconocer, llamadas de entrenamiento. Una característica representativa de este tipo de filtros es su capacidad de ser optimizados para minorizar la sensibilidad a cambios de escala, rotación o iluminación (Seung-Hung y Javidi, 2002) (Javidi, *et al*, 1997). Un filtro compuesto puede definirse como una combinación lineal de distintos filtros de acoplamiento (Casasent, 1984), es decir, sea $\{F_m(u, v)$; m = 1, 2, 3,… N} un conjunto de imágenes de entrenamiento (linealmente independientes) podemos expresar la respuesta al impulso de un filtro no lineal compuesto como:

$$H_{CP} = \sum_{m=1}^{N} \left| F_m(u,v) \right|^{k} e^{-j\phi_m(u,v)}, \quad 0<k<1 \qquad (32)$$

donde N es el número total de imágenes de entrenamiento.

Es importante señalar que la selección apropiada de las imágenes de entrenamiento es la base principal al diseñar un filtro compuesto, una mala selección arrojará malos resultados denotando por lo tanto la mala calidad del filtro.

Existen dos tipos de filtro compuesto, uno donde las imágenes de entrenamiento consideradas en la generación del filtro son variaciones de escala, rotación e iluminación del objeto que se desea reconocer, y otro donde las imágenes de entrenamiento están conformadas por distintos objetos (multi-objeto). La figura 7 muestra un diagrama general de las operaciones de un filtro compuesto, usando como imágenes de entrenamiento varias rotaciones de la letra E.

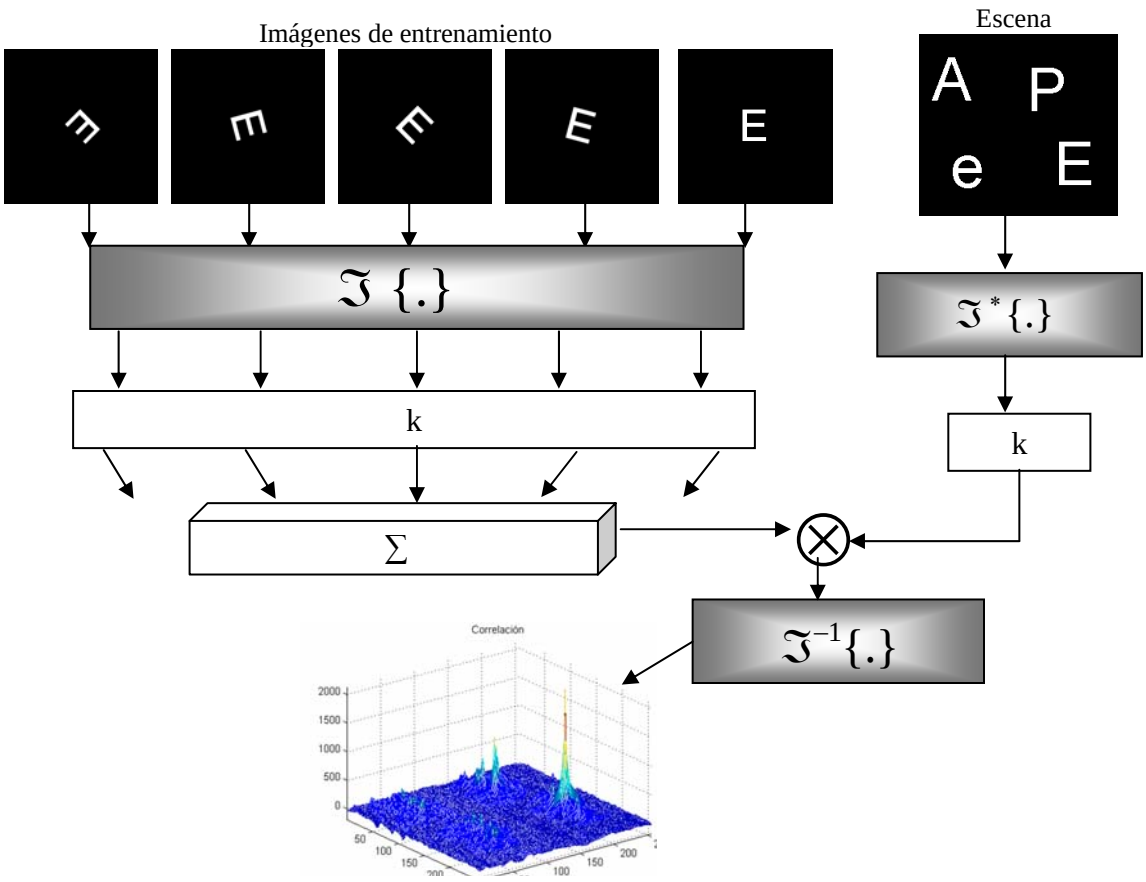


Figura 7. Fabricación y procesado de un filtro compuesto.

2.4 Reconocimiento de Patrones

Gran cantidad de decisiones o interpretaciones se pueden obtener del análisis de la información contenida en una imagen de algún escenario en particular. Casi en su totalidad la mayoría de los objetos que encontramos en el mundo físico son estructuras en tres dimensiones, si se toma una colección de estos únicamente, tendríamos un escenario y desde el punto de vista en dos dimensiones el resultado seria una imagen. Esta escena, siempre se vera afectada por algunas variables en particular:

1. Distancia y ángulo de la superficie del objeto con respecto al origen de la iluminación.
2. Composición espectral de la iluminación.
3. Propiedades de reflexión de los objetos dentro de la escena.

4. Características espectrales, procesamiento y no linealidad de los sistemas de sensores.
5. Contenido de la escena.

La toma de decisiones basada en información contenida en la escena de una imagen entra directamente en el estudio del reconocimiento de patrones (PR, *Pattern Recognition*), disciplina matemática basada en modelos y métodos para la decisión de ciertos procesos. Estos métodos pueden ser aplicados para problemas con la escena y la toma de decisiones basada e su información (Hall, 1979).

El PR de una manera general podemos considerarlo como una clasificación de objetos basada en sus características (Friedman y Kandel, 1999);

2.4.1 Correlación en PR

El termino correlación, matemáticamente se refiere a la relación de una variable con respecto de otra, dos variables tienen un alto valor de correlación cuando los valores de una de ellas varían sistemáticamente con respecto a los valores de la otra, es decir, si tenemos dos variables (A y B), existe correlación si al aumentar los valores de A lo hacen también los de B y viceversa. En el caso del PR no hay variante, consiste en determinar la localización de una imagen template dentro de una escena (Hall, 1979), algunos otros autores prefieren el termino kernel a la imagen a buscar (González y Woods, 2008) o de una manera mas burda imagen filtro (kernel) e imagen problema (escena).

El PR por medio de correlación de variables es una subrama de reconocimiento estadístico de patrones, debido a la búsqueda de similitud entre la señal de referencia o imagen (kernel) y la de la señal u objeto en examinación, se busca la similitud entre estas dos, siendo simples estadísticas basadas en el conocimiento que se tenga del objeto.

Un punto muy importante dentro del reconocimiento por correlación es la posibilidad de obtener resultados independientemente de la posición o tamaño del objeto a reconocer pero sobre todo el uso de la señal completa y no fragmentos de “vecindarios” como se usa en filtros espaciales, no requiere preprocesados de la imagen.

De acuerdo con el teorema de correlación, descrito en la ecuación (21), la correlación entre dos variables se puede calcular tanto en el dominio espacial (haciendo operaciones directamente sobre los valores de las funciones), como en el dominio de Fourier (obteniendo sus FT, multiplicándolas y después obteniendo la transformada inversa del resultado).

En este trabajo de tesis se utilizará la ecuación (21) para obtener la correlación entre dos imágenes, la figura 8 muestra de forma grafica el proceso para obtener la correlación de dos imágenes.

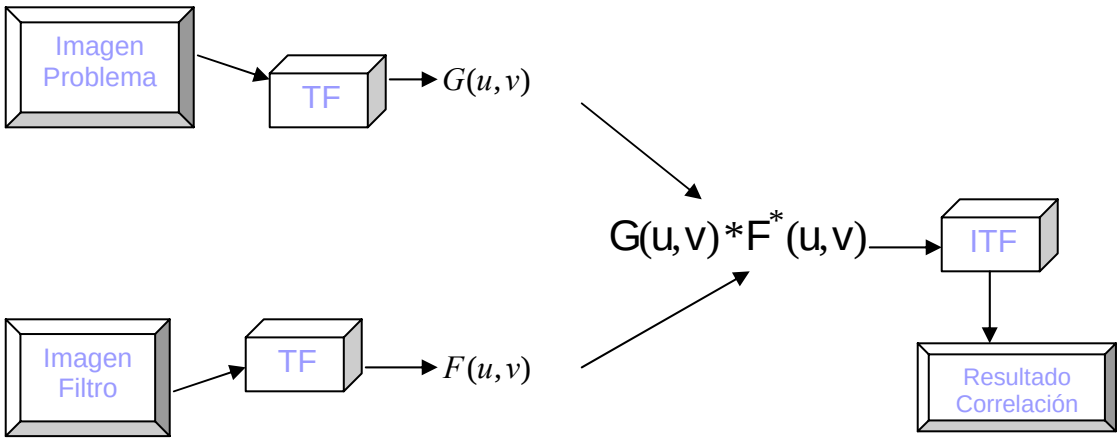


Figura 8.- Diagrama del proceso para obtener la correlación de dos imágenes.

2.4.2 Convolución en PR

La convolución es una operación similar a la correlación, matemáticamente es la forma de combinar dos señales *f* y *g*, que producirán una tercer señal *h*, la cual es vista como una versión modificada de alguna de las dos imágenes originales. Es una operación de gran

importancia en manejo de señales pero también es usada en estadística y por supuesto en procesamiento de imágenes.

La convolución para señales continuas se presenta en la ecuación (13), para el caso de imágenes (señales discretas) su forma seria:

$$h(x,y) = f(x,y) * g(x,y) = \sum_{x'=0}^{M-1}\sum_{y'=0}^{N-1} f(x',y')g(x-x',y-y'),' \qquad (33)$$

donde M y N representan los valores totales en píxeles de la imagen y * denota la convolución.

La diferencia entre la correlación y la convolución como se muestra en las ecuaciones (13), (17) y (18) se encuentra en el conjugado de la segunda función de la ecuación (18), lo que indica que para funciones reales y simétricas las dos operaciones son matemáticamente equivalentes.

2.5 Software Existente

El aumento tecnológico de los últimos años y el crecimiento del interés por el DIP ha permitido ver mas resultados en el campo de reconocimiento de patrones, la capacidad de los computadores actuales permiten mayor cantidad de procesos en muy poco tiempo comparado con 10 años atrás, los medios de almacenamiento tan grandes permiten mayor número de imágenes a trabajar, los lenguajes de programación especializados en el área técnica y científica han facilitado la integración de personal ajeno al campo computacional dentro de sus trabajos de campo o investigaciones, esto y más es cuantificable en los resultados en diversas partes del mundo, tanto en centros de investigación como universidades, donde el objetivo ha sido automatizar en su mejor forma los procesos del DIP.

2.5.1 Internacional

El ámbito internacional es muy amplio y por consiguiente los resultados también, los países en los que más se ha explotado esta área son España, EUA, Inglaterra y Japón, lugares donde el DIP ha sido muy explotado.

2.5.1.1 Mercado

La existencia de productos en el mercado internacional enfocados al DIP no es muy grande pero con los suficientes productos de calidad para dar un buen soporte a esta. Usualmente se centran en el procesado de formas, segmentación de imágenes, detección de bordes, contornos, por mencionar algunos; en otros casos el software es solo una herramienta de ayuda para el hardware. En su mayoría estos productos son enfocados para un tipo de trabajo específico o campo de estudio, en otros casos la diversidad de usos es grande pero por consiguiente la manipulación es bastante complicada. Algunos ejemplos de productos que se aproximan más a la detección de una imagen dentro de otra y no a la evaluación de valores en esta, son:

Infaimon (I1), compañía española que trabaja en visión artificial, encontraremos desde software hasta hardware para este fin, muy interesante y gran cantidad de productos.

Recognition Science (I2), empresa americana enfocada principalmente en aplicaciones biomédicas, al parecer se trabaja sobre pedido pero hay algunas herramientas ya en venta.

En (I3) se encuentran amplia gama de productos para trabajo con imágenes, no cuenta con demos disponibles para valorar el producto.

(I4) es otra página española con software a la venta para el reconocimiento de patrones en una imagen.

Attrasoft (I5) es una compañía con aplicaciones para reconocimiento de patrones, sus aplicaciones son diversas.

Por su parte (I6), comercializa distintos productos, dentro de ellos hay uno para reconocimiento de placas de vehículos con ayuda directa de hardware.

Un excelente proyecto es (I7), trabaja bajo código de consola, así que se deben conocer los comandos. Esté dio pauta para crea una compañía que produce software para reconocimiento de imágenes basados en redes neuronales http://www.spikenet-technology.com/, su enfoque esta en seguridad biométrica. Sus productos son herramientas de desarrollo para algunos lenguajes de programación, no parecen ser ya aplicaciones libres sino una simple herramienta.

2.5.1.2 Instituciones y Centros de Investigación

Los centros de investigación e instituciones de estudios son una parte muy importante en el creciente desarrollo del DIP, muchas de las investigaciones ya sea por parte de estudiantes o de investigadores del área, son la pauta para proyectos más grandes como los mencionados en el apartado anterior.

En *Reconocimiento de objetos en color procesando un sólo canal basándose en el histograma* (Corbalán, *et al*, 2003), se plantea un método de reconocimiento de objetos basado en el histograma de la imagen en los tres campos de color (RGB), el uso de métodos de correlación para reconocimiento de objetos considerando la información de color permite mayor capacidad de discriminación pero el proceso es lento y costoso a nivel computacional, es por eso que en el método se plantea un preprocesado tanto de la imagen a analizar como de la escena donde se encuentra, obteniendo ciertos niveles de gris cuyas frecuencias son superiores a un umbral determinado en los tres campos de color. Esto es, si se denota V_i^T como el número total de píxeles en nivel de gris '*g*' en el canal '*i*' (i=R, G, B) y que el fondo del objeto es negro, es decir *g*=0, entonces el número total de píxeles o área del objeto a reconocer en el canal '*i*' esta dado por la siguiente formula:

$$V_i^T = \sum_{1}^{255} V_i^g \ . \qquad (34)$$

El preprocesado que se propone consiste en ordenar en orden decreciente los valores de V_i^g en cada canal generando una secuencia numérica $(V_i^g)_j$, así el primer valor de esta secuencia corresponde a la moda del histograma, a este valor se le añade el siguiente $(V_i^g)_2$ hasta que el último exceda un porcentaje dado del total del área del objeto a reconocer en ese canal *i*; se determina un umbral *U* que corresponde a un porcentaje del área del objeto a reconocer. Finalmente se obtienen los niveles de gris N_i mediante la formula:

$$\sum_{j=1}^{N_i} (V_i^g)_j \geq U.V_i^T \qquad (35)$$

Donde para cada canal '*i*' se aplica una binarización píxel a píxel en cada canal y se genera una nueva imagen comparando cada resultado de la binarización mediante la operación OR entre los 3 canales, obteniendo los valores a correlacionar con el fondo que pasa también por el mismo proceso.

Los resultados planteados aquí son favorables tanto en reconocimiento como en costo computacional, pero la asignación manual del umbral puede determinar ciertos errores de calidad en la identificación, la presencia de imágenes muy coloridas pueden ser muy bien aceptadas en este proceso y el procesado del fondo genera a largo plazo costo computacional.

López en *Implementación multiplataforma de algoritmos de procesamiento de imágenes biomédicas 2D en MatLab y C++* (López, 2003), se enfoca totalmente en imágenes biomédicas específicamente imágenes de ADN, con la finalidad de determinar el daño en este por la radiación ionizante (cometas), permitiendo valorar y cuantificar el daño en distintos niveles de radiación. Su base matemática para el procesado de las imágenes de los COMETAS se rige mediante los llamados contornos activos, que es una técnica de segmentación de imágenes muy robusta y por tanto requiere más tiempo de cálculo. Estos contornos modelan las fronteras entre el objeto, el fondo y el resto de objetos de la imagen, permitiendo extraer los contornos del objeto de interés basándose en modelos que utilizan información a priori de la forma de los objetos.

Basados en la técnica anterior se hacen algunos algoritmos en MatLab para extraer desde tamaño de la cabeza del COMETA hasta su centro, posteriormente se genera una Interfaz Grafica de Usuario (GUI, *Graphic User Interface*) en este mismo para la interacción con el usuario, estas son acopladas directamente por el director de tesis para que trabajen en conjunto.

Finalmente los algoritmos son compilados con el MatLab Compiler generando código C para un uso posterior en Visual C++, en este aspecto se menciona la compilación para consola y no para entorno visual, en capítulos finales se habla sobre la posibilidad de hacer la aplicación independiente pero no se ven imágenes o comentarios de algo realmente hecho.

El método utilizado es muy robusto y por tanto genera mucho costo computacional, el uso de GUI's en MatLab es bueno como herramienta de muestra pero no para aplicaciones independientes, después de compilado el código era necesaria la inserción de un path como referencia a una librería de MatLab, es una muestra clara del objetivo de esta tesis con sus respectivas diferencias en bases matemáticas, enfoque y herramientas a utilizar.

Una tesis significativa por algunos aspectos pero un poco confusa con respecto de su titulo y su contenido es *Análisis de imágenes biomédicas para la cuantificación del nivel de fragmentación de los espermatozoides humanos y de la carga vírica de la hepatitis C* (Santos, 2005), donde el objeto de estudio esta basado casi en su totalidad en algunas otras tesis, se mencionan otras herramientas para generar aplicaciones independientes tales como: instaladores, graficadotes, librerías de ayuda, programas para generar dependencias UML, entre otros. Lo importante en este caso de estudio es la implementación que se hizo de dos aplicaciones que estaban desarrolladas en MatLab y fueron pasadas a un entorno .NET, donde su objetivo es determinar algunos parámetros de espermatozoides en base al análisis de la imagen de estos. Como en otros casos el análisis de la imagen es en base a contornos y forma principalmente, pero no a una correlación de un objeto con otro como es el caso de este trabajo de investigación.

2.5.2 Nacional

El desarrollo de un país es claramente reflejado gracias a la cantidad y calidad de investigación que se desarrolla en el, México ha comenzado ya hace un par de años en adentrarse en el campo del DIP, fruto de ello son diversos centros de investigación con interés en diversas áreas, tal es el caso del Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOP) en Puebla, donde su campo de estudio es "el reconocimiento lógico combinatorio de patrones, reconocimiento del habla y llanto de bebes, aprendizaje automático en el reconocimiento de patrones, entre otros"; el Centro de Investigación Científica y de Estudios Superiores de Ensenada (CICESE) enfocado al "Análisis de imágenes y percepción remota, desarrollo de algoritmos de correlación invariantes para el reconocimiento de partículas biogénicas, etc."; el Centro de Investigación en Computación (CIC) con "Análisis de retinas humanas, de imágenes para el control de sistemas electromecánicos, etc." como líneas de estudio; "el reconocimiento de patrones, visión artificial y visión robótica" en el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) en Cuernavaca.

Dentro de este entorno nacional algunas referencias importantes que encierran el DIP y por consiguiente este trabajo de tesis. En la transición de métodos ópticos a medios digitales en el reconocimiento de imágenes es relevante mencionar el trabajo de Zavala-Hamz y Álvarez-Borrego (1997), *Técnicas digitales invariantes a rotación para el reconocimiento de copépodos calanoides*, donde se reconoció algunas especies de copépodos a nivel género sin importar la orientación del organismo, esto con el desarrollo de un algoritmo para crear un filtro armónico circular (FAC), pasando desde la obtención de la imagen en formato digital hasta su procesado con este FAC; la parte que resalta de este trabajo es que el desarrollo se hizo por completo en código de Fortran generando poco más de 1000 líneas de código dentro de su programa principal como de sus funciones auxiliares, ejemplo del camino por el que se envolvería el DIP y por consiguiente el PR.

2.6 Sumario

Como se ha visto a lo largo de este capítulo, los resultados dentro del área que compete a este trabajo de tesis son pocos por la diversidad de formas existentes para poder reconocer un objeto (redes neuronales, filtros espaciales, cálculos estadísticos, algoritmos genéticos), es difícil poder delimitar una comparativa entre lo existente y lo que se desea generar principalmente a que no hay relación o similitud cercana entre lo documentado para estos fines.

La tabla 1 muestra algunas características específicas de lo existente y de la aplicación a desarrollar.

Tabla 1.- Comparativa de sistemas existentes.

	Digital/Óptico	PR	PORTABILIDAD	COSTO COMPUTACIONAL	OBJETO DE ESTUDIO
Corbalán *et al*, 2003	Digital	Si	No	Si	Imágenes
López, 2003	Digital	No	Si	-	Imágenes medicas
Santos, 2005	Digital	No	Si	-	Imágenes medicas
Zavala-Hamz y Álvarez-Borrego, 1997	Digital	Si	No	Si	Imágenes microorganismos
SISREC	Digital	Si	Si	Si	Imágenes cualquiera

CAPÍTULO 3

METODOLOGÍA

El procesado de imágenes e identificación de objetos mediante técnicas de correlación actualmente está apoyado por programas de carácter científico como lo es MatLab, debido a su velocidad de procesado en ecuaciones matemáticas; es por esta razón que el desarrollo de esta tesis abarca no solo la programación dentro de nuestro lenguaje de alto nivel sino también dentro de MatLab.

En la actualidad la documentación en el desarrollo de software ha cambiado de una forma radical al surgir las llamadas “metodologías ágiles”, las cuales se centran en cuatro puntos importantes (I8):

1. **Individuos e interacciones** sobre *procesos y herramientas.*
2. **Software que funciona** sobre *documentación exhaustiva.*
3. **Colaboración con el cliente** sobre *negociación de contratos.*
4. **Responder ante el cambio** sobre *seguimiento de un plan.*

En ellas, aunque los factores de la derecha son de gran valor, aquí son de mayor peso los de la izquierda, razón por la cual, veremos una escasa documentación debido a la interacción cliente-diseñador-programador a lo largo del desarrollo; una de las metodologías ágiles de más renombre es la llamada XP (eXtreme Programming) (I9), caracterizada por desarrollo en pequeños módulos seguidos de pruebas y posteriormente la unificación de estos en un sistema completo, todo esto permitiendo la retroalimentación de cualquier módulo en cualquier momento.

Trabajar con dos lenguajes de programación distintos, no es tarea fácil. La exitosa interacción de un lenguaje con el otro depende de la calidad de desarrollo y el trabajo en pequeños módulos, la retroalimentación continua, hace posible conseguir esto.

La figura 9 muestra el diagrama de flujo, del procedimiento general utilizado para obtener una aplicación.

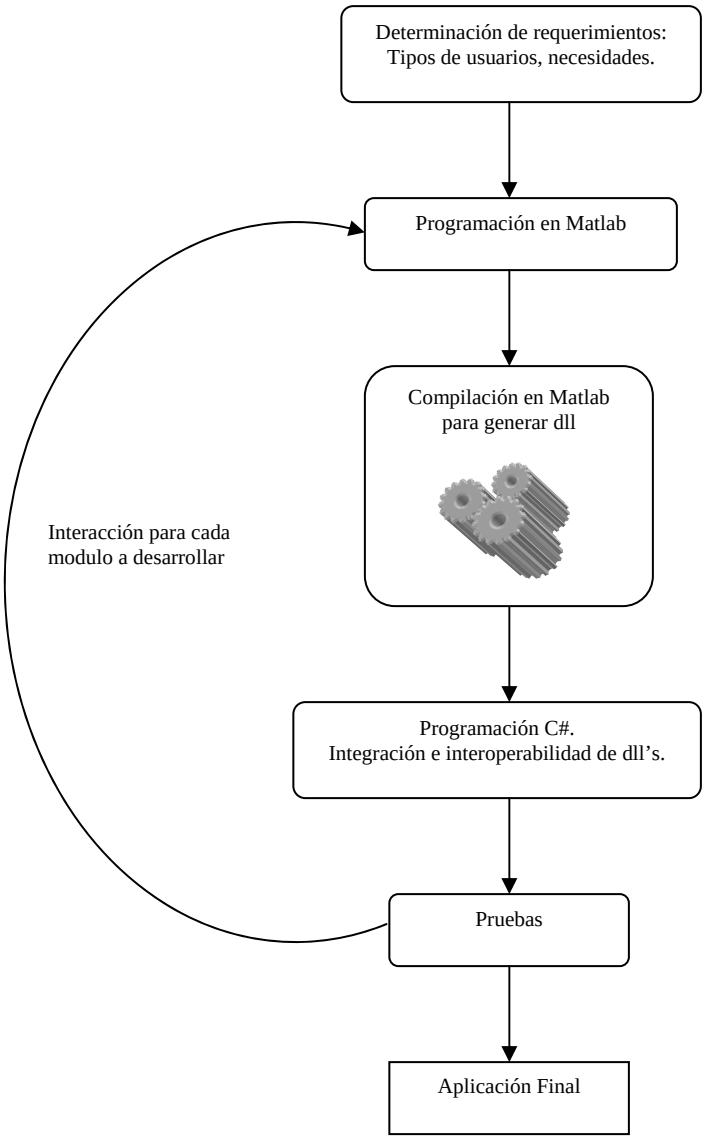


Figura 9.- Procedimiento general utilizado para obtener la aplicación.

3.1 Determinación de requerimientos.

3.1.1 Tipos de usuarios y necesidades.

El objetivo principal de un trabajo de investigación es darle respuesta a necesidades de determinados tipos de usuarios, este es el fin de este procedimiento, determinar los usuarios y sus necesidades para darle forma y guía al desarrollo de la aplicación, Sistema de Reconocimiento (SISREC), como se le llamará de aquí en adelante.

Aun cuando el SISREC está enfocado para todo tipo de usuario que requiera hacer reconocimiento de imágenes sin saber la base matemática de esto, fue necesario centrarse en una población específica que al hacer uso de él agilizaría su trabajo de una forma sustancial.

Los usuarios potenciales son hallados en centros de investigación científica, en este caso se eligieron investigadores y profesores de la Facultad de Ciencias Marinas, la Facultad de Ciencias, ambas de la Universidad Autónoma de Baja California (UABC), Unidad Ensenada, el Departamento de Oceanografía Física del Centro de Investigación Científica y de Educación Superior de Ensenada (CICESE) y el Laboratorio de Zoología Acuática de la Facultad de Estudios Superiores Iztacala perteneciente a la Universidad Autónoma de México (UNAM). Se elaboró una encuesta en un documento de Word con cuadros de controles para poder ser llenado sin prisa alguna en envíos de formularios. Posteriormente la encuesta fue enviada por correo electrónico a los investigadores después de una plática previa y, una vez contestada, fue devuelta utilizando el mismo medio.

Es necesario aclarar que la encuesta de opinión no se basó en estudios estadísticos debido a la naturaleza de la tesis. La encuesta en cuestión se encuentra en el Apéndice A de este trabajo.

3.1.2 Requerimientos técnicos.

Aun cuando la tecnología se encuentra avanzando a pasos agigantados se determinaron los requerimientos mínimos en base a lo observado en la encuesta de opinión y a lo existente en el mercado. De una manera global el SISREC trabaja en una sola estación en sistemas x86, ya sea de 32 bits (XPSp2, Vista) o 64 bits (Vista); su desarrollo está basado por un lado en tecnología .NET de Microsoft© haciendo uso del Component Framework 2.0 y por otro MathWorks© con su Matlab Component Runtime 7.6 (MCR), trabaja con imágenes de mapas de bits, específicamente "bmp", desde 256x256 píxeles de resolución tanto en blanco y negro como en color, maneja diferentes tipos de filtros a escoger por el usuario, los resultados tienen la capacidad de ser guardados en estándar JPEG o en formato BMP.

En la figura 10 se muestra un diagrama general del flujo de datos en SISREC, comenzando con la obtención de imágenes por algún medio electrónico como cámara digital o microscopio (a), seleccionadas y cargadas a SISREC donde comienza la interacción de módulos entre .Net, dll generada y el MCR (b) para finalizar con los resultados y la posibilidad de imprimirlos o guardarlos en formato electrónico (c).

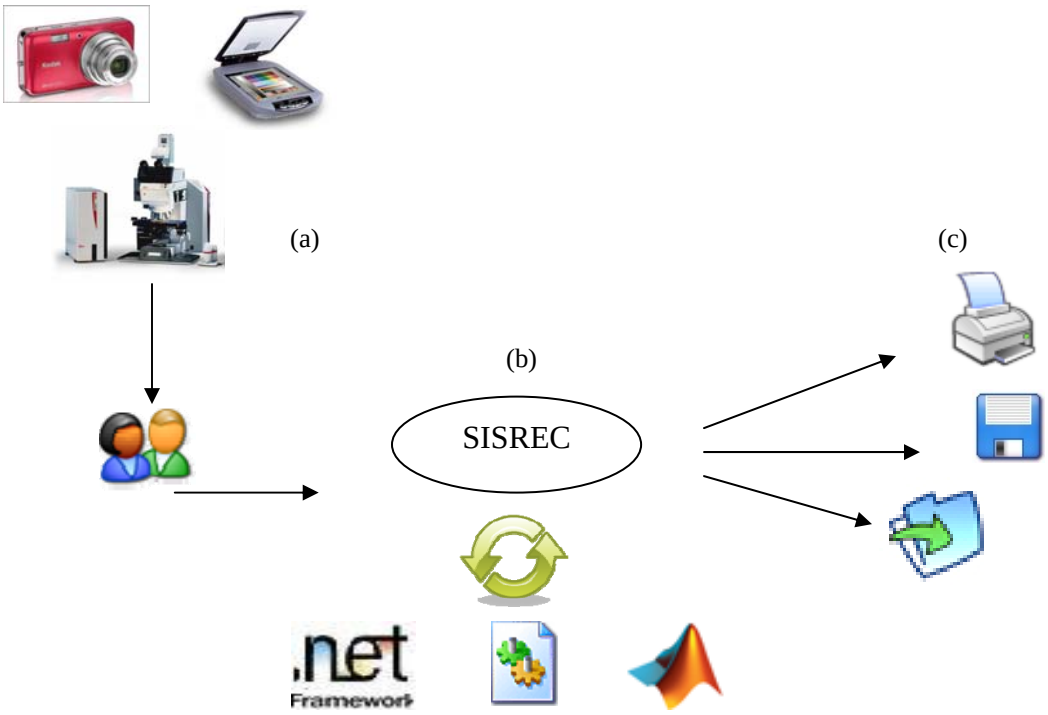
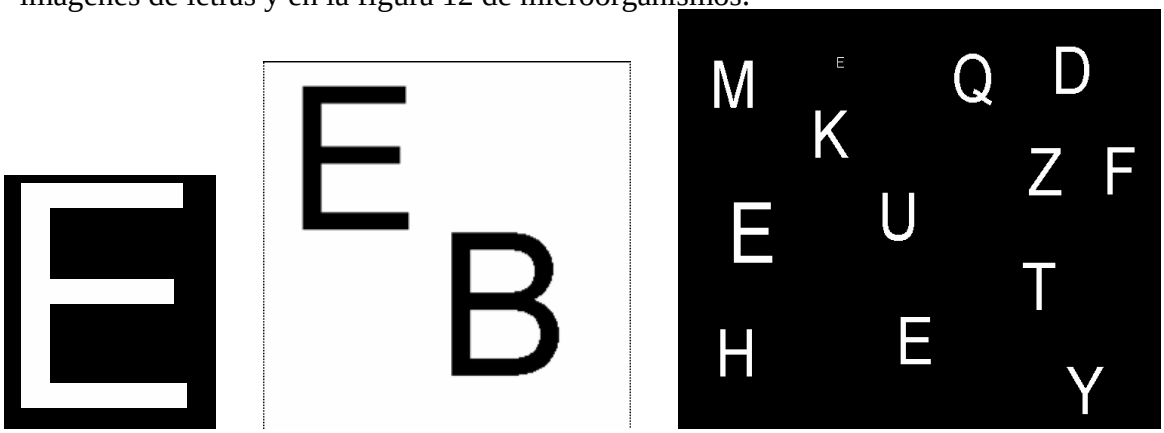


Figura 10.- Diagrama de trabajo de SISREC.

3.2 Obtención de imágenes.

Antes de poder trabajar con imágenes reales fue necesario generar imágenes sencillas para realizar pruebas preliminares del SISREC, para ello se dibujaron en Microsoft© Paint algunas letras “E” que servirían como filtro o kernel en las pruebas. Además se generaron imágenes problema o escena con letras “E” acompañadas de otras letras como la “B”,”F”, etc., todas ellas de diferentes resoluciones.

Se obtuvieron algunas imágenes de microorganismos del Grupo de Procesamiento de Imágenes del Departamento de Óptica del CICESE, a estas fue necesario trabajarlas para sacar de ellas mismas imágenes problemas o kernel, ya que, en algunos casos la muestra era de un solo microorganismo y en algunas otras de varios. La figura 11 muestra un grupo de imágenes de letras y en la figura 12 de microorganismos.



(a) (b) (c)

Figura 11.- Imágenes de letras diseñadas en Paint. kernel “E” (a), escena “EB” (b) y escena “multiletra” (c).

(a) (b) (c)

Figura 12.- Imágenes de microorganismos. Algunas son kernel (a) y otras escenas (b) y (c).

3.3 Matlab

La programación en Matlab difiere de los lenguajes de programación conocidos en algunos detalles, siendo los más importantes son:

1) No genera archivos ejecutables.
2) El código puede ser escrito directo en consola y ejecutarlo.
3) Los programas son de dos tipos script o funciones, en los primeros no hay recepción ni envío de variables a otros programas o funciones externas, en los segundos si se reciben y envían datos.
4) Su enfoque es 100% matemático por ello todo se trabaja en forma de matrices o vectores.

Conocidas estas diferencias se realizaron las primeras pruebas en forma de script con la finalidad de corroborar el funcionamiento de los filtros con cada una de las imágenes problema.

Debido a los requerimientos del compilador de MatLab para poder generar una librería e interactuar con .Net, el script se separó en funciones, que son los métodos principales sobre las cuales trabaja SISREC, el código fuente de estos se encuentran en el apéndice B.

Los métodos principales son:

- LoadImageFil_v2
  - Carga imágenes kernel para procesar.

- LoadImageProb_v2
  - Carga escenas para procesar.

- MainFft_v3
  - Raíz de transformadas de Fourier y algunas operaciones de filtros.

- MainFftNon_v3
  - Raíz de transformadas de Fourier y algunas operaciones del filtro compuesto.

- MatrizFusion
  - Centrado del kernel (Véase 3.3.1).

- ValMaxImag
  - Ubicación del valor máximo del pico de correlación.

- OperCorrMulti_v23
  - Manejo de datos de correlación, busqueda de numerosos picos en cada una de las imágenes.

- CreateFigureCorr
  - Creación de figuras de resultados.

- SumEspecFilNon_v11
  - Encargada de sumar los espectros pertenecientes a cada imagen filtro y regresar una suma del mismo.

El código perteneciente a cada uno de estos métodos se encuentra disponible en el apéndice B. Se utilizaron algunos más pero estos son los principales y mas importantes dentro de las operaciones de correlación.

3.3.1 Problemática de centrado

Debido a que el usuario no genera sus imágenes como se hizo en las pruebas con las letras, se tuvo que idear una forma de poder centrar el kernel de forma manual sin perder información en el proceso. Ya que la correlación está dada sobre dos matrices del mismo tamaño y porque el filtro debe ser de menor tamaño a la escena (González y Woods, 2008) es posible que el usuario tenga imágenes kernel mal centradas y ello propiciaría errores en la identificación, se generó un algoritmo en MatLab que centra nuestro kernel dentro de un fondo sin información (ceros) del tamaño de la escena donde se está comparando. En la figura 13 se muestran algunas imágenes kernel mal centradas y una óptima para usar.

(a) (b) (c)

Figura 13.- Imágenes kernel. Kernel listo para centrar (a), kernel mal centrado (b) y kernel centrado manualmente con riesgo a errores (c).

Este algoritmo es determinante en el tiempo que tarde la identificación porque genera en tiempo de ejecución, una imagen nueva del tamaño de la escena y posteriormente es fusionada a el kernel a ella píxel a píxel, esto es, entre más grande sea nuestra escena mayor será tiempo de procesado (256x256, 1027x768, 1280x1400, etc.), por ello siempre es recomendable el uso de imágenes de 256x256, aun así SISREC es capaz de trabajar con imágenes de MxN dimensiones.

El procedimiento para fusionar el filtro con el fondo se muestra en la figura 14 y 15, primero se toma el kernel (K) y se obtienen sus dimensiones de alto y ancho (figura 14)

posteriormente se saca un punto medio en renglones y columnas, llamado centro del kernel. Se toman las dimensiones de la escena (S) para obtener al igual que en el kernel, un centro de escena. Éste centro será el mismo en las dos imágenes, una vez que se fusionen, y lo llamaremos centro absoluto. Posteriormente genera, en tiempo de ejecución, la imagen de fondo de las dimensiones de la escena (Sf), se realizan cálculos matemáticos entre los puntos medios del kernel y la escena para encontrar el centro absoluto y que el centrado sea correcto (figura 15).

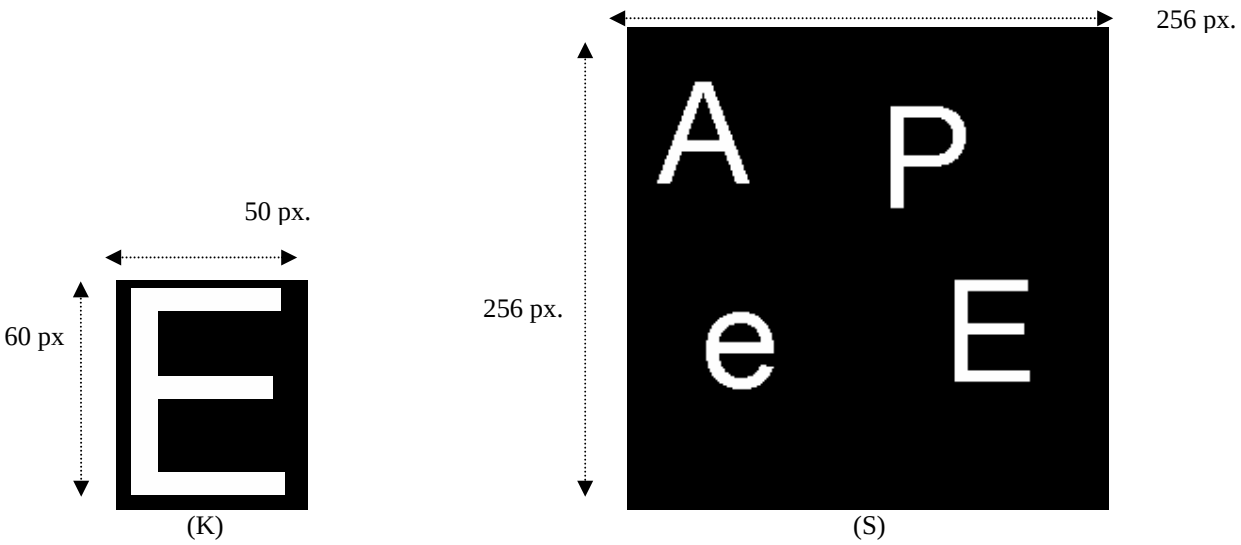


Figura 14.- Dimensiones de las imágenes. Kernel de 60x50 (K) y escena de 256x256 (S).

Finalmente, en tiempo de ejecución, se pinta el resultado recorriendo píxel a píxel el fondo (i, j) hasta localizar la coordenada (Mi, Ni) que se obtiene del punto medio de los renglones de la escena y el kernel, ahí comienza a colocarse el kernel hasta encontrar la coordenada (Mi, Nf), dada por los puntos medios de las columnas de la escena y el kernel, para continuar únicamente con el fondo. Se prosigue con interacciones píxel a píxel incrementando los respectivos valores en cada una hasta llegar a la coordenada (Mf, Ni) y (Mf, Nf) que determinará el alto y ancho máximo respectivamente del kernel fusionado en el fondo, dando como resultado una nueva imagen a partir de una fusión de dos.

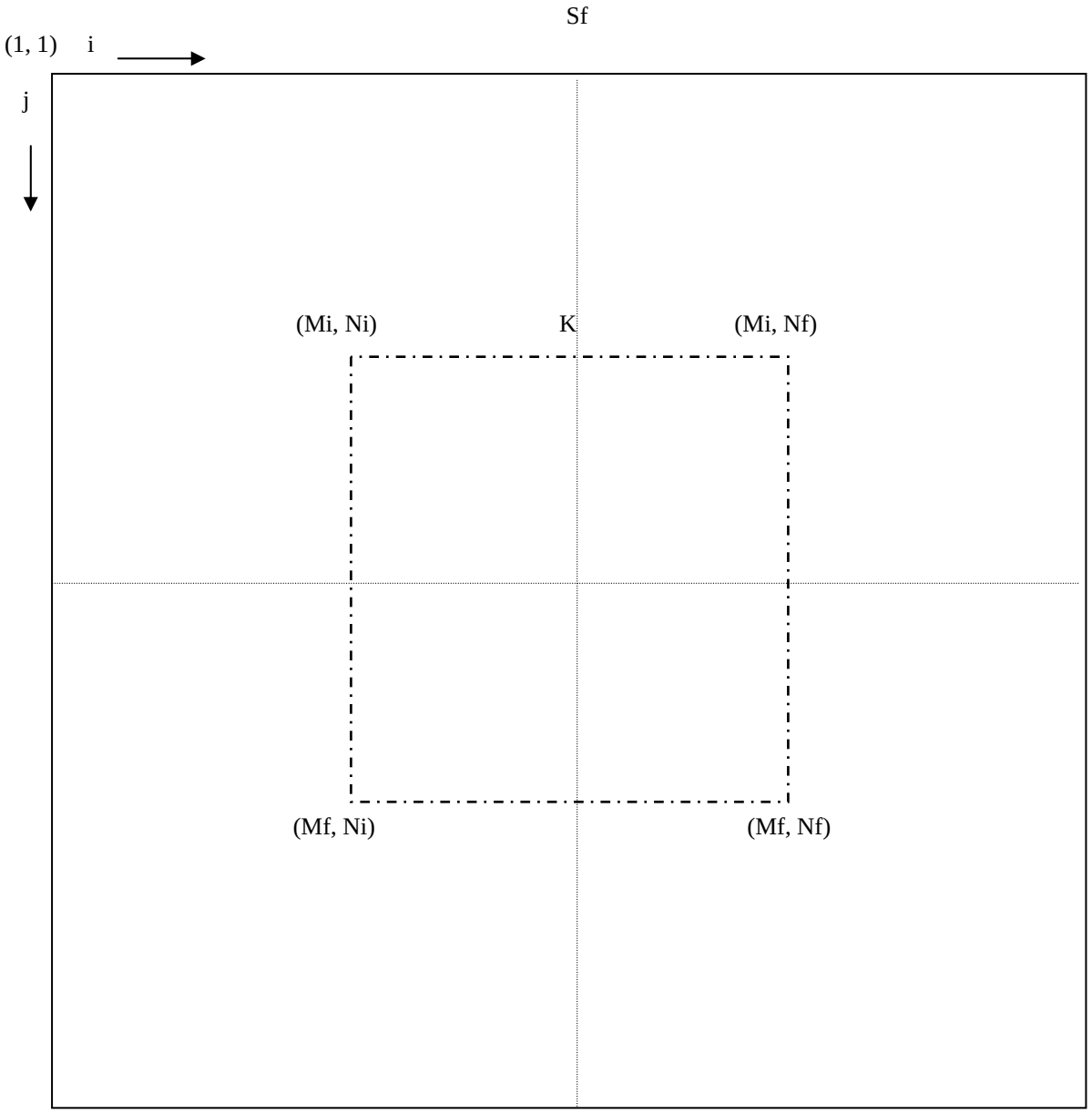


Figura 15.- Fondo fusionado con kernel. Fondo (s) del tamaño de la escena y kernel (k) centrado en ella píxel a píxel.

El resultado de este algoritmo ya depurado es el método MatrizFusion que se encuentra en el apéndice B.

3.3.2 Problemática de la imagen impar

Es muy común encontrar imágenes de resoluciones impares, es decir imágenes que sus resoluciones no sean divisibles entre 2. Por ejemplo, valores en resolución de 126x255, 255x255, 512x825 que tienen por lo menos una de sus dimensiones con valores impares. El trabajo en esta tesis es controlado para evitar este tipo de problemas, pero en el mundo real estas imágenes están presentes y por ello se implementó un código para evaluar si existe resolución impar y si así fuera, corregir la irregularidad en tamaño, permitiendo poder hacer los cálculos correctos al momento de la fusión y no tener divisiones entre 0.

Al recibir una imagen de dimensiones impares, SISREC valorará cuál lado no contiene la suficiente información y lo recortará sin afectar el contenido de la imagen (ya sea kernel o escena). Si la imagen es de 126x255, SISREC buscará dentro de los renglones finales el que contenga menos información y será omitido al momento de los cálculos. Si la imagen es de 512*825, entonces juzgará entre los valores de las columnas; en la figura 16 y 17 se muestran dos ejemplos de imágenes del mundo real con resolución impar y la ubicación de su posible recorte.

Figura 16.- Imagen de la escuela normal de Austria con resolución 400X299. Recorte en las columnas.

Figura 17.- Imagen de frutero de 477x498 píxeles. Recorte en los renglones.

Cuando se habla de cortar un renglón o columna de píxeles es posible pensar en perdida de información, pero como se puede ver en las figuras anteriores, esta es casi nula e insignificante para la correlación.

3.4 Generación de dll.

MatLab cuenta con la opción de generar archivos compilados a través del MatLab Compiler, que trabaja de forma general, de dos maneras: 1) Generando archivos .MEX, que son funciones en lenguaje C escritas directamente en MatLab o en C y usadas en MatLab para generar librerías externas para algunas otras funciones (*.h,*.lib). Su función depende directamente de la existencia de MatLab en el equipo donde se utilicen. 2) Generando archivos EX, COM, NET o JAV, que son librerías (*.dll) especificas, según la aplicación donde se usarán, dependientes únicamente de ciertas rutinas de MathWorks© y que

generarán aplicaciones independientes. En la figura 18 se muestra un diagrama de las opciones de trabajo del MatLab Compiler.

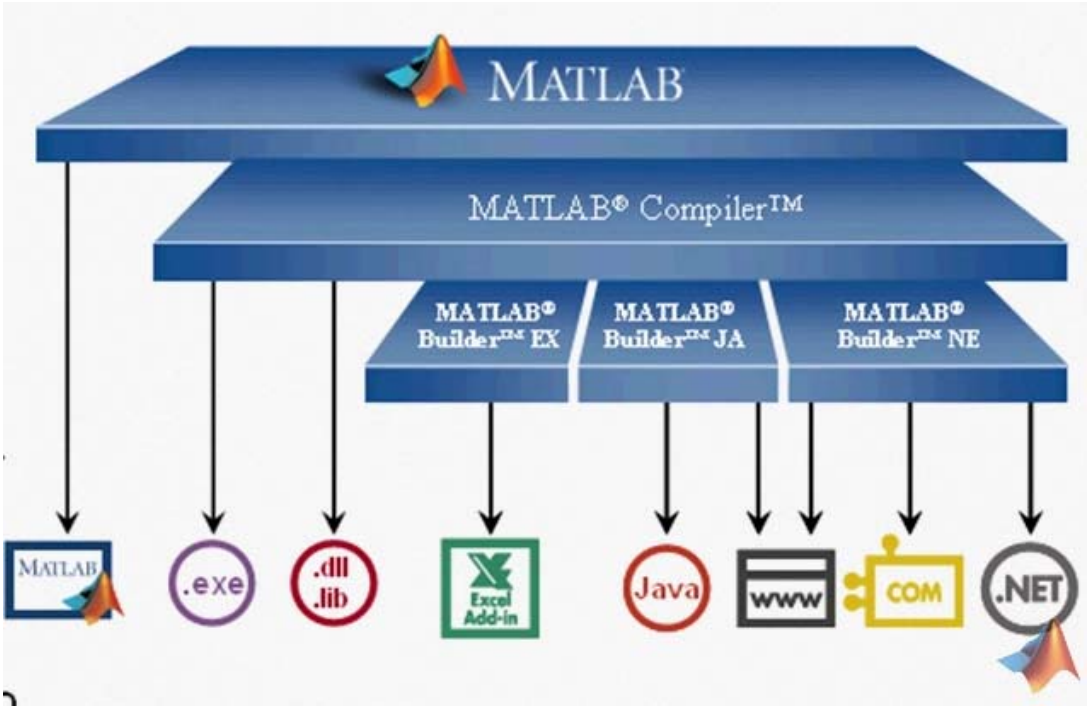


Figura 18.- Diagrama de variantes del MatLab Compiler y sus herramientas.

En esta tesis se trabaja sobre las segundas y se hace uso de la herramienta MatLab Builder NE para su compilado. La librería generada, MatlabCompV23.dll contiene las funciones de MatLab desarrolladas y comentadas en la sección 3.3 y a la cual se le llamará COMPONENTE.

La compilación genera un componente .Net constituido por una clase llamada MyFunc, que a su vez estará conformada por las funciones hechas en MatLab, que al ya ser parte ya de la clase MyFunc se denotarán como METODOS de la clase.

Para obtener un componente funcional es necesaria la debida configuración dentro del compilador, todo con base en las necesidades de la aplicación y a las herramientas usadas en nuestras funciones. En el caso particular del componente de SISREC las configuraciones

importantes están en el tipo de framework a utilizar, las opciones de debug, la generación del MCR para el cliente o equipo de desarrollo (en caso de no ser el mismo) y la más importante las toolbox usadas para auxilio de nuestras funciones. La siguiente serie de figuras (19-23) muestra estos puntos importantes de la configuración.

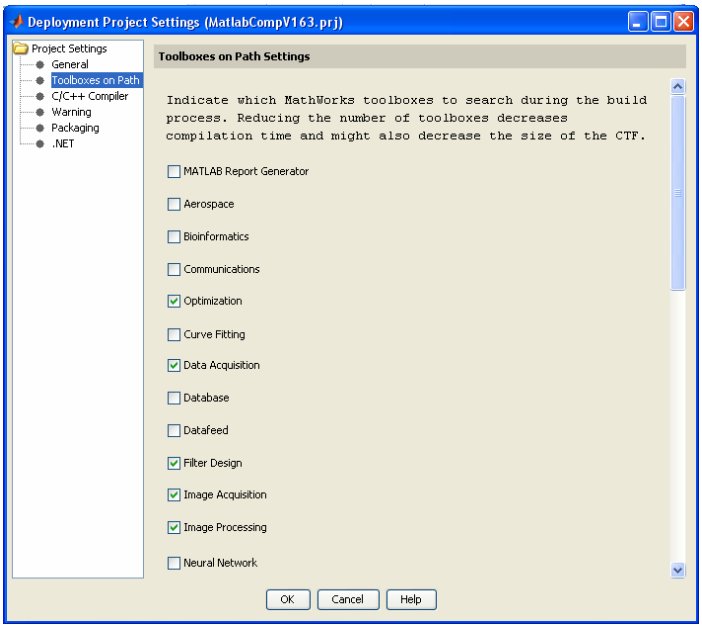


Figura 19.-Lista de toolbox de Matlab. Marcadas únicamente utilizadás en nuestras funciones, en este caso Optimization, Data Acquisition, Image Acquisition e Image Processing.

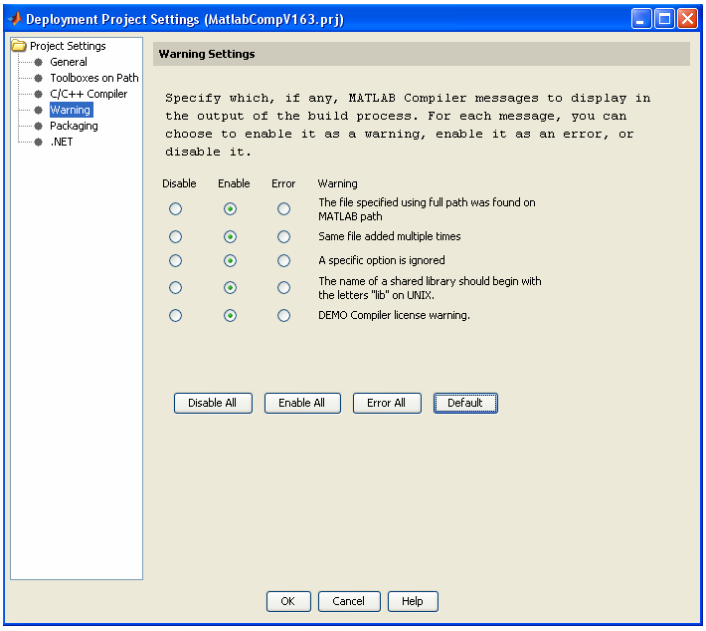


Figura 20.- Muestreo de alertas. Habilitar o deshabilitar los mensajes de precaución dentro de la compilación.

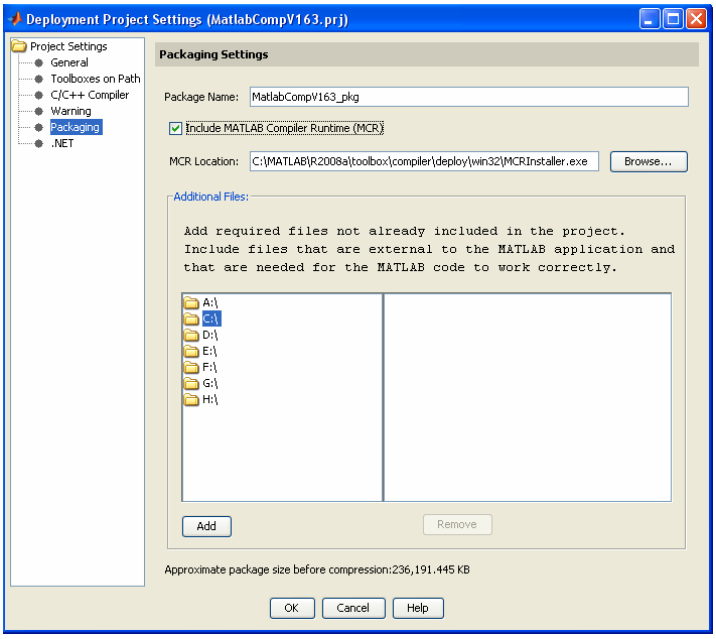


Figura 21.- Empaquetado. Genera el MCR correspondiente al componente, ya sea para desarrollar en otra estación o añadirlo a la versión distribuible.

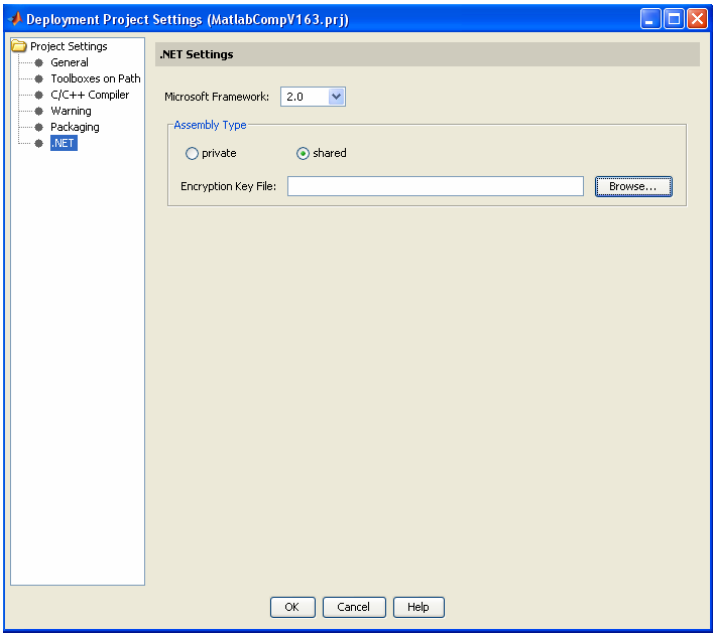


Figura 22.- Opciones .Net. Selección de framework a utilizar y tipo de componente a generar, compartido o público.

Finalmente la opción más importante se muestra en la figura 23. Esta permite la posibilidad de hacer un semi-debug dentro de nuestro componente, en tiempo de ejecución, al momento de utilizarlo en el desarrollo dentro de .Net; no afecta si no es marcado, pero no marcarlo haría que la detección de errores dentro del componente fuera mas complicada al no saber con exactitud la línea de error.

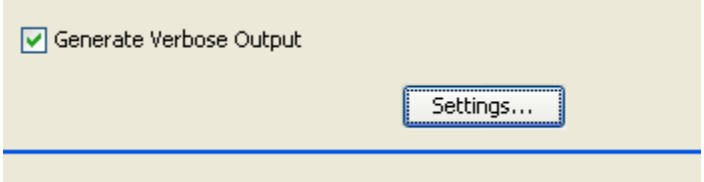


Figura 23.- Opción para debug interno en tiempo de ejecución.

3.5 Desarrollo de la aplicación: .Net

3.5.1 Diferencia de datos nativos (.Net) y MWArrays (MatLab)

Debido a que MatLab trabaja todos sus datos en forma de vectores es necesaria cierta conversión entre una variable de tipo de dato nativo de .Net (int, string, double, etc.) a uno del tipo necesario en nuestro componente, en este caso son llamados MWArrays (MWNumericArray, MWArray, etc.). En la tabla 2 aparecen algunas de las variables usadas dentro de SISREC, tanto en formato nativo como su homóloga en MWArray.

Tabla 2.- Variables en SISREC.

MWArray	Nativo
`MWArray pathFilMat`	`string pathFilNet`
`MWArray pathProbMat`	`string pathProbNet`
`MWNumericArray opFilMat`	`int opFilNet`
`MWNumericArray kMat`	`double kNet`
`MWNumericArray altoFilMat`	`int altoFilNet`
`MWNumericArray anchoFilMat`	`int anchoFilNet`
`MWNumericArray modAltoFilMat`	`int modAltoFilNet`

`MWNumericArray modAnchoFilMat`	`int modAnchoFilNet`
`MWNumericArray altoProbMat`	`int altoProbNet`
`MWNumericArray anchoProbMat`	`int anchoProbNet`
`MWNumericArray modAltoProbMat`	`int modAltoProbNet`
`MWNumericArray modAnchoProbMat`	`int modAnchoProbNet`
`MWNumericArray centAltoProbMat`	`double centAltoProbNet`
`MWNumericArray centAnchoProbMat`	`double centAnchoProbNet`
`MWNumericArray centAltoFilMat`	`double centAltoFilNet`
`MWNumericArray centAnchoFilMat`	`double centAnchoFilNet`
`MWNumericArray MiMat`	`int MiNet`
`MWNumericArray MfMat`	`int MfNet`
`MWNumericArray NiMat`	`int NiNet`
`MWNumericArray NfMat`	`int NfNet`
`MWNumericArray corrLoadMat`	`double[,] matrizCorrNet`
`MWNumericArray iMat`	`int iNet`
`MWNumericArray jMat`	`int jNet`
`MWNumericArray tmpM`	`double[,] tmpN`
`MWNumericArray cooXMat`	`int cooXNet`
`MWNumericArray cooYMat`	`int cooYNet`

Es importante mencionar que MWArray es una librería de clases, por consiguiente está conformada por un espacio de nombres (namespace) y éste a su vez por clases con cierta jerarquía, métodos, miembros y con la capacidad de indexar tipos de datos, detectar errores y por supuesto formatear de datos. Esto implica que el formateo de datos entre nativo y MWArray no sea implícito (int = MWNumericArray), por lo que es necesario respetar las propiedades y características de esa clase en particular. En el apéndice C se anexa toda la documentación original de la librería de clases MWArray disponible únicamente en línea en el sitio de MathWorks.

3.5.2 **Pantallas**

SISREC maneja una interfaz sencilla y compacta, únicamente con las funciones necesarias para hacer de la identificación un trabajo fácil y por demás sencillo.

La figura 24, muestra la pantalla principal de SISREC, donde el usuario cargará las imágenes kernel, escena y seleccionará el tipo de filtro a usar. Muestra también la ubicación de cada imagen y sus respectivas dimensiones.

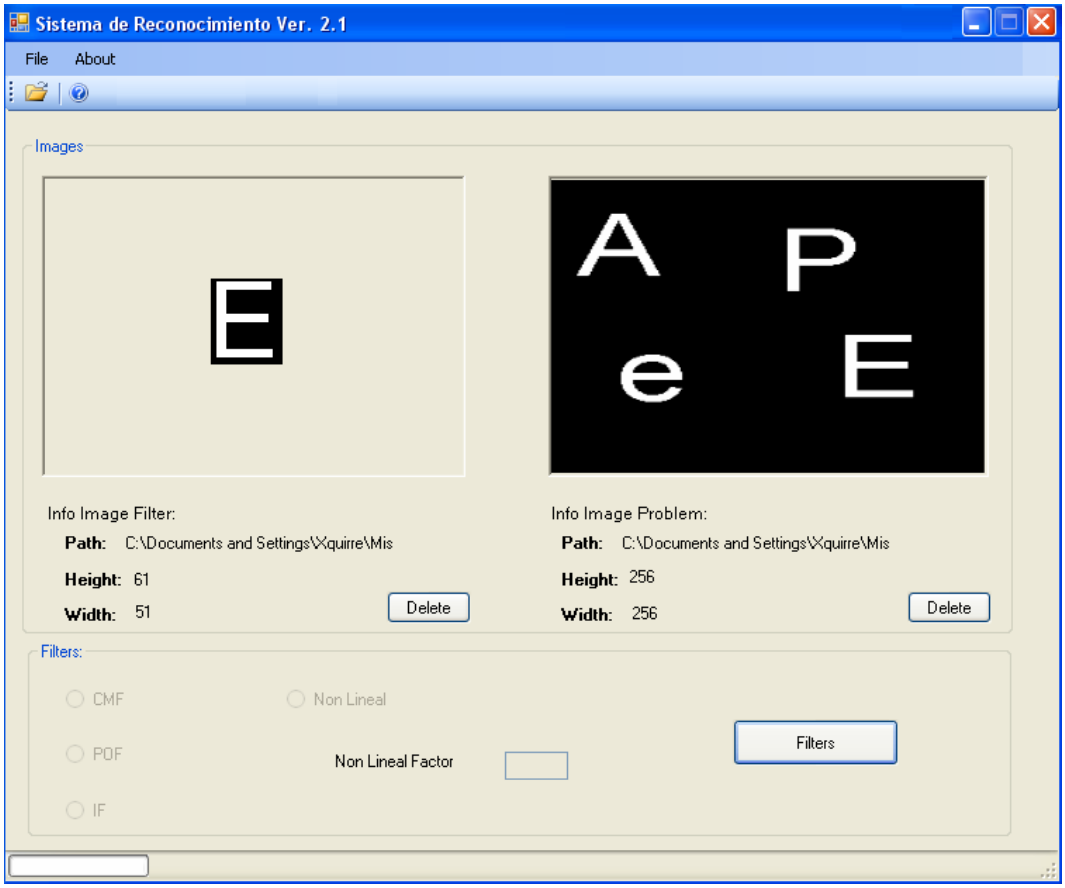


Figura 24.- Pantalla principal de SISREC.

Los menús y submenús existentes en la pantalla principal son:

- File
  - Open

- Filter: seleccionar de una ubicación la imagen kernel, su shortcut key es F2.
- Problem: seleccionar de una ubicación la escena, su shortcut key es F3.

➢ Exit: salir de SISREC, CTL+ F4.

- About: información sobre el desarrollador.

La pantalla de resultados se observa en la figura 25, donde el usuario podrá ver los resultados de la correlación entre sus imágenes tanto en la misma imagen marcada por un punto dentro de ella, como en forma de gráfico tridimensional donde se observará la ubicación del pico de correlación así como el valor del pico máximo de la correlación de cada imagen con la opción de importar para cálculos estadísticos.

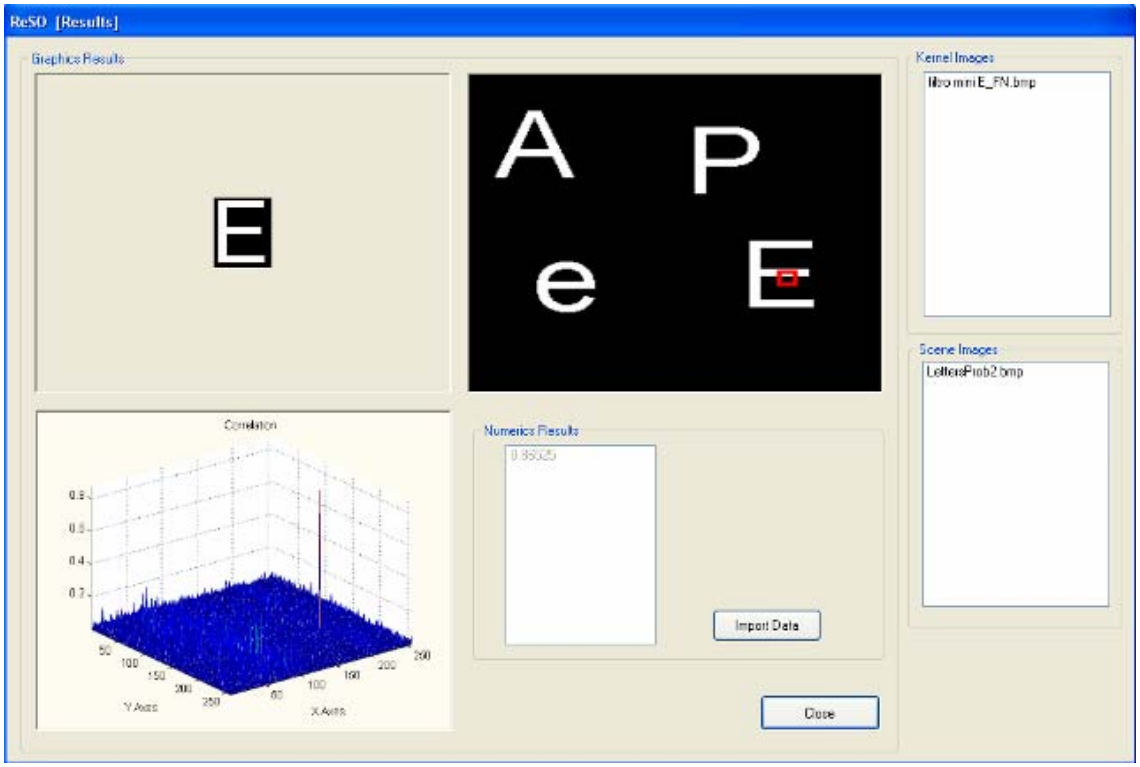


Figura 25.- Pantalla de resultados.

Es posible ver cada imagen resultante por separado en una pantalla individual mostrada en la figura 26. En ella podremos respaldar la imagen resultado en formato bmp o jpg, así como enviarla directo a una impresora.

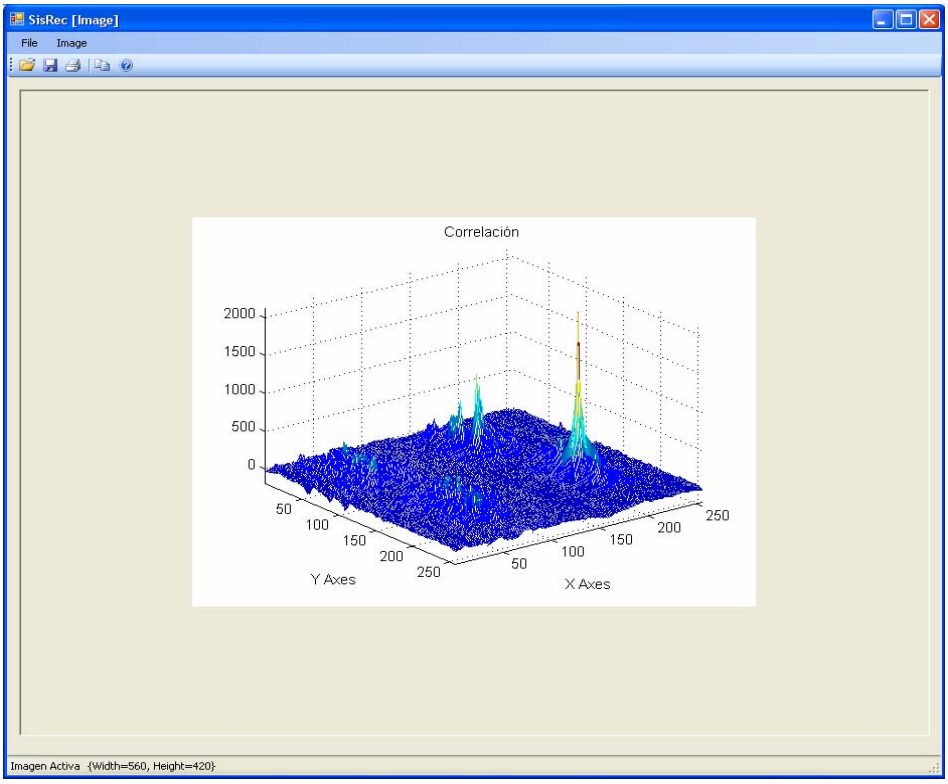


Figura 26.- Editor de imagen de SISREC.

Esta pantalla editor contiene los siguientes menús y submenús:

- File
  - Close Image: cierra la actual imagen mostrada en el editor, CTL+I.
  - Save as: guardar la imagen actual en disco, CTL+S.
  - Print: imprimir la imagen, CTL+P.
  - Exit: cerrar la pantalla del editor, CTL+X.
- Image
  - Copy to: copia a portapapeles la imagen actual, CTL+C.
  - Original: muestra la imagen en tamaño original, F2.
  - Stretch: expande la imagen al tamaño del editor, F3.
  - Center: centra la imagen en el editor, F4.
  - Turn Horizontal: hace un giro a la imagen en forma horizontal, CTL+H.
  - Turn Vertical: gira la imagen verticalmente, CTL+V.
  - Turn 90°: gira la imagen activa 90 grados de su posición actual.
  - Turn 180: gira la imagen 180 grados.

CAPÍTULO 4

RESULTADOS Y DISCUSIONES

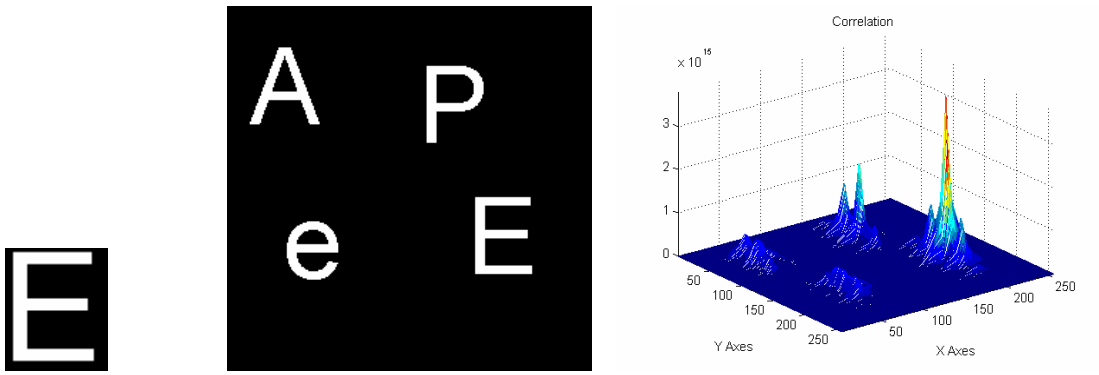
Debido a la naturaleza y eficacia de los filtros solo se realizaron pruebas básicas con letras en el filtro CMF, algunas más exhaustivas en el POF e IF y finalmente la gran mayoría en el filtro compuesto. Se añadieron ciertas restricciones dependiendo del filtro a usar con la finalidad de tener un mejor rendimiento computacional.

4.1 Filtros Lineales

Las restricciones con filtros lineales son: una imagen kernel contra *N* número de imágenes problema y dimensiones de estas ultimas indiferente entre si. La antigüedad de estos filtros es visible en sus resultados al no ser variantes a rotación ni escala, fue por ello que se determinaron estas restricciones, ya que era generar costo computacional extra a sabiendas de que los resultados son de menor calidad comparados con los del filtro no lineal.

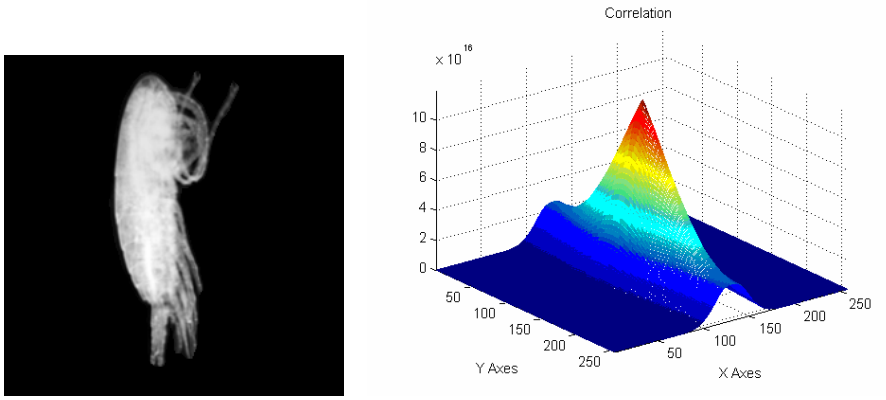
El usar únicamente una imagen kernel permite que solo se haga una operación de MatrizFusion (3.3.1) para cada imagen problema y si fueran mas de una esta operación se repetiría N veces dependiendo de la cantidad de imágenes kernel.

La figura 27 muestra los resultados con el filtro CMF utilizando la letra E como filtro y en la figura 28 usando un copépodo.



(a) (b) (c) (

Figura 27.- Resultados de correlación utilizando el filtro CMF con letra E como kernel (a) y un grupo de letras problema (b), plano de salida de la correlación (c).

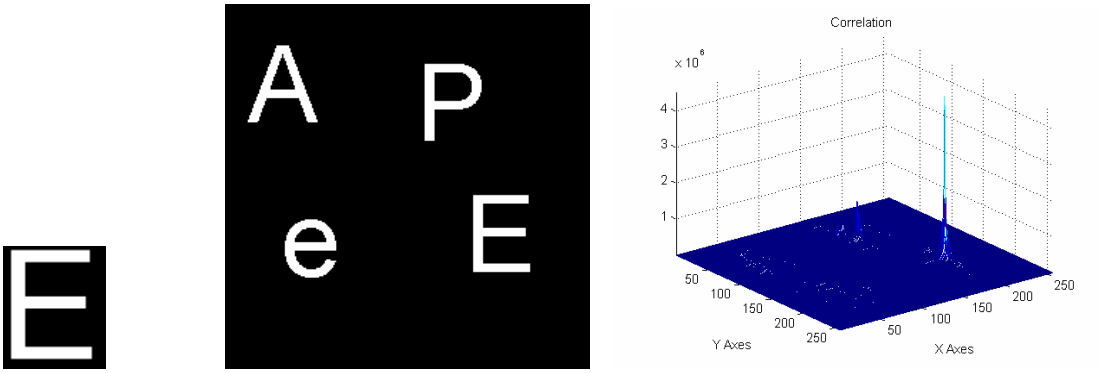


(a) (b)

Figura 28.- Resultados de correlación utilizando el filtro CMF con un copépodo correlacionado con el mismo (a) y (b) plano de salida de la correlación.

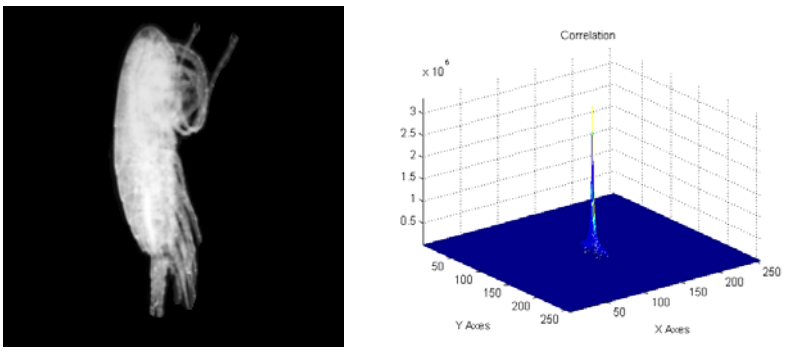
Es notable la falta de discriminación en imágenes con más tonalidades de blanco/negro como lo fue el copépodo a comparación de las letras, es por esto que únicamente se utilizó el filtro CMF para pruebas con letras.

Por su parte, el filtro POF al tener más poder de discriminación mostró el siguiente resultado con las mismas letras (figura 29) y el mismo copépodo (figura 30).



(a) (b) (c)

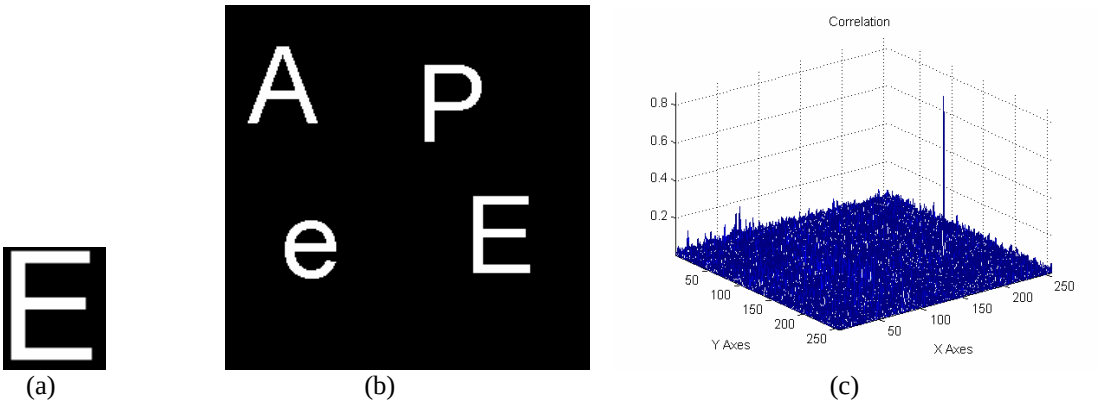
Figura 29.- Resultados de correlación utilizando el filtro POF con letra E como kernel (a) y de letras problema (b), plano de salida de correlación (c).



(a) (b)

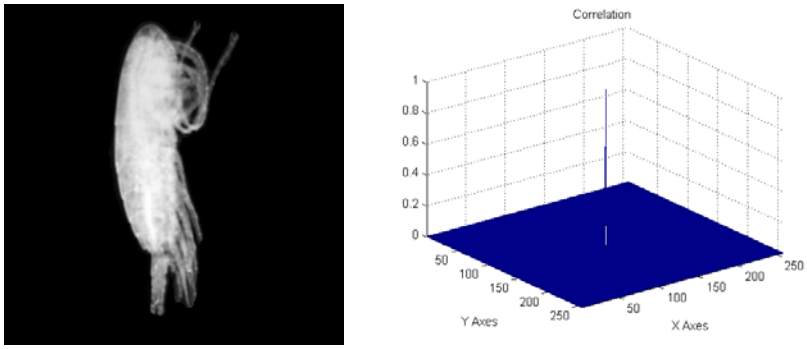
Figura 30.- Resultados de correlación utilizando el filtro POF con un copépodo correlacionado con el mismo (a) y (b) plano de salida de correlación.

Finalmente el filtro IF, correlacionado con las mismas imágenes dió el resultado mostrado en la figura 31 y 32.



(a) (b) (c)

Figura 31.- Resultados de correlación utilizando el filtro IF con letra E como kernel (a) un conjunto de letras problema (b), plano de salida de correlación (c).



(a) (b)

Figura 32.- Resultados de correlación utilizando el filtro IF con un copépodo correlacionado con el mismo (a) y (b) plano de salida de correlación.

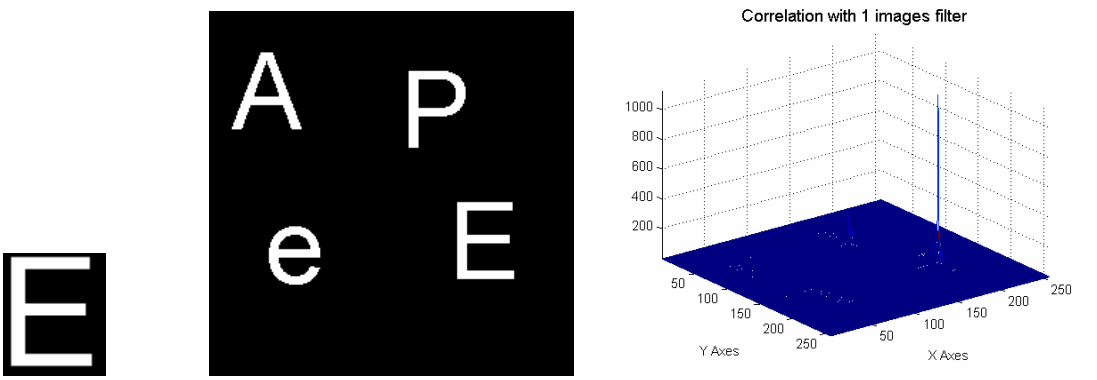
Los resultados tanto en el IF como en el POF son de mucha mayor calidad comparados con el CMF, es por eso que se realizaron mas pruebas con estos últimos dos para ser comparadas con el filtro compuesto.

4.2 Filtro No Lineal

4.2.1 Filtro No Lineal Compuesto

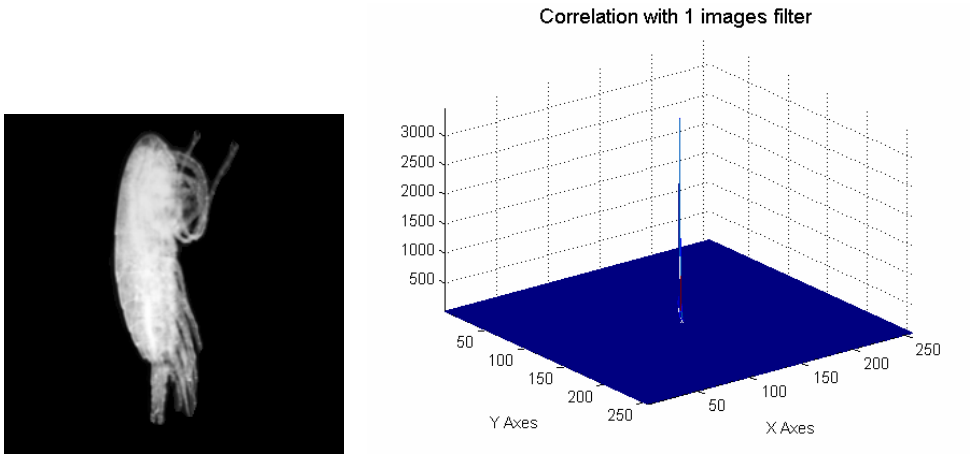
La eficacia de un filtro no lineal compuesto frente a un lineal es muy clara principalmente por las variantes que nos da el número de imágenes que contendrá el filtro como de la variable *k*, factor de no linealidad. Las restricciones dentro del filtro no lineal compuesto fueron N numero de imágenes para entrenar al filtro con un máximo de 216 (Guerrero Moreno, 2008) para garantizar una confiabilidad superior al 80% y N numero de imágenes problema pero de las mismas dimensiones entre si.

En la figura 33 y 34 se muestra un ejemplo de una operación aplicada a la letra E y al copépodo ya conocido, pero con un nivel de k=0.1 y únicamente una imagen filtro.



(a) (b) (c)

Figura 33.- Resultados con un filtro No Lineal con letra E como kernel (a) un conjunto de letras problema (b), plano de salida de correlación (c) con k=0.1.



(a) (b)

Figura 34.- Resultado con un filtro No lineal con un copépodo correlacionado con el mismo (a) y (b) plano de salida de correlación con k=0.1.

4.2.2 Filtro No Lineal Invariante

Con el fin de profundizar mas la calidad en resultados de un filtro no lineal se añadió un filtro invariante a rotación y escala (Coronel-Beltrán y Álvarez-Borrego, 2008). En este únicamente son mostrados los picos de acuerdo a la escala y rotación de la imagen con respecto del centro, es decir, el plano de salida estará dando en razón Y-escala y X-rotación. Es importante mencionar que no se realizaron pruebas exhaustivas con este filtro al no ser nuestro centro de estudio tal cual lo es el compuesto.

En la figura 35 podemos ver la in variancia a escala en el resultado de correlacionar una imagen filtro de la letra E con una imagen problema de varias de ellas.

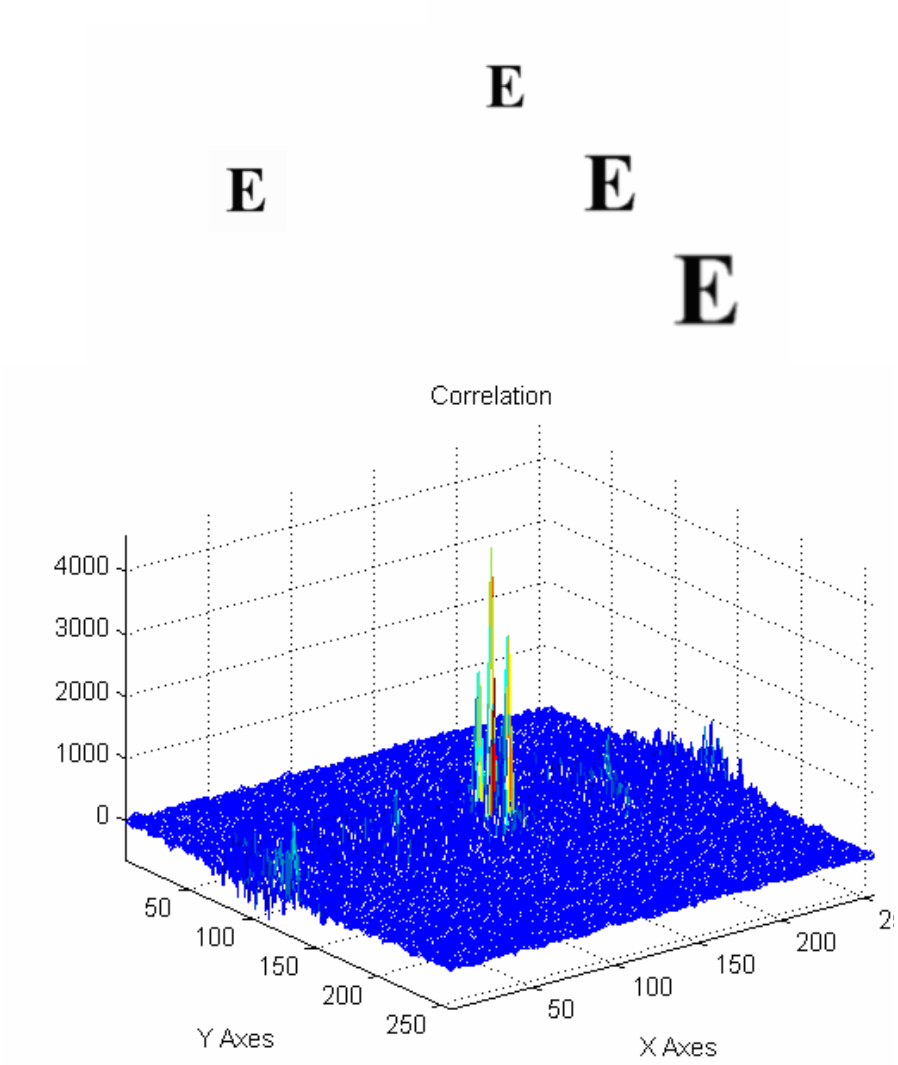


Figura 35.- Resultado de aplicar un filtro invariante a posición, rotación y escala.

4.3 Resultado Visual

El mostrar la ubicación de un pico de correlación no es suficiente para demostrar una correcta identificación, principalmente para el usuario ajeno a estas técnicas, fue por ello necesario dibujar una marca de referencia dentro de la imagen problema.

Debido a la variabilidad en calidad, tamaño e información de las imágenes que se puedan utilizar se optó por no normalizar los resultados y trabajar con valores naturales, es decir, la gráfica de los picos no estará en escala de 0-1 sino en valores arbitrarios, dependiendo de la imagen. Esto permitió delimitar un umbral de confianza superior al 90% independientemente de la calidad y tamaño de la imagen.

Considerando primero la correlación del filtro con él mismo, se obtuvo un valor numérico máximo y uno mínimo que determinarían el porcentaje, como se muestra en la figura 36, donde el valor máximo del pico es 5305873.9, por lo tanto el valor mínimo a considerar como aceptable será hasta 530587.3 para imágenes donde se requiera buscar esta figura, que corresponde hasta el 90% en el POF.

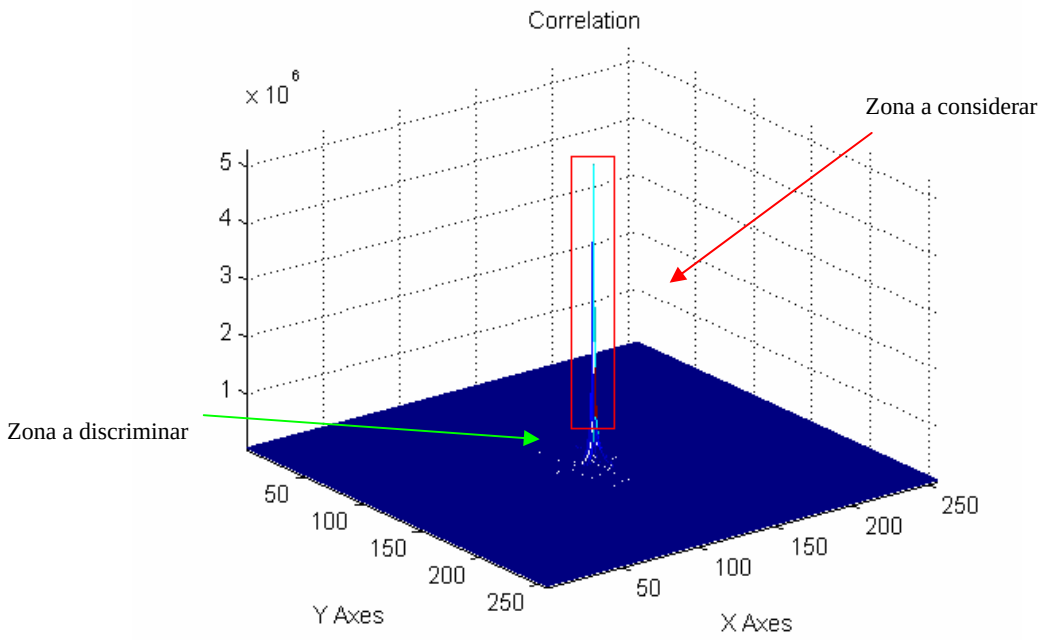


Figura 36.- Correlación de una imagen consigo misma utilizando el filtro POF para determinar un umbral de confianza.

Para el caso del CMF y el IF con la misma imagen los resultados se ven en la figura 37, con valores de $1.79x10^{19}$ y 1 como máximos, respectivamente para cada uno.

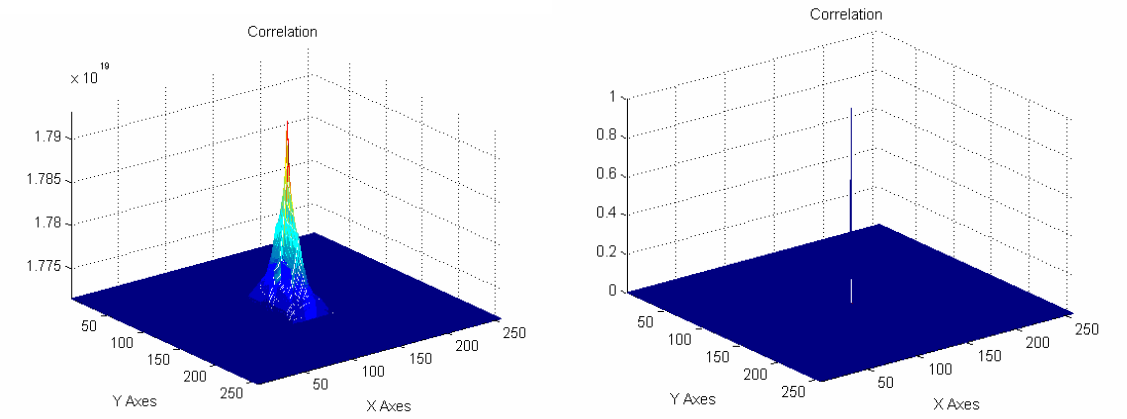


Figura 37.- Correlación de una imagen consigo misma con los filtros CMF e IF respectivamente, para determinar el umbral de confianza.

Es notable la diferencia de resultados de un filtro con respecto de otro pero con este método fue posible delimitar datos viables y no viables independientemente de la imagen y el tipo de filtro lineal a ocupar, en la figura 38 se observa un resultado ya aplicado a una imagen problema, donde el valor del pico máximo de la correlación con el filtro de fase es 5190851.5, que pertenece al rango de confiabilidad y por consiguiente si es pintado dentro de los resultados, por el contrario las pequeños protuberancias que se ven en la parte baja de la imagen, no son mostradas al estar por debajo del nivel de confiabilidad.

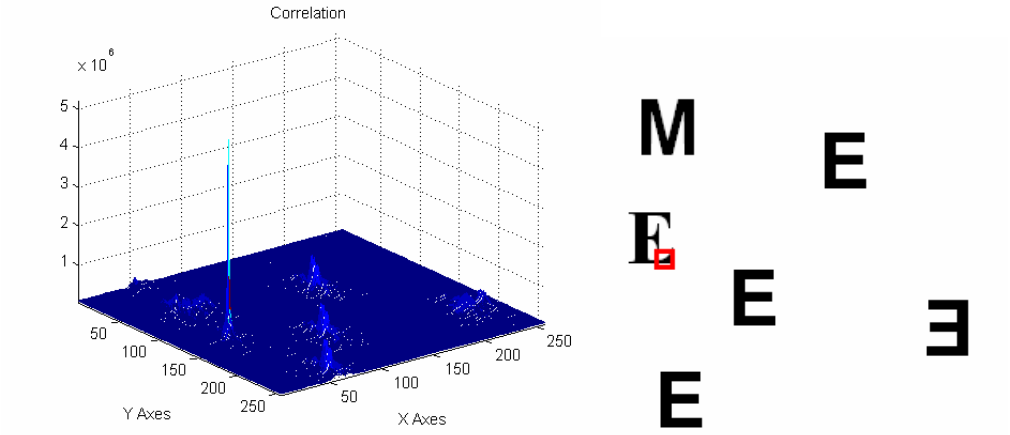


Figura 38.- Resultado de aplicar el nivel de confiabilidad a los resultados cuando se utiliza el filtro POF.

En el caso de IF los resultados son muy similares al POF, caso contrario del CMF que genera demasiado ruido y por consiguiente no es muy común su uso dentro del reconocimiento de imágenes, estos resultados los podemos ver en la figura 39, con un valor de .96, siendo del rango [0.1-1] para filtro POF y en la figura 40 para filtro CMF donde, al ser notable el ruido y la poca discriminación nos muestra resultados erróneos aun dentro del nivel de confiabilidad.

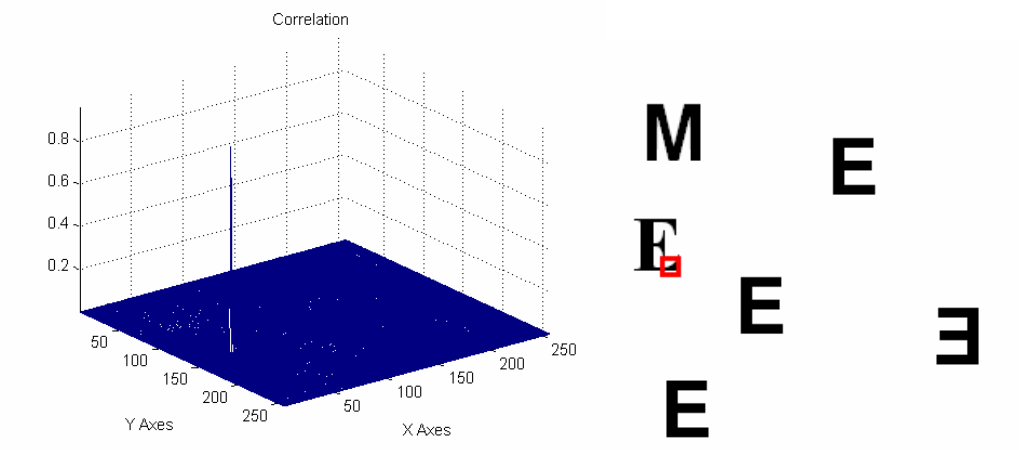


Figura 39.- Resultado de aplicar el nivel de confiabilidad a los resultados cuando se utiliza el filtro IF.

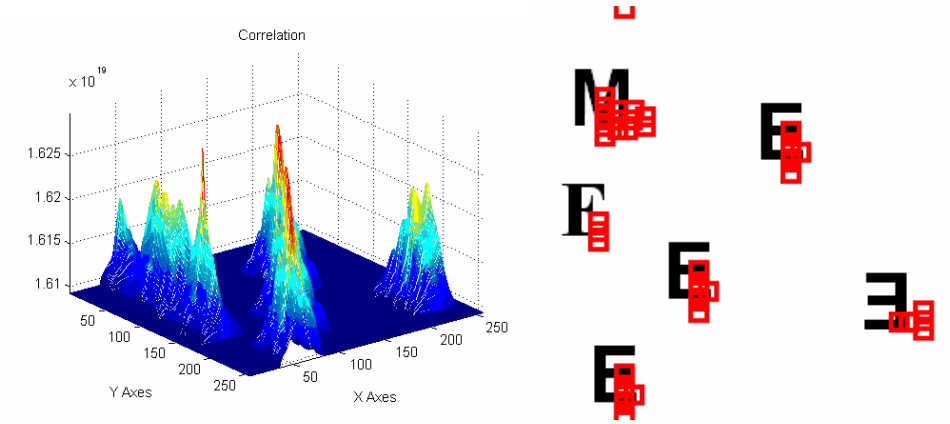


Figura 40.- Resultado de aplicar el nivel de confiabilidad a los resultados cuando se utiliza el filtro CMF.

En el caso del filtro compuesto la situación fue distinta, ya que los resultados son afectados por algunas variables más: número de imágenes del filtro, valor de *k* y por supuesto calidad y tamaño de las imágenes. En este caso se buscó una continuidad que no afectara menos del

90% de confiabilidad en los resultados, llegando a una posible similitud hasta el uso de k=0.3, después de ahí los valores discrepaban enormemente, la continuidad hallada fue:

Para una imagen filtro con valores de *k*<=0.3, el umbral está dado por la ecuación

$$U = max*k, \tag{36}$$

para N imágenes y valores de *k*<=0.3, el umbral está dado por el valor máximo de la auto correlación multiplicada por 0.1 que nos garantiza no menos del 90% y dividido por el número de imágenes entre *k*,

$$U = (max*0.1) / (numImg/k), \tag{37}$$

donde *U* representa el valor mínimo del umbral, *max* el máximo de la auto correlación, *k* un valor entre [0- 0.3] y *numImg* el número total de imágenes del filtro.

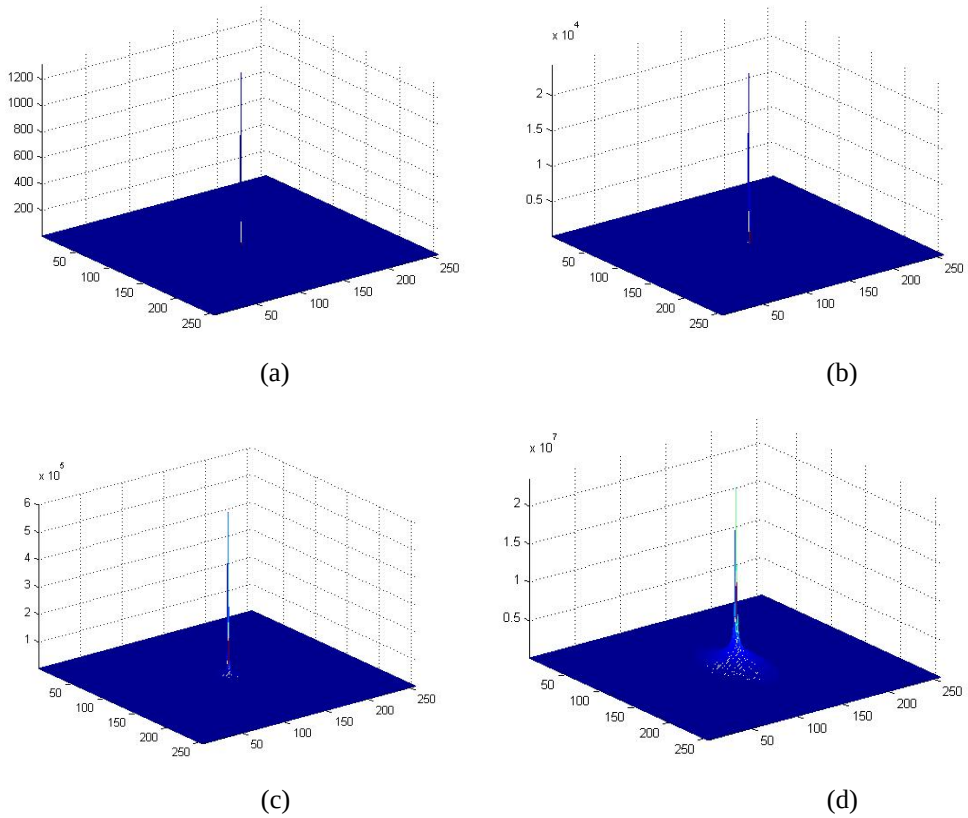


Figura 41.- Resultado de la auto correlación de un filtro compuesto con 9 imágenes de copépodos y con valores de *k*=0.1 (a), *k*=0.2 (b), *k*=0.3 (c) y *k*=0.4 (d).

En la figura anterior (41) se muestra la auto correlación de un filtro con 9 imágenes de copépodos, donde el valor máximo es 601770 y el limite a considerar dentro del umbral 2005.9 con *k*=0.3, 24258 a 53.9 para *k*=0.2, 1310.3 a 1.45 con *k*=0.1 y 23416000 a 104071.1 con *k*=0.4. Es notable la diferencia al subir el valor de *k* a 0.4, los rangos del umbral son demasiado grandes y como se aprecia en la figura anterior se comienza a perder la línea fina generando un poco de ruido.

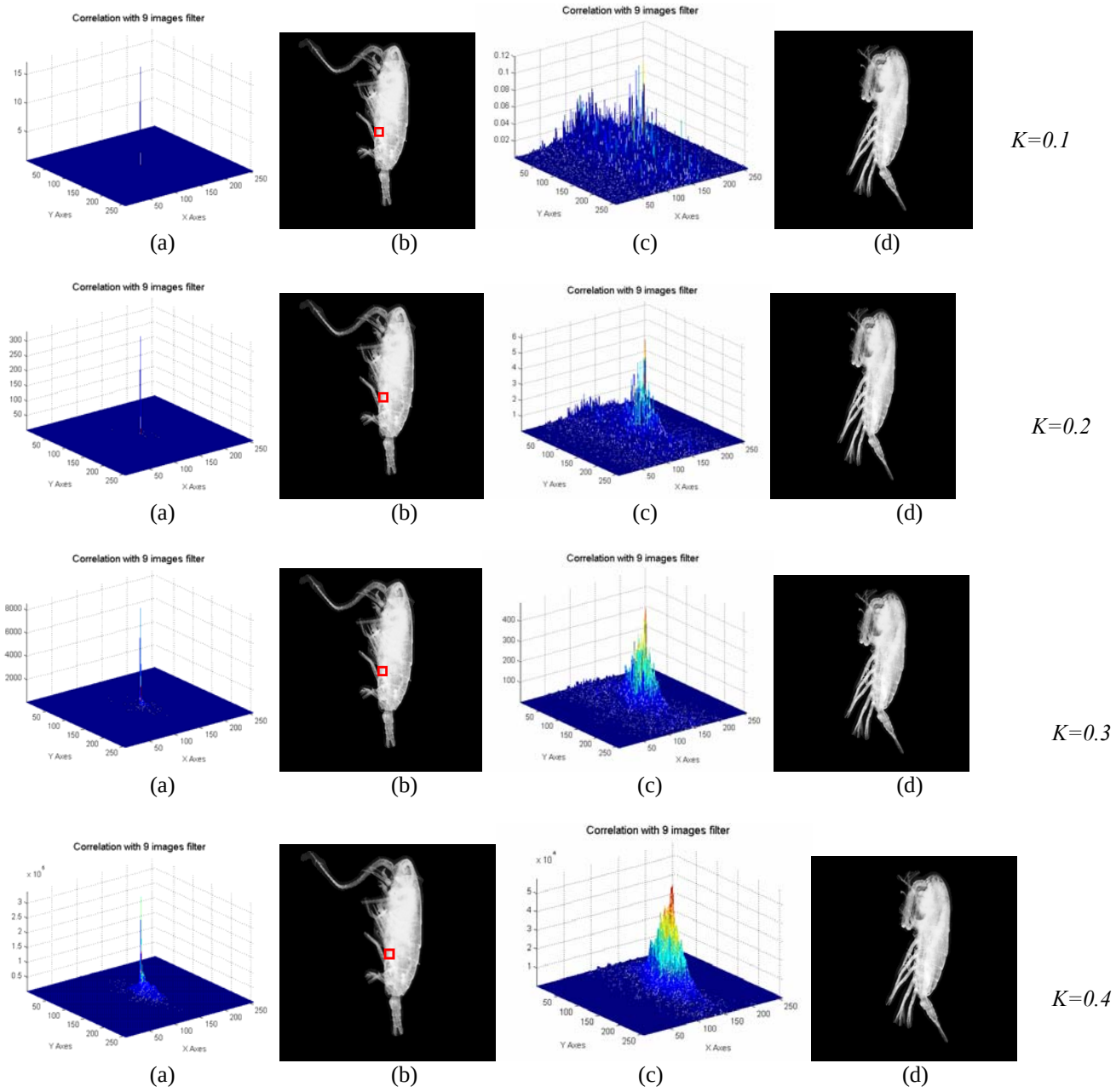


Figura 42.- Resultado de aplicar el nivel de confiabilidad para pintar en la imagen resultado utilizando 9 imágenes para el filtro y distintos valores para *k*, (a) pico de identificación correcto, (b) figura resultado con marca, (c) pico de correlación de objeto no identificado y (d) objeto no identificado y sin marca.

En la figura 42 se puede apreciar mejor el resultado de aplicar el umbral con la correlación de un filtro de 9 imágenes contra una perteneciente a él y otra ajena, considerando los cuatro valores de *k* anteriores.

4.4 Resultados Numéricos

Siempre que se realizan identificaciones taxonómicas es necesario validar los resultados visuales con resultados numéricos con el fin de darles un mayor respaldo a ellos, por consiguiente se anexó a la aplicación una opción de importar datos para poder ser procesados en algún programa estadístico. En las figuras siguientes se observan los resultados de las imágenes trabajadas con el filtro lineal compuesto.

Se utilizaron 420 imágenes de copépodos en 2 modalidades distintas (con fondo blanco y con fondo negro) dando un total de 840; cada grupo de 420 está conformado por 7 especies distintas de copépodos divididos en hembras y machos, segmentados de la siguiente manera 1-60 especie A (*Calanus pacificus*), 61-120 especie B (*Rhincalanus nasutus*), 121-180 especie C (*Centropages furcatus*), 181-240 especie D (*Pleuromamma gracilis*), 241-300 especie E (*Temora discaudata*), 301-360 especie F (*Acartia tonsa*), 361-420 especie G (*Centropages hamatus*), donde los primeros 30 siempre son hembras y los segundos machos.

Se realizó un filtro compuesto de 15 imágenes de cada grupo (hembra/macho/especie) y se comparó frente a las 420, primero con las imágenes de fondo blanco y posteriormente con las de fondo negro, todas con un valor de k=0.1 (Guerrero Moreno, 2008). Los resultados estadísticos de las imágenes con fondo blanco aparecen en las figuras siguientes donde se puede apreciar que el filtro discrimina no sólo entre especie sino también en sexo y logrando una confiabilidad del 100%.

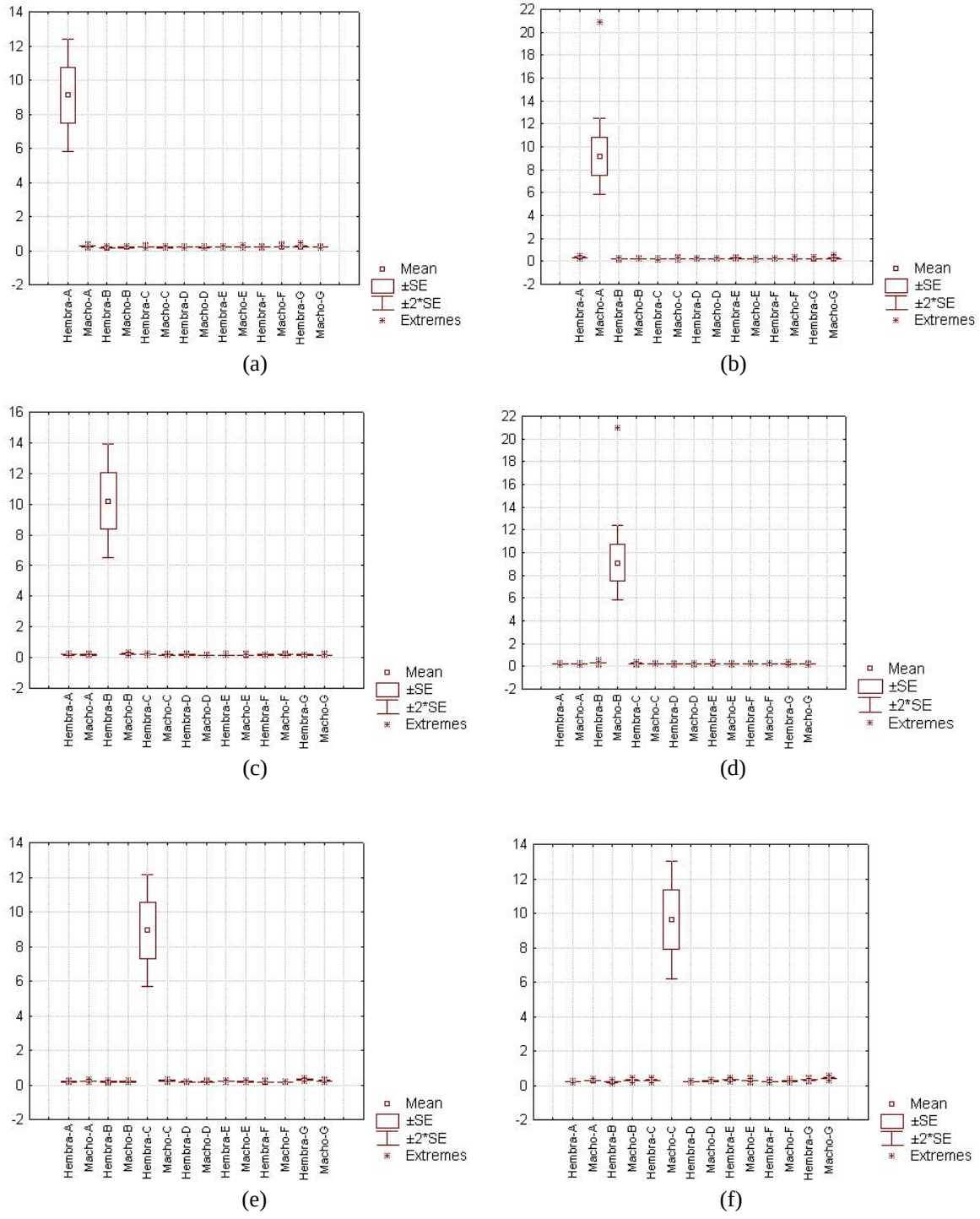


Figura 43.- Resultados obtenidos de aplicar el filtro no lineal compuesto con *k*=0.1 a 7 diferentes especies de copépodos.

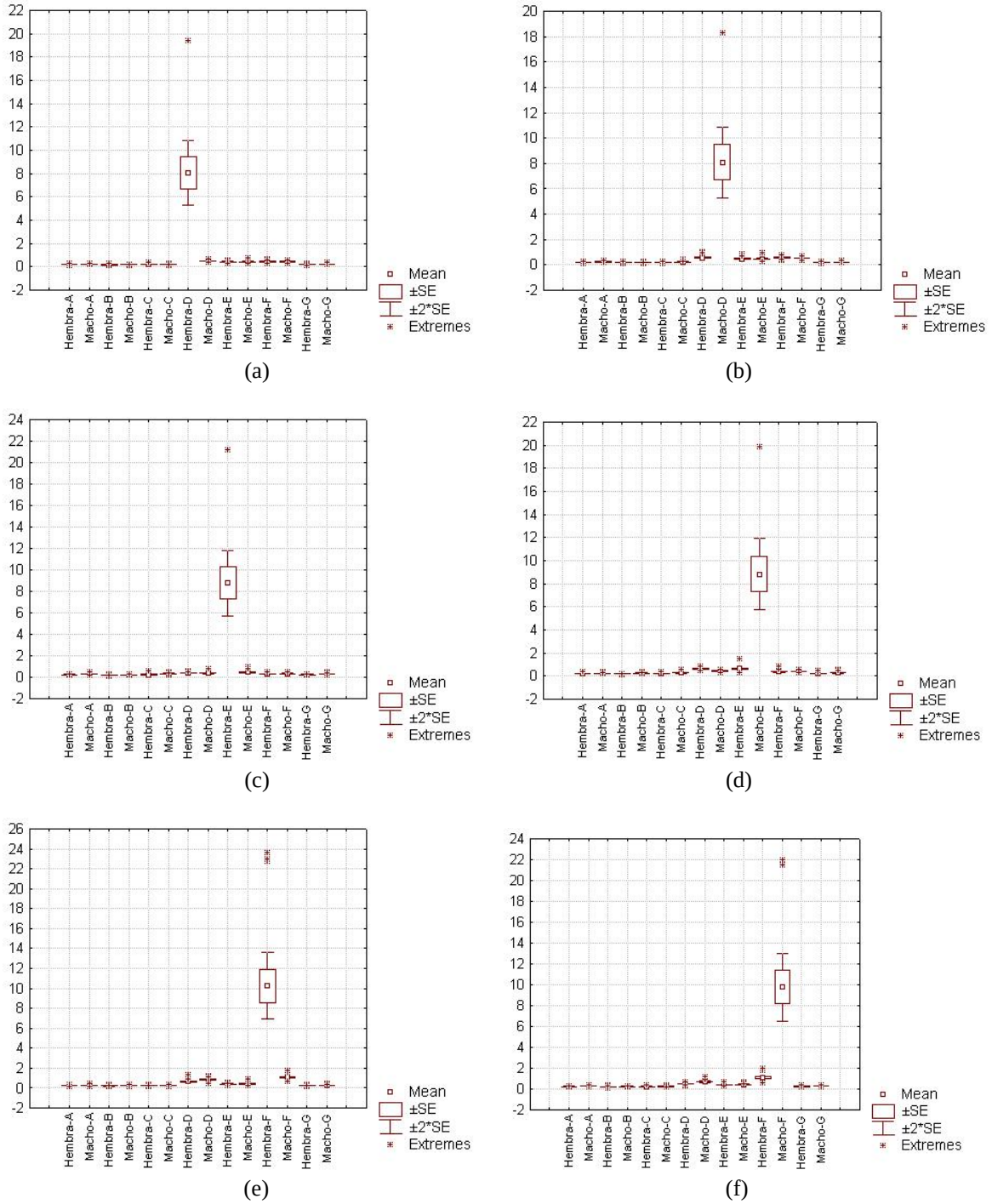


Figura 44.- Continuación de los resultados obtenidos de aplicar el filtro no lineal compuesto con *k*=0.1 a 7 diferentes especies de copépodos.

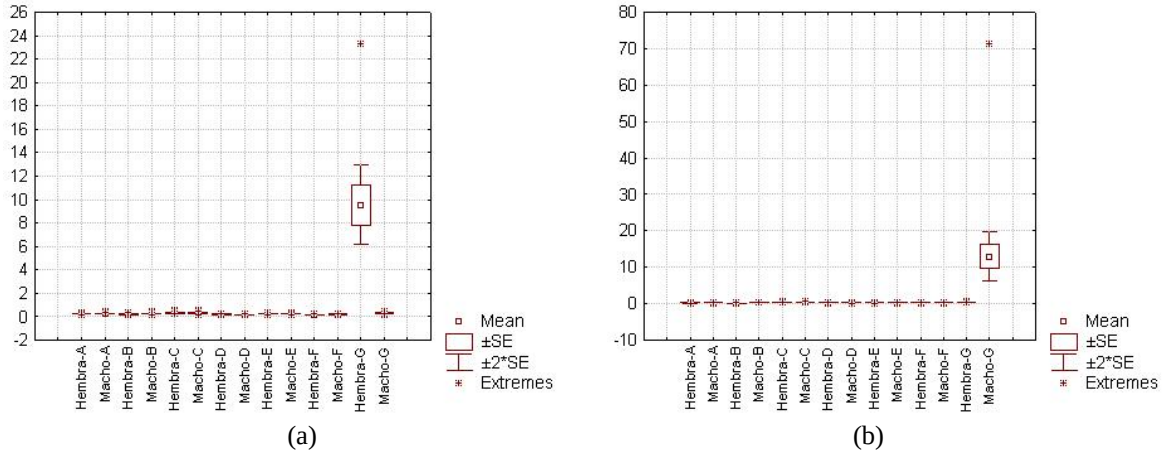


Figura 45.- Continuación de los resultados obtenidos de aplicar el filtro no lineal compuesto con *k*=0.1 a 7 diferentes especies de copépodos.

En la figura 46 se muestran algunas de las imágenes utilizadas en el proceso.

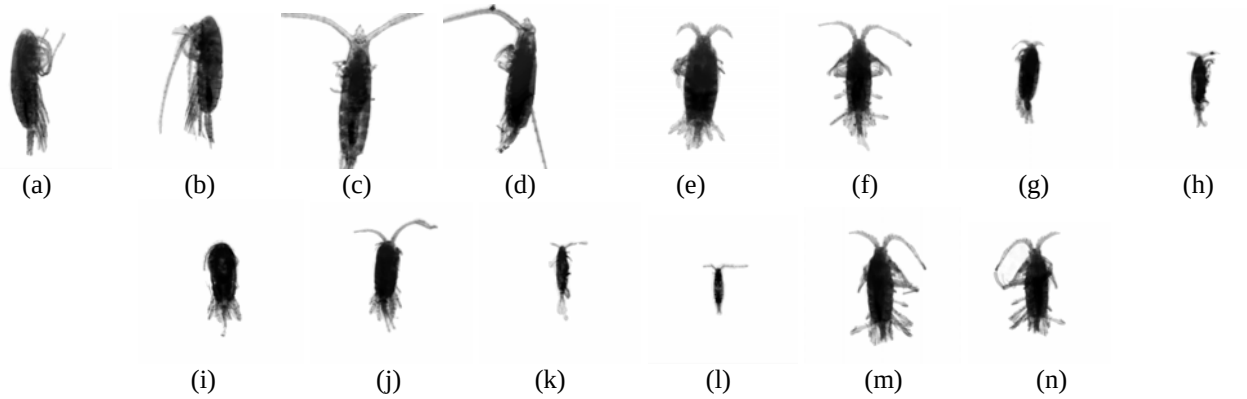


Figura 46.- Imágenes de copépodos con fondo blanco utilizadas, 7 especies distintas separadas en hembras y machos respectivamente, *Calanus pacificus* (a, b), *Rhincalanus nasutus* (c, d), *Centropages furcatus* (e, f), *Pleuromamma gracilis* (g, h), *Temora discaudata* (i, j), *Acartia tonsa* (k, l), *Centropages hamatus* (m, n).

En las figura 47 aparecen algunos resultados estadísticos de las imágenes con fondo negro, observándose una discriminación del 100% entre especies y género, permitiendo también demostrar que no importa el fondo de la imagen (ya sea blanco o negro) los resultados son igualmente confiables.

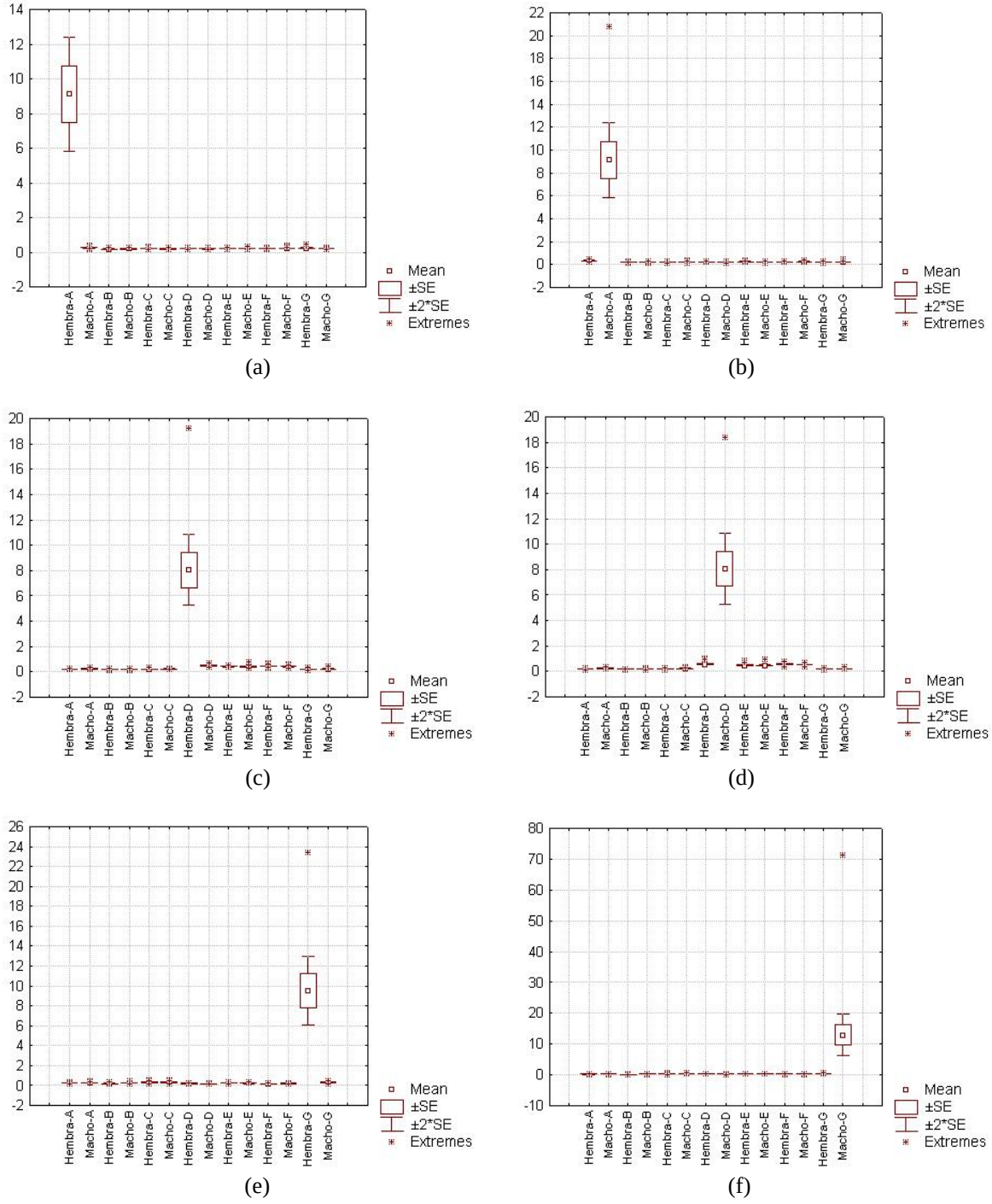


Figura 47.- Algunos de los resultados obtenidos de aplicar el filtro no lineal compuesto con *k*=0.1 a 7 diferentes especies de copépodos con fondo negreo.

La figura 48 muestra las imágenes de los copépodos con fondo negro.

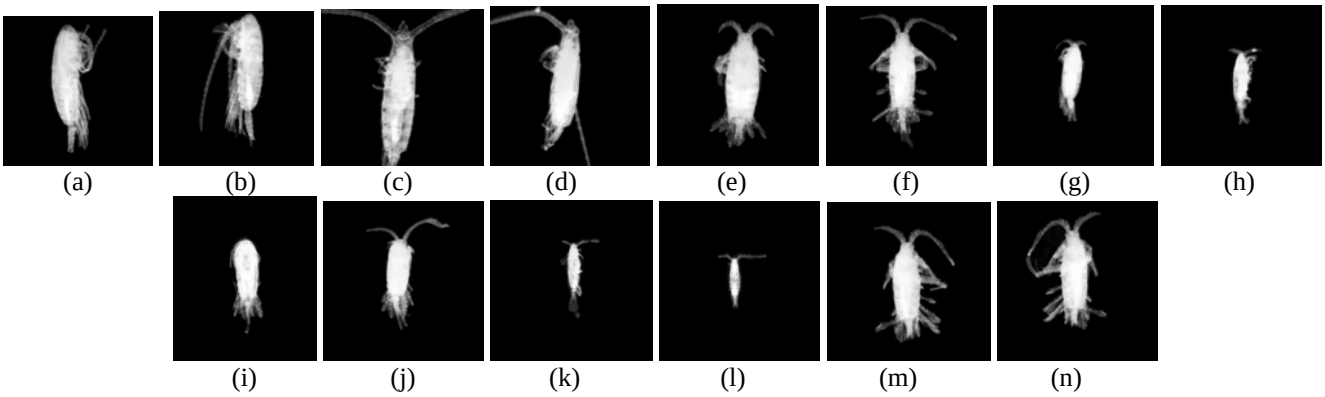


Figura 48.- Imágenes de copépodos utilizadas, 7 especies distintas separadas en hembras y machos respectivamente, *Calanus pacificus* (a, b), *Rhincalanus nasutus* (c, d), *Centropages furcatus* (e, f), *Pleuromamma gracilis* (g, h), *Temora discaudata* (i, j), *Acartia tonsa* (k, l), *Centropages hamatus* (m, n).

4.5 Muestras y pruebas adicionales

Gracias a la colaboración de la FES Iztacala se obtuvieron muestras de algunos rotíferos y poder probar el software al tratar de discriminar entre morfologías distintas del microorganismo, algunas muestras fueron tomadas en el departamento de óptica del CICESE y algunas más en la misma FES. Se obtuvieron resultados satisfactorios y superiores al 90% de confiabilidad, en las figuras 49 y 50 se pueden apreciar algunos de estos resultados y las imágenes empleadas para ello.

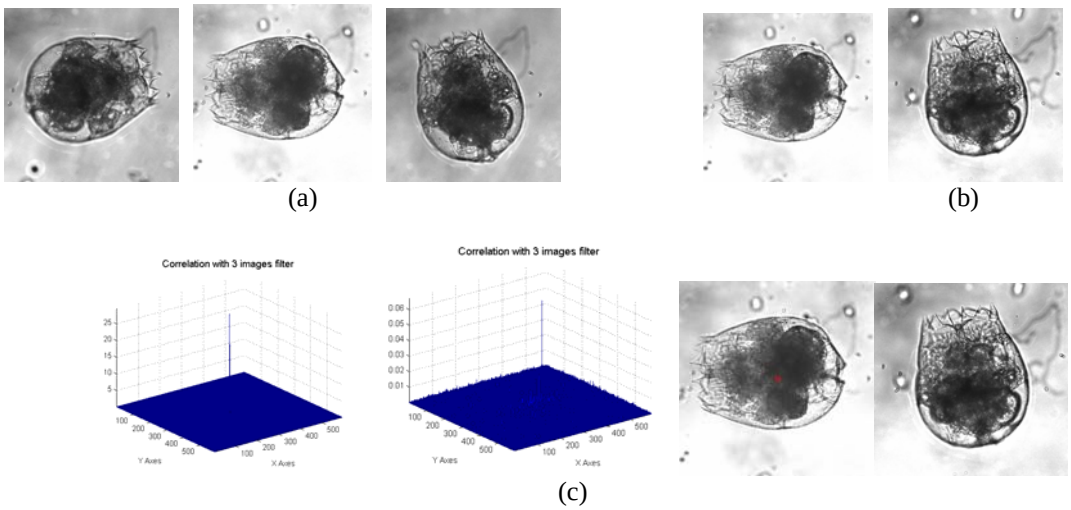


Figura 49.- Imágenes de Brachionus utilizadas, imágenes del filtro (a), imágenes problema (b) y resultados gráficos de la correlación (c).

En la siguiente figura se muestran algunas imágenes de rotíferos, debido a la falta de variedad de muestras se realizó artificialmente una imagen problema con un conjunto de ellas, se eligieron algunas para pertenecer al filtro y así buscarla dentro de la fabricada, los resultados fueron los mostrados en la figura 50.

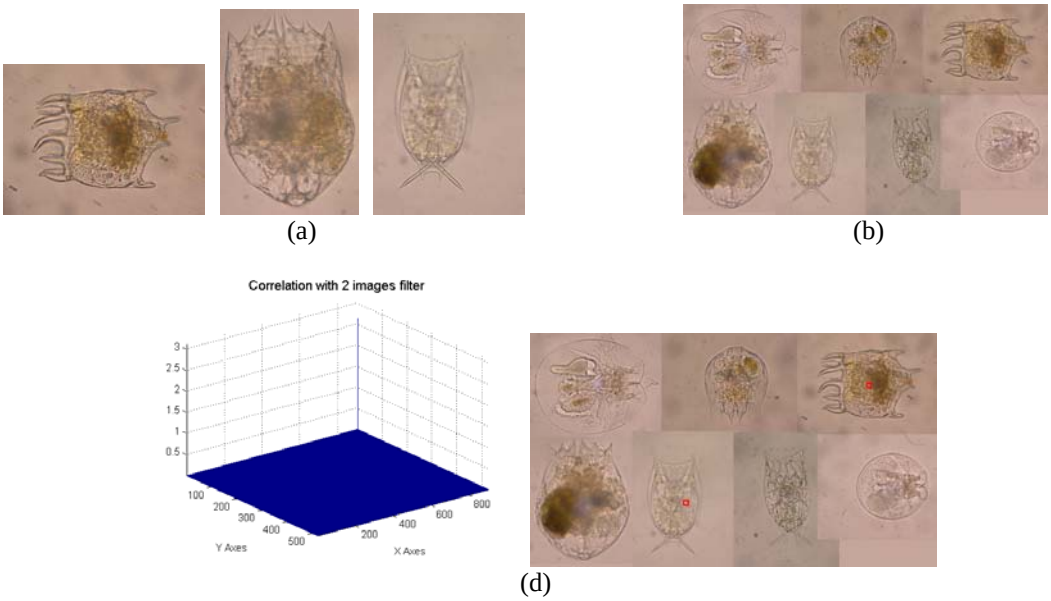


Figura 50.- Correlación de 3 imágenes filtro de rotíferos (a) y una imagen problema fabricada manualmente (b) y sus resultados gráficos (c).

En las figuras anteriores es notable la capacidad de discriminación del filtro, se encontraron algunos problemas debido a la iluminación variante en cada muestra ya que este filtro no es invariante a esta, pero no impidió permanecer dentro del 90% de confiabilidad necesitado para declarar una correcta identificación.

CAPÍTULO 5

CONCLUSIONES

En este trabajo se presenta una nueva aplicación capaz de trabajar con diferentes tipos de filtros para reconocimiento de patrones, distribuible e instalable en cualquier equipo de escritorio de tipo PC. Desarrollada bajo la robustez que permite .Net y el poder matemático de un software científico como lo es MatLab, trabaja de trasfondo con 3 filtros lineales básicos, el CMF originario en 1964, el POF introducido 20 años después y finalmente el IF que es capaz de darnos picos de correlación mas delgados pero poca discriminación en algunos casos; 2 filtros no lineales, uno compuesto y otro invariante a rotación y escala.

Por lo tanto la hipótesis de este estudio se valida al demostrar que el uso de nuevos filtros no lineales es posible al haber generado una aplicación que permita el trabajo con ellos de una manera sencilla y directa, sin la necesidad de conocer las bases matemáticas que ellos implican.

Se trabajó con series de letras y algunas imágenes de microorganismos demostrando que es posible generar un filtro compuesto en tiempo de ejecución, realizando identificaciones de objetos en tiempo real independientemente del número de imágenes que contenga el filtro, permitiendo tener mas del 90% de confiabilidad en las identificaciones.

Es posible trabajar con N número de muestras o imágenes a localizar independientemente de las utilizadas en el filtro. El rendimiento de la aplicación va de la mano con las características del equipo, trabajando desde un Pentium II hasta un equipo con procesador a 64 bits, variando por supuesto la velocidad a la hora de mostrar los resultados más no la calidad de estos.

Se encontró cierta estabilidad en equipos con sistema operativo Windows XP, Vista requiere más atención debido a que fue necesario cambiar los atributos del ejecutable y

correrlo como administrador. De igual forma el desempeño es mucho mayor en equipos con Vista a equipos con Xp y por consiguiente a equipos de 32 y 64 bits.

Se encontró cierta baja de desempeño al encontrar imágenes con variantes de iluminación debido a que nuestros filtros no son variantes a ella, aun así los resultados se encontraron superiores a los 90% de confiabilidad esperados.

La oportunidad de trabajar con muestras ajenas al laboratorio permitió comprobar aun mas la forma de crear nuestro umbral para pintar nuestros resultados.

Es importante el tener una base de datos muy extensa de imágenes para poder generar filtros compuestos de calidad y por consiguiente garantizar una identificación mas optima, claramente es necesario un trabajo futuro que permita recaudar y comenzar esta base de datos, de igual forma una automatización tal vez mayor, no solo en la creación del filtro y la correlación con algunas escenas, sino desde el punto de obtener la imagen en tiempo real del microscopio y a su vez comenzar la identificación.

Un punto importante dentro de la obtención automática radica también que muchas ocasiones no se requiere tomar toda la muestra, solo pequeños sectores, ese tipo de automatización ahorraría mucho tiempo computacional.

Es importante también generar nuevos filtros compuestos que sean invariantes a iluminación, situación muy presentada en algunas muestras debido a la falta de un estándar a la hora de tomarlas.

Por el lado del software seria importante un énfasis a operabilidad vía Web, tal vez no con toda la funcionalidad del trabajo en la misma estación pero si con la suficiente robustez para garantizar un trabajo a distancia a través de la red. Las restricciones y limitaciones del software propietario siempre son limitantes, por ello seria recomendable la búsqueda de alternativas libres de licenciamientos y distribuciones.

El uso de más formatos gráficos como JPG, TIF, entre otros, dentro de la aplicación también debe ser considerado como una opción más dentro de ella.

Finalmente y de una manera general se generó una aplicación capaz de hacer identificaciones automáticas de microorganismos de 5 formas distintas (filtros), ahorrando el tiempo de hacerlo manualmente en mas de un 200%, con una interfaz sencilla y la posibilidad de guardar e importar los resultados generados para cálculos estadísticos.

REFERENCIAS

- Álvarez-Borrego Josué, Rosa Reyna Mouriño-Pérez, J. L. Pech-Pacheco y G. Cristóbal, 2000. *"Identificación de Vibrio cholerae 01 mediante correlación invariante".* XIII Reunión Anual de Óptica, Puebla, Puebla, 70-72 p.

- Álvarez-Borrego Josué y María Cristína Chávez Sánchez, 2001. *"Detection of IHHN virus in shrimp tissue by digital color correlation"*. Aquaculture, 194(1-2):1-9.

- Álvarez-Borrego, J., Mouriño Pérez, R., G. Cristóbal Pérez y J. Luís Pech Pacheco. 2002. *"Invariant recognition of polychromatic images of Vibrio cholerae 01".* Optical Engineering, 41(4): 827-833 p.

- Álvarez-Borrego Josué y Ernestina Castro Longoria, 2003. *"Discrimination between Acartia (Copepoda: Calanoida) species using their diffraction pattern in a position, rotation invariant digital correlation"*. Journal of Plankton Research, 25(2):229-233 p.

- Álvarez-Borrego, J. y Fájer-Ávila, E.J., 2006. *"Identification of Platyhelminth parasites of the Wild Bullseye Pufferfish (Sphoeroides annulatus) using an invariant digital color correlation"*. Revista de Biología Marina y Oceanografía, 41(1):129-139 p.

- Álvarez-Borrego J., Gallardo Escárate C., V. Kober, O. R. López Bonilla, 2007. "*Genome size estimation: a new methodology*". SPIE, Sixth Symposium Optics in Industry, 6422(04):1-11 p.

- Álvarez-Borrego, 2008. Comunicación personal.

- Bueno-Ibarra, M. A., 2005. *"Desarrollo de una tecnología sistematizada para la adquisición y análisis de partículas biogénicas"*. Tesis Doctoral, IPN. 228 pp.

- Bracewell, R.N., 2000. "*The Fourier transform and its applications"*. McGraw Hill. 3ra Ed., Boston. 616 pp.

- Casasent, D., 1984. *"Unified synthetic discriminant function computacional formulation"*. Applied Optics, 23(10):1620-1627 p.

- Castro-Longoria Ernestina, Josué Álvarez-Borrego y José Luis Pech-Pacheco, 2001. *"Identification of species of calanoid copepods using a new invariant correlation algorithm"*. Crustaceana, 74(10):1029-1039 p.

- Cho Z.-H., J. P. Jones, y M. Singh., 1993. *"Fundations of Medical Imaging"*. John Wiley & Sons, Inc., New York.

- Corbalán M, M. S. Millán y M.J. Yzuel, 2003. "*Reconocimiento de objetos en color procesando un sólo canal basándose en el histograma". P*ublicación de la 7ma reunión nacional de óptica, España 2003. Encontrado en enero 2008, http://www.optica.unican.es/RNO7/Contribuciones/articulospdf/corbalan2.pdf

- Coronel Beltrán, A. y Álvarez-Borrego, J., 2008. "*Nonlinear filter for pattern recognition using the scale transform". SPIE,* Applications of Digital Image Processing XXXI, 7073(2H):1-11.

- Fájer Ávila Emma Josefina y Álvarez-Borrego J., 2002. *"Invariant digital color correlation for the identification of worm parasites from bullseye pufferfish"*. SPIE-The International Society for Optical Engineering. Vol. 4790:511-517 p.

- Flores Núñez J. L., G. García Torales, J. Álvarez-Borrego, 2008. *"Inspección óptica automatizada de circuitos impresos"*. En: Josué Álvarez-Borrego y María Cristina Chávez Sánchez, Editores, Introducción a la Identificación Automática de Organismos y Estructuras Microscópicas y Macroscópicas. Ediciones de la Noche. ISBN: 978-97027-1396-8. 217-226 p.

- Forero Manuel G., Sierra-Ballén E. L., Álvarez-Borrego J., Cristóbal Gabriel, Pech-Pacheco J. L., Alcalá Luis, Desco Manuel, 2001. *"Automatic sputum color image segmentation for tuberculosis diagnosis",* SPIE, Algorithms and Systems for Optical Information Processing V., Baharam Javidi and Demetri Psaltis, Editors, Vol. 4471: 251-261 p.

- Forero Manuel G., Sroubek Filip, Álvarez-Borrego J., Malpica Norberto, Cristóbal Gabriel, Santos Andrés, Alcalá Luis, Desco Manuel, Cohen León, 2002. *"Segmentation autofocusing and signature extraction of tuberculosis sputum images".*SPIE-The International Society for Optical Engineering.

- Friedman Menahem y Kandel Abraham, 1999. *"Introduction to pattern recognition: statistical, structural, neural and fuzzy logia approaches*". Imperial College Press, London, 329 pp.

- Gallardo, C., Álvarez-Borrego Josué, M. A. Del Río-Portilla., V. Kober, 2004. "*Karyotype of the red abalone Haliotis rufescens (Archaeogastropoda: Haliotidae), using image analysis"*. Journal of Shellfish Research, 23(1):205-209 p.

- Gallardo Escárate, Cristian, 2005. *"Análisis citogenético en tres especies de abulones de Baja California mediante procesamiento digital de imágenes"*. Tesis Doctoral. Cicese, 139 pp.

- González-Fraga, J. A., V. Kober y Josué Álvarez-Borrego, 2005. *"Recognition of partially occluded objects using correlation filters with training"*. SPIE. Applications of Digital Image Processing XXVIII, 5909: 1-7 p.

- González-Fraga A. J., Vitaly Kober y Josué Álvarez Borrego., 2006. *"Adaptive SDF filters for recognition of partially occluded objects"*. SPIE, Applications of Digital Image Processing XXIX, 6312(1G):1-11 p.

- González Rafael C. y Woods Richard E., *"Digital Image Processing".* Prentice Hall 3ra Edición, USA 2008, 793 pp.

- Green W. B., 1983. *"Introduction to Digital Image Processing"*. Pages 1-23 in V. N. R. C. Inc., ed. Digital Image Processing - A System Approach. Van Nostrand Reinhold Co. Inc., New York.

- Guerrero Moreno, R., 2008. *Correlación invariante de objetos utilizando filtros compuestos no lineales".* Tesis Maestría. CICESE, 90 pp.

- Hall, Ernest L., 1979. *"Computer image processing and recognition*". New York, Academic, 584 pp.

- Hernández Constante Jorge, Josué Álvarez Borrego, Oscar Roberto López Bonilla, Rubén Ruelas Lepe, Jorge L. Flores, 2007. *"Discriminación entre calidad de paletas mediante un sistema digital que detecta posición y rotación de objetos*". CONCIBE Universidad de Guadalajara. 1030:1-7 p.

- Javidi, B., 1989. *"Nonlinear joint power spectrum based optical correlation".* Appl. Opt., 28 (12): 2358-2367 p.

- Javidi, B., Wang, W., Zhang, G., 1997. *"Composite Fourier-plane nonlinear filter for distortion-invariant pattern recognition"* Optical Engineering, 36 (10): 2690-2696 p.

- Kober, H. Díaz Ramírez, J. A. González Fraga y Josué Álvarez Borrego, 2007. *"Real time pattern recognition with adaptive correlation filters"*. Cap. 27, 515 pp., En: G. Obinata and A. Dutta, "Vision Systems, Segmentation and Pattern Recognition", Editorial Collegiums, I-Tech Education and Publishing. June 2007, Croatia.

- López Peinado D., 2003. "*Implementación multiplataforma de algoritmos de procesamiento de imágenes biomédicas 2D en MatLab y C++".* Tesis Maestría, España. Encontrado en mayo2008. http://www.elai.upm.es/spain/Investiga/GCII/personal/dlopez/dlopez.html#GUI%20MATLAB

- Mandell S. L., 1979. *"Computer and Data Processing concepts and applications"*. Evolution Of Computers. West Publishing Company, St. Paul, 22-36 p.

- Pech Pacheco J. L., 1998. *"Identificación de partículas biogénicas"*. Tesis Doctoral CICESE.

- Pech Pacheco J. L. y J. Álvarez-Borrego, 1998. *"Optical-digital system applied to the identification of five phytoplakton species"*. J. Marine Biology. 132(3): 357-366 p.

- Pratt W. K., 1991. "*Digital image processing"*. 2nd Edition, Wiley Interscience Press, USA, 62-72 p.

- Rodríguez-Santiago, M., 2003. *"Identificación de especies ectoparásitas del género Trichodina (Ciliophora: Peritrichida) en Tilapia nilotica mediante correlación invariante con filtros compuestos"*. Tesis de Maestría CIAD-Unidad Mazatlán.

- Rosa Reyna Mouriño-Pérez, Josué Álvarez-Borrego y Cristian Gallardo-Escárate, 2006. *"Digital Color Correlation for the Recognition of Vibrio cholerae 01 in Laboratory and Environmental Samples"*. Revista de Biología Marina y Oceanografía, 41(1):77-86 p.

- Russ J. C., 1995. "*The Image Processing Handbook*". 2nd. Edition, CRC Press, Inc., Boca Raton, Florida.

- Santos Carrera L., 2005. "*Análisis de imágenes biomédicas para la cuantificación del nivel de fragmentación de los espermatozoides humanos y de la carga vírica de la hepatitis C".* Tesis Maestría, España.
  Encontrado noviembre 2007.
  http://www.elai.upm.es/spain/Investiga/GCII/personal/lsantos/LSANTOS.HTMV

- Seung-Hung H. y Javidi, B., 2002. *"Optimum nonlinear composite filter for distortion-intolerant pattern recognition".* Appl. Opt., 41 (11): 2172-2178 p.

- VanderLugt, A., 1964. *"Signal detection by complex filters"*. IEEE Trans. Inf. Theory Theory, IT-10: 139-145 p.

- Watkins C., A. Sadun, y S. Marenka, 1993. *"Modern Image Processing: Warphing, Morphing and Classical Techniques"*. Academic Press, Inc., Cambridge, MA.

- Winder, Steve, 2002. *"Analog and digital filter design*", 2da edición, Newnes, USA, 450 pp.

- Zavala Hamz V. y Álvarez-Borrego J., 1997. *"Circular harmonic filters for Recognition of Marine Microorgranism"*. Applied Optics. 36(2): 484-489 p.

- Zavala Hamz V., Álvarez-Borrego J., Trujillo-Ortíz A., 1996. *"Diffraction Pattems as a Tool to Recognize Copepods"*. Journal of Plankton Research. 18(8):1471-1484 p.

Medios Electrónicos:

I1) http://www.infaimon.com/catalog/catalog.php (consulta septiembre 2008).

I2) http://www.recognitionscience.com/index.htm (consulta diciembre 2008).

I3) http://aertia.com/productos.asp?pid=284&pg=ds (consulta agosto 2008).

I4) http://www.metaemotion.com/es/bienvenido.html (consulta agosto 2008).

I5) http://www.attrasoft.com/default.asp (consulta julio 2008).

I6) http://www.brivas.com/inicio/index.htm (consulta julio 2008).

I7) http://www.sccn.ucsd.edu/~arno/spikenet/ (consulta julio 2008).

I8) http://www.agilemanifesto.org (consulta septiembre 2008).

I9) http://www.extremeprogramming.org/index.html (consulta julio 2008).

Imagen suecia: http://www.dondeviajar.es/files/2008/03/normal_austria_viena1.jpg (consulta octubre 2008).

Imagen fruta: http://wonderfulflowers.biz/catalogo/images/frutas%20y%20chocolates.jpg (consulta octubre 2008).

APÉNDICES

Apéndice A. Encuesta de opinión.

Los resultados de está encuesta no fueron analizados por medios estadísticos su finalidad fue únicamente como guía de requerimientos del sistema a realizar y como soporte a este trabajo de tesis.

Sujetos: Dirigido a quienes han desempeñado el puesto de investigador.

Objetivo: Determinar el uso de herramientas manuales en investigación de campo basadas en identificación de organismos u otros, así como valorar el equipo ocupado como auxiliar de sus tareas.

Nombre y Titulo:

Área de investigación:

Puesto:

1.- ¿En su trabajo de investigación es común la clasificación e identificación taxonómica?

☐ Si ☐ No

Si su respuesta fue “SI” continué por el contrario pase a la pregunta 4.
2.- ¿Que material o equipo ocupa para esta tarea?

3.- Cuanto tiempo tarda aproximadamente en una clasificación.

☐ 5-20 min. ☐ 30-60 min. ☐ 1-2 hrs. ☐ +2

Otros

4.- ¿Sabe o conoce de algún método para hacer más fácil y rápida la tarea de identificación?

☐ Si ☐ No

5.- ¿Sabia usted que existen sistemas ópticos y digitales que agilizan esta tarea?

☐ Si ☐ No

6.- ¿Hace uso de ellos?

☐ Si ☐ No

¿Porque?

7.- ¿Cree usted que si existiera una herramienta computacional lo suficientemente sencilla que le ayudara en su trabajo la utilizaría?

☐ Si ☐ No

¿Porque?

8.- ¿El equipo de computo con el que usted cuenta que versión de sistema operativo tiene?

☐ Windows 98 ☐ Windows Xp ☐ Windows Vista ☐ +2

9.- ¿Conoce la cantidad de memoria de su equipo?, ¿cual es?

10.- Por ultimo, ¿le gustaría participar como testeador de un software de reconocimiento de organismos con muestras de su trabajo de investigación?

GRACIAS.

Apéndice B. Métodos principales.

MatrizFusion.m

```
%       EN ESTA FUNCION FUSIONAREMOS EL FILTRO CON EL FONDO PARA ELLO
%       RECIBIREMOS LOS VALORES DE LOS LIMITES DE LAS IMAGENES DE EL
%       PROGRAMA PRINCIPAL Y LA IMAGEN FONDO,
%       DEVOLVEREMOS LA MATRIZ FUSIONADO Y EN FORMATO DOUBLE
%       RECIBIMOS: ALTO/ANCHO DE LA IMAGEN PROB,ALTO/ANCHO IMAGEN FILTRO
%       MATRIZ DE FONDO (zeros),IMAGEN FILTRO,VALORES DE INICIO Y FIN DE
IMAGENES
%       PROB Y FILTRO


function [zaFusion] = MatrizFusion
(pAltoProb,pAnchoProb,pZa,pMi,pNi,pMf,pNf,pA,pAnchoFil,pAltoFil)
    Fi=1;
    Fj=1;
    for ii=1:pAltoProb
        for jj=1:pAnchoProb
             ZaI=pZa(ii,jj);
             ZaJ=pZa(pMi,pNi);
             if ((ZaI==0)&&(ZaJ==0))
                for m = pMi:pMf
                    for n = pNi:pNf
                        pZa(m,n) = pA(Fi,Fj);
                        if Fj==pAnchoFil
                            Fj=1;
                            break
                        end
                        Fj=Fj+1;
                    end
                    if Fi==pAltoFil
                        Fi=1;
                        break
                    end
                    Fi=Fi+1;
                end
            end
        end
    end
    % fin de for de suma filtro/fondo
    zaFusion = double(pZa);


**********************************************************************
```

MainFftNon_v3.m

```
%%  ESTA FUNCION RECIBIRA LA SUMA DE LOS ESPECTROS, LA IMAGEN PROBLEMA Y
EL
%%  PARAMETRO K; HARA LA CORRELACION DE UNA CON LA OTRA REGRESANDONOS EL
%%  VALOR DE LA CORRELACION LISTO PARA GRAFICAR. APLICAMOS EL VALOR DE
%%  UMBRAL U=MAX*K O U=(MAX*0.1)/(NUMiMG/K) DEPENDIENDO DEL NUMERO DE
```

```
%%  IMAGENES



function [Corr,valLimPico] = MainFftNon_v3 (pSumEspecNon, pB, pK,
pNumTotalImg)

    %coor con el mismo
    esp=pSumEspecNon.*conj(pSumEspecNon);
    lomismo=real(fftshift(ifft2(fftshift(esp))));
    lomismo=abs(lomismo).^2;
    longitud=max(max(lomismo));

    %valor para OperCorrMulti k determinara el umbral limite a buscar
    %puntos
    if (pNumTotalImg==1)
        valLimPico=longitud*pK;
    end
    if (pNumTotalImg>1)
        valLimPico=longitud*(.01);
        valLimPico=(valLimPico./(pNumTotalImg./pK));
    end

    %ecuaciones de la escena
    fb=fftshift(fft2(fftshift(pB))); % Transformada de Fourier de b
    fbmod=abs(fb).^(pK); % Se obtiene el modulo b y se le aplica no
linealidad de 0.1
    fbphase=angle(fb); % Se obtiene la fase de b
    resulb=fbmod.*exp(i*fbphase);% Se recombinan el modulo (con no
linealidad) y la fase

    %correlacionamos a-b (pSumEspecNon-pB)
    espec=resulb.*conj(pSumEspecNon); % Se obtiene el espectro cruzado de
resul y resulb (equivalente a la correlacion en el dominio espacial)
    Corr=real(fftshift(ifft2(fftshift(espec)))); % Se antitransforma,
para obtener la correlacion en dominio espacial
    Corr=Corr.^2;

************************************************************
```

MainFft_v3.m

```
%%FUNCION QUE RECIBE LA FUSION, LA IMAGEN PROBLEMA Y EL PARAMETRO DE
FILTRO
%%LINEAL, REGRESANDO LOS DATOS LISTOS PARA GRAFICAR

function [Corr,valLimPico] = MainFft_v3 (pZaFusion, pB, pOpcFil)
    %iniciamos transformadas
    Fa = fftshift(fft2(fftshift(pZaFusion)));
    Fb = fftshift(fft2(fftshift(pB)));
    %seleccionamos filtro
    if (pOpcFil == 1)
        %correlacionamos cn el mismo
        esp=Fa.*conj(Fa);
        lomismo=real(fftshift(ifft2(fftshift(esp))));
```

```
        lomismo=abs(lomismo).^2;
        longitud=max(max(lomismo));

        Espec = Fb.*conj(Fa);%filtro clásico
    end
    if (pOpcFil == 2)
        pha=angle(Fa);
        %correlacionamos con el mismo
        esp=Fa.*conj(exp(i*pha));
        lomismo=real(fftshift(ifft2(fftshift(esp))));
        lomismo=abs(lomismo).^2;
        longitud=max(max(lomismo));

        Espec=Fb.*conj(exp(i*pha)); %Filtro de fase
    end
    if (pOpcFil == 3)
        tmp=1./Fa;
        alto = size(tmp,1);
        ancho = size(tmp,2);
        for iFin=1:alto
            for jFin=1:ancho
                tmpInf=isinf(tmp(iFin,jFin));%eliminamos INF
                if (tmpInf==1)
                    tmp(iFin,jFin)=0;
                end
            end
        end
        %correlacionamos con el mismo
        esp=Fa.*(tmp);%conj(tmp);
        lomismo=real(fftshift(ifft2(fftshift(esp))));
        lomismo=abs(lomismo).^2;
        longitud=max(max(lomismo));

        Espec=Fb.*(tmp);%Filtro inverso
    end
    %valor para OperCorrMulti k determinara el umbral limite a buscar
puntos
    valLimPico=longitud.*(0.1);
    %correlacion
    Corr = real(fftshift(ifft2(fftshift(Espec))));
    Corr = Corr.^2;
```

****

LoadImageProb_v2.m

```
%EN ESTA FUNCION CARGAREMOS LA(S) IMAGENES
%INDEPENDIENTEMENTE DE SI SON FILTRO O
%PROBLEMA, SI ES 1 O VARIAS

function [imaProbLoad] = LoadImageProb_v2 (pPathImageProb)
    b = imread(pPathImageProb);
    dimenB = size(b);
    tamB = length (dimenB);
    if (tamB==3)
        b = 0.2989 * b(:,:,1) + 0.5870 * b(:,:,2) + 0.1140 * b(:,:,3);
```

```
    end

    b = double(b);
    AltoProb=size(b,1);     %alto de la imagen problema, viene el dato
desde la función
    AnchoProb=size(b,2);    %ancho de la imagen problema, viene el dato
desde la función
    ModAltoProb = mod(AltoProb,2);
    ModAnchoProb = mod(AnchoProb,2)   ;

    if ModAltoProb ~= 0
        b(AltoProb,:)=[];
    end
    if ModAnchoProb ~= 0
        b(:,AnchoProb)=[];
    end
    imaProbLoad = b;
```

****

LoadImageFil_v2.m

```
%EN ESTA FUNCION CARGAREMOS LA(S) IMAGENES
%INDEPENDIENTEMENTE DE SI SON FILTRO O
%PROBLEMA, SI ES 1 O VARIAS

function [imaFilLoad] = LoadImageFil_v2 (pPathImageFil)
    a = imread(pPathImageFil);
    dimenA = size(a);
    tamA = length (dimenA);
    if (tamA==3)
        a = 0.2989 * a(:,:,1) + 0.5870 * a(:,:,2) + 0.1140 * a(:,:,3);
    end

    a=double(a);
    AltoFil = size(a,1);    %obtenemos el alto de la imagen filtro,
numero de renglones
    AnchoFil = size(a,2);   %obtenemos el ancho de la imagen filtro,
numero de columnas
    ModAltoFil = mod(AltoFil,2);
    ModAnchoFil = mod(AnchoFil,2);

     if ModAltoFil ~= 0
        a(AltoFil,:)=[];
    end
    if ModAnchoFil ~= 0
        a(:,AnchoFil)=[];
    end
    imaFilLoad = a;
```

****

ValMaxImg.m

```
%%FUNCION QUE REGRESARA EL VALOR DEL MAXIMO DE LA IMAGEN DE ENTRADA,
```

```
%%UNICAMENTE MANDARA UN VALOR POR ENTRADA, ESTO ES PARA RESULTADOS Y
FINES
%%ESTADISTICOS

function [resValMax] = ValMaxImg(pImg)
    max1=max(max(pImg));
    resValMax=max1;
    resValMax=num2str(resValMax);

**************************************************************
```

SumEspecFilNon_v11.m

```
%% FUNCION QUE RECIBIRA LA MATRIZ ZEROS, VALOR K, NUM TOTAL DE IMAGENS
DEL
%% FILTRO Y LO PRINCIPAL UN CELL ARRAY CON LOS VALORES DE LA FUSION DE
%% TODAS LAS IMAGENES YA CENTRADAS EN UN FONDO NEGRO. REGRESARA EL VALOR
%% EN sumEspecNon=ZaRes DE LA OPERACION DEL FILTRO LISTO PARA
CORRELACIONAR.
%%

function [sumEspecNon]= sumEspecFilNon_v11
(pZa,pZaNew,pArrayTmp,pK,pNumImg)
    pNumImg=pNumImg+1;
    if (pNumImg==1)
        ZaRes=pZa;
    else
        ZaRes=pZaNew;
    end
    fa=fftshift(fft2(fftshift(pArrayTmp)));
    famod=abs(fa).^pK;
    faphase=angle(fa);
    resImg=famod.*exp(i*faphase);
    ZaRes=ZaRes+resImg;
    sumEspecNon = ZaRes;

**************************************************************
```

OperCorrMulti_v23.m

```
%%RECOBIMOS EL VALOR DE LA CORRELACION Y
%%REGRESAMOS LAS COORDENADAS Y LA CANTIDAD DE PICOS ENCONTRADOS EN ELLA

function [numIter,cellCooRes] =
OperCorrMulti_v23(pCorr,pNumImgProb,pValLimPico)
    %cell que gurdara las coordenadas de los picos
    cellCoo=cell(0,0);
    zTmp=zeros(11);
    count=1;
    noMax=0;
    tmpCorr=pCorr;
```

```
divMax=pValLimPico;
while (noMax==0)
    if (count==1)
        maxI=max(max(tmpCorr));
        [xx,yy]=find(maxI==tmpCorr);
        tamX=size(xx,1);
        tamY=size(yy,1);
        if (tamX>1)
            xx=xx(1);
        end
        if (tamY>1)
            yy=yy(1);
        end
    else
        xI=xx-5;
        xF=xx+5;
        yI=yy-5;
        yF=yy+5;
        if ((xI<=0)|(xF<=0)|(yI<=0)|(yF<=0))
            noMax=1;
            break;
        end
        tmpCorr(xI:xF,yI:yF)=zTmp;
        maxI=max(max(tmpCorr));
        [xx,yy]=find(maxI==tmpCorr);
        tamX=size(xx,1);
        tamY=size(yy,1);
        if (tamX>1)
            xx=xx(1);
        end
        if (tamY>1)
            yy=yy(1);
        end
        if (maxI < divMax)
            noMax=1;
            break;
        end
    end
    if (maxI < divMax)
        noMax=1;
        break;
    end
    cellCoo(count,1)=mat2cell(pNumImgProb);
    cellCoo(count,2)=mat2cell(xx);
    cellCoo(count,3)=mat2cell(yy);
    count=count+1;
end
cellCooRes=cell2mat(cellCoo);
numIter=count;
```

****

CreateFigureCorr.m

```
%%FUNCION QUE CREARA LA FIGURA Y LA ALMACENARA EN DISCO, OBTIENE DATOS DE
%%LA CORRELACION, PATH DE TRABAJO, NUM DE IMAGENES, INDEX DE IMAGEN,
%%NOMBRES DE LOS EJES
```

```
function
CreateFigureCorr(pZdata1,pNameX,pNameY,pPathImag,pK,pNumImg,pIndexImgChar)
% Create figure
figure1 = figure('Name','Correlation result',...
    'Color',[0.9412 0.9412 0.9412]);

% Create axes
axes1 =
axes('Parent',figure1,'YDir','reverse','FontWeight','demi','FontSize',11,
...
    'FontName','Eras Light ITC');
view([-37.5 30]);
grid('on');

mesh((pZdata1),'Parent',axes1), axis ij tight;

% Create xlabel
if (pNameX==0)
    xlabel('Axis X');
else
    xlabel(pNameX);
end
% Create ylabel
if (pNameY==0)
    ylabel('Axis Y');
else
    ylabel(pNameY);
end
% Create title
if (pK == 0)
    title('Correlation');
else
    title(['Correlation with ',num2str(pNumImg),' images
filter'],'fontsize',14)
end
PathImagB=[pPathImag,'\tmpCorr',num2str(pIndexImgChar),'.bmp'];
print ('-dbitmap', '-zbuffer', PathImagB);
close;
```

Apéndice C. Documentación NWArray

Matlab Builder NE MWArray Librería de clases.

Namespace: **MathWorks.MATLAB:NET.Arrays**

Las clases en este namespace permiten acceder a arreglos de Matlab de cualquier modulo compilado. Las clases en modo jerarquico son:

Clases

Clase	Descripción
MWArray	MWArray is an abstract class that serves as the root of the MATLAB array class hierarchy. It encapsulates a native MATLAB mxArray and provides a managed API for accessing, formatting, and manipulating the native array.
MWCellArray	MWCellArray derives from MWArray and is the managed representation of the MATLAB cell array. Each element in a cell array is a container that can hold an MWArray or one of its derived types, including another MWCellArray.
MWCharArray	MWCharArray derives from MWArray and is the managed representation of the MATLAB character array. Like its MATLAB equivalent, MWCharArray provides support for string creation and manipulation.
MWIndexArray	MWIndexArray is an abstract class that serves as the root for the MWArray indexing classes. These classes, represent array types that can be used as input parameters to the array indexing operator [].
MWLogicalArray	MWLogicalArray is the managed representation of the MATLAB logical array. Like its MATLAB equivalent, an MWLogicalArray contains only ones and zeros (true/false).
MWNumericArray	MWNumericArray is the managed representation of the MATLAB numeric array types. Like its MATLAB equivalent, it is the default array type used by most of the MATLAB math functions.

MWStructArray	MWStructArray is the managed representation of the MATLAB structure array. Like its MATLAB equivalent, it consists of field values associated with field names.

Enumerations

Enumeration	Description
MWArrayComplexity	MATLAB numeric array complexity enumerator.
MWArrayComponent	MATLAB numeric array component enumerator.
MWArrayType	MMArray type enumeration.
MWNumericType	MATLAB numeric array data type enumerator.

MWArray Class
MWArray Members

MWArray overview

Public Static Fields

MCRAppInitialized	Internal static member variable - Initialization flag

Public Static Properties

NativeGCBlockSize	Static property representing the garbage collector native memory allocation trigger size in bytes.
NativeGCEnabled	Static property representing the native array GC enabled flag.

Public Static Methods

DisposeArray	This dispose method recursively frees the resources of the MWArray instances contained in the Object argument.

Public Static Type Conversions

Implicit Double to MWArray Conversion	Implicit cast from a native double scalar value to an MWNumericArray.
Implicit String to MWArray Conversion	Implicit cast from a native character string to an MWCharArray.

Public Instance Properties

ArrayType	Read only property returning the derived type of the MWArray
Dimensions	Read only property returning a native integer array containing the size of each dimension of the MWArray.
IsCellArray	Returns "true" for an MWCellArray.
IsCharArray	Returns true for an MWCharArray.
IsDisposed	Read only property returning the dispose status of an MWArray
IsEmpty	Returns true if the MWArray is empty.
IsLogicalArray	Returns true for an MWLogicalArray.
IsNumericArray	Returns true for an MWNumericArray.
IsStructArray	Returns true for an MWStructArray.
Item	Generic MATLAB array indexer returning the result as an MWArray
NumberofDimensions	Read only property returning the number of dimensions in the MWArray.
NumberOfElements	Read only property returning the number of elements in the MWArray.

Public Instance Methods

Clone	Makes a deep copy of an MWArray.
Dispose	Releases resources of the MWArray and the native mxArray that it encapsulates.
Equals	Compares two MWArray instances for equality (as defined by mxIsIdentical).
GetHashCode	Returns the hashcode of the MWArray.
GetObjectData	Serialization function.
GetType (inherited from **Object**)	Gets the Type of the current instance.
ToString	Returns a formatted string representing the contents of the MWArray

MWCellArray Class
MWCellArray Members

MWCellArray overview

Public Static Properties

Empty	Read only property returning a writeable version of an empty MWCellArray.

Public Instance Constructors

MWCellArray	Overloaded. MATLAB cell array constructors.

Public Instance Properties

ArrayType (inherited from **MWArray**)	Read only property returning the derived type of the MWArray
Dimensions (inherited from **MWArray**)	Read only property returning a native integer array containing the size of each dimension of the MWArray.

IsCellArray (inherited from **MWArray**)	Returns "true" for an MWCellArray.
IsCharArray (inherited from **MWArray**)	Returns true for an MWCharArray.
IsDisposed (inherited from **MWArray**)	Read only property returning the dispose status of an MWArray
IsEmpty (inherited from **MWArray**)	Returns true if the MWArray is empty.
IsLogicalArray (inherited from **MWArray**)	Returns true for an MWLogicalArray.
IsNumericArray (inherited from **MWArray**)	Returns true for an MWNumericArray.
IsStructArray (inherited from **MWArray**)	Returns true for an MWStructArray.
Item	The MATLAB cell array indexer
NumberofDimensions (inherited from **MWArray**)	Read only property returning the number of dimensions in the MWArray.
NumberOfElements (inherited from **MWArray**)	Read only property returning the number of elements in the MWArray.

Public Instance Methods

Clone	Makes a deep copy of a MATLAB cell array
Dispose (inherited from **MWArray**)	Releases resources of the MWArray and the native mxArray that it encapsulates.
Equals	Compares the current MWCellArray instance with the specified cell array for equality; returning a boolean value.
GetHashCode	Returns the hashcode for the MATLAB cell array
GetObjectData (inherited from **MWArray**)	Serialization function.
GetType (inherited from **Object**)	Gets the Type of the current instance.

ToString	Returns a formatted string representing the MATLAB cell array

MWCharArray Class
MWCharArray Members

MWCharArray overview

Public Static Properties

Empty	Read only property returning a writeable version of an empty MWCharArray.

Public Static Type Conversions

Explicit MWCharArray to Char Conversion	Explicit cast from a MATLAB MWCharArray scalar to a native character value.
Implicit String to MWCharArray Conversion	Implicit cast from a native string to a MATLAB character array.

Public Instance Constructors

MWCharArray	Overloaded. MATLAB character array constructors.

Public Instance Properties

ArrayType (inherited from **MWArray**)	Read only property returning the derived type of the MWArray
Dimensions (inherited from **MWArray**)	Read only property returning a native integer array containing the size of each dimension of the MWArray.
IsCellArray (inherited from **MWArray**)	Returns "true" for an MWCellArray.
IsCharArray (inherited from **MWArray**)	Returns true for an MWCharArray.
IsDisposed (inherited from **MWArray**)	Read only property returning the dispose status of an MWArray
IsEmpty (inherited from **MWArray**)	Returns true if the MWArray is empty.

IsLogicalArray (inherited from **MWArray**)	Returns true for an MWLogicalArray.
IsNumericArray (inherited from **MWArray**)	Returns true for an MWNumericArray.
IsStructArray (inherited from **MWArray**)	Returns true for an MWStructArray.
Item	Overloaded. The MATLAB character array indexer
NumberofDimensions (inherited from **MWArray**)	Read only property returning the number of dimensions in the MWArray.
NumberOfElements (inherited from **MWArray**)	Read only property returning the number of elements in the MWArray.

Public Instance Methods

Clone	Makes a deep copy of a MATLAB character array
Dispose (inherited from **MWArray**)	Releases resources of the MWArray and the native mxArray that it encapsulates.
Equals	Compares two MATLAB character arrays for equality.
GetHashCode	Returns the hashcode for the MATLAB character array
GetObjectData (inherited from **MWArray**)	Serialization function.
GetType (inherited from **Object**)	Gets the Type of the current instance.
ToArray	Returns a native string array with the same dimensions and values as the character array.
ToString	Returns a native string representing the character array

MWIndexArray Class
MWIndexArray Members

MWIndexArray overview

Public Static Type Conversions

Implicit Byte[] to MWIndexArray Conversion	Implicit cast from a native byte array to a MATLAB index array.
Implicit Int16[] to MWIndexArray Conversion	Implicit cast from a native short array to a MATLAB index array.
Implicit Int32 to MWIndexArray Conversion	Implicit cast from an integer to a MATLAB index array.
Implicit Int32[] to MWIndexArray Conversion	Implicit cast from a native integer array to a MATLAB index array.
Implicit Int64[] to MWIndexArray Conversion	Implicit cast from a native long array to a MATLAB index array.

Public Instance Properties

ArrayType (inherited from **MWArray**)	Read only property returning the derived type of the MWArray
Dimensions (inherited from **MWArray**)	Read only property returning a native integer array containing the size of each dimension of the MWArray.
IsCellArray (inherited from **MWArray**)	Returns "true" for an MWCellArray.
IsCharArray (inherited from **MWArray**)	Returns true for an MWCharArray.
IsDisposed (inherited from **MWArray**)	Read only property returning the dispose status of an MWArray
IsEmpty (inherited from **MWArray**)	Returns true if the MWArray is empty.
IsLogicalArray (inherited from **MWArray**)	Returns true for an MWLogicalArray.
IsNumericArray (inherited from **MWArray**)	Returns true for an MWNumericArray.
IsSparse	Returns true if the array is a sparse array.
IsStructArray (inherited from **MWArray**)	Returns true for an MWStructArray.

Item (inherited from **MWArray**)	Generic MATLAB array indexer returning the result as an MWArray
NonZeroMaxStorage	Returns the number of storage locations allocated for the nonzero elements of a sparse matrix.
NumberofDimensions (inherited from **MWArray**)	Read only property returning the number of dimensions in the MWArray.
NumberOfElements (inherited from **MWArray**)	Read only property returning the number of elements in the MWArray.

Public Instance Methods

Clone	Makes a deep copy of a MATLAB index array.
Dispose (inherited from **MWArray**)	Releases resources of the MWArray and the native mxArray that it encapsulates.
Equals	Compares two MATLAB index arrays for equality.
GetHashCode	Returns the hashcode for the specified MATLAB index array
GetObjectData (inherited from **MWArray**)	Serialization function.
GetType (inherited from **Object**)	Gets the Type of the current instance.
ToString	Returns a native string representing the MATLAB index array

MWLogicalArray Class
MWLogicalArray Members

MWLogicalArray overview

Public Static Properties

Empty	Read only property returning a writeable version

	of an empty MWLogicalArray.

Public Static Methods

MakeSparse	Overloaded. Constructs a sparse logical matrix.

Public Static Type Conversions

Implicit MWLogicalArray to Boolean Conversion	Implicit cast from an MWLogicalArray to a boolean value.
Implicit Boolean to MWLogicalArray Conversion	Implicit cast from a native boolean scalar to an MWLogicalArray.

Public Instance Constructors

MWLogicalArray	Overloaded. MATLAB logical array constructors.

Public Instance Properties

ArrayType (inherited from **MWArray**)	Read only property returning the derived type of the MWArray
Dimensions (inherited from **MWArray**)	Read only property returning a native integer array containing the size of each dimension of the MWArray.
IsCellArray (inherited from **MWArray**)	Returns "true" for an MWCellArray.
IsCharArray (inherited from **MWArray**)	Returns true for an MWCharArray.
IsDisposed (inherited from **MWArray**)	Read only property returning the dispose status of an MWArray
IsEmpty (inherited from **MWArray**)	Returns true if the MWArray is empty.
IsLogicalArray (inherited from **MWArray**)	Returns true for an MWLogicalArray.
IsNumericArray (inherited from **MWArray**)	Returns true for an MWNumericArray.
IsScalar	Returns true if the array is a logical scalar.

IsSparse (inherited from **MWIndexArray**)	Returns true if the array is a sparse array.
IsStructArray (inherited from **MWArray**)	Returns true for an MWStructArray.
Item	Overloaded. The MATLAB logical array indexer
NonZeroMaxStorage (inherited from **MWIndexArray**)	Returns the number of storage locations allocated for the nonzero elements of a sparse matrix.
NumberofDimensions (inherited from **MWArray**)	Read only property returning the number of dimensions in the MWArray.
NumberOfElements (inherited from **MWArray**)	Read only property returning the number of elements in the MWArray.

Public Instance Methods

Clone	Makes a deep copy of a logical array
Dispose (inherited from **MWArray**)	Releases resources of the MWArray and the native mxArray that it encapsulates.
Equals	Compares two logical arrays for equality returning a boolean value.
GetHashCode	Returns the hashcode for the logical array
GetObjectData (inherited from **MWArray**)	Serialization function.
GetType (inherited from **Object**)	Gets the Type of the current instance.
ToArray	Converts the MWLogicalArray into a native boolean array with the same dimensions and values as the logical array.
ToString	Returns a string representing the logical array.
ToVector	Returns a copy of the elements in the logical array as a one dimension native boolean array

MWNumericArray Class
MWNumericArray Members

MWNumericArray overview

Public Static Fields

Inf	A MATLAB numeric array constant representing infinity.
NaN	MATLAB numeric array constant representing a non-number.

Public Static Properties

Empty	Read-only property returning a writeable version of an empty MWNumericArray.
FloatingPointAccuracy	Read-only property returning the distance from 1.0 to the next largest floating-point number. This property is a measure of the MATLAB floating-point accuracy.

Public Static Methods

MakeSparse	Overloaded. Constructs a sparse numeric matrix.

Public Static Type Conversions

Explicit MWNumericArray to Byte Conversion	Implicit cast from a MATLAB uint8 numeric scalar to a byte value.
Explicit MWNumericArray to Double Conversion	Implicit cast from a MATLAB double numeric scalar to a double value.
Explicit MWNumericArray to Int16 Conversion	Implicit cast from a MATLAB Int16 numeric scalar to a short value.
Explicit MWNumericArray to Int32 Conversion	Implicit cast from a MATLAB Int32 numeric scalar to an integer value.
Explicit MWNumericArray to Int64 Conversion	Implicit cast from a MATLAB Int64 numeric scalar to a long value.

Explicit MWNumericArray to Single Conversion	Implicit cast from a MATLAB single numeric scalar to a floating point value.
Implicit Array to MWNumericArray Conversion	Implicit cast from a generic native array to a MATLAB numeric array of the same numeric type.
Implicit Byte to MWNumericArray Conversion	Implicit cast from a native byte value to a MATLAB uint8 numeric scalar.
Implicit Byte[] to MWNumericArray Conversion	Implicit cast from a one dimensional native array of byte values to a MATLAB uint8 real numeric array.
Implicit Double to MWNumericArray Conversion	Implicit cast from a native double scalar to a MATLAB double numeric scalar.
Implicit Double[] to MWNumericArray Conversion	Implicit cast from a one dimensional native array of double values to a MATLAB double real numeric array.
Implicit Int16 to MWNumericArray Conversion	Implicit cast from a native short value to a MATLAB int16 numeric scalar.
Implicit Int16[] to MWNumericArray Conversion	Implicit cast from a one dimensional native array of short values to a MATLAB Int16 real numeric array.
Implicit Int32 to MWNumericArray Conversion	Implicit cast from a native integer scalar to a MATLAB double numeric scalar.
Implicit Int32[] to MWNumericArray Conversion	Implicit cast from a one dimensional native array of integer values to a MATLAB double real numeric array.
Implicit Int64 to MWNumericArray Conversion	Implicit cast from a native long scalar to a MATLAB Int64 numeric scalar.
Implicit Int64[] to MWNumericArray Conversion	Implicit cast from a one dimensional native array of long values to a MATLAB Int64 real numeric array.
Implicit Single to MWNumericArray Conversion	Implicit cast from a native floating point scalar to a MATLAB single numeric scalar.

Implicit Single[] to MWNumericArray Conversion	Implicit cast from a one dimensional native array of floating point values to a MATLAB single real numeric array.

Public Instance Constructors

MWNumericArray	Overloaded. MATLAB numeric array constructors.

Public Instance Properties

ArrayType (inherited from **MWArray**)	Read only property returning the derived type of the MWArray
Dimensions (inherited from **MWArray**)	Read only property returning a native integer array containing the size of each dimension of the MWArray.
IsByte	Returns true if the MATLAB numeric array contains byte (unsigned int) data types.
IsCellArray (inherited from **MWArray**)	Returns "true" for an MWCellArray.
IsCharArray (inherited from **MWArray**)	Returns true for an MWCharArray.
IsComplex	Returns true if the MATLAB numeric array contains complex data.
IsDisposed (inherited from **MWArray**)	Read only property returning the dispose status of an MWArray
IsDouble	Returns true if the MATLAB numeric array contains double data types.
IsEmpty (inherited from **MWArray**)	Returns true if the MWArray is empty.
IsFloat	Returns true if the MATLAB numeric array contains floating point data types.
IsInteger	Returns true if the MATLAB numeric array contains integer data types.
IsLogicalArray (inherited from **MWArray**)	Returns true for an MWLogicalArray.

IsLong	Returns true if the MATLAB numeric array contains long data types.
IsNumericArray (inherited from **MWArray**)	Returns true for an MWNumericArray.
IsShort	Returns true if the MATLAB numeric array contains short data types.
IsSparse (inherited from **MWIndexArray**)	Returns true if the array is a sparse array.
IsStructArray (inherited from **MWArray**)	Returns true for an MWStructArray.
Item	Overloaded. MATLAB real numeric array real component indexer
NonZeroMaxStorage (inherited from **MWIndexArray**)	Returns the number of storage locations allocated for the nonzero elements of a sparse matrix.
NumberofDimensions (inherited from **MWArray**)	Read only property returning the number of dimensions in the MWArray.
NumberOfElements (inherited from **MWArray**)	Read only property returning the number of elements in the MWArray.
NumericType	Returns the data type of the MATLAB numeric array.

Public Instance Methods

Clone	Makes a deep copy of a MATLAB numeric array.
Dispose (inherited from **MWArray**)	Releases resources of the MWArray and the native mxArray that it encapsulates.
Equals	Compares two MATLAB numeric arrays for equality.
GetHashCode	Returns the hashcode for the MATLAB numeric array

GetObjectData (inherited from **MWArray**)	Serialization function.
GetType (inherited from **Object**)	Gets the Type of the current instance.
ToArray	Returns the requested (real/imaginary) component of a MATLAB numeric array as a native array with the same dimensions as the numeric array.
ToScalarByte	Converts the MATLAB uint8 numeric scalar to a byte value.
ToScalarDouble	Converts the MATLAB double numeric scalar to a double value.
ToScalarFloat	Converts the MATLAB single numeric scalar to a floating point value.
ToScalarInteger	Converts the MATLAB int32 numeric scalar to an integer value.
ToScalarLong	Converts the MATLAB int64 numeric scalar to a long value.
ToScalarShort	Converts the MATLAB int16 numeric scalar to a short value.
ToString	Returns a string representing the contents of the numeric array
ToVector	Returns the requested (real/imaginary) component of a MATLAB numeric array as a native array.

MWStructArray Class
MWStructArray Members

MWStructArray overview

Public Static Properties

Empty	Read only property returning a writeable version of an empty MWStructArray.

Public Instance Constructors

MWStructArray	Overloaded. MATLAB structure array constructors.

Public Instance Properties

ArrayType (inherited from **MWArray**)	Read only property returning the derived type of the MWArray
Dimensions (inherited from **MWArray**)	Read only property returning a native integer array containing the size of each dimension of the MWArray.
FieldNames	Returns the field names of the MATLAB structure array as a native array of strings.
IsCellArray (inherited from **MWArray**)	Returns "true" for an MWCellArray.
IsCharArray (inherited from **MWArray**)	Returns true for an MWCharArray.
IsDisposed (inherited from **MWArray**)	Read only property returning the dispose status of an MWArray
IsEmpty (inherited from **MWArray**)	Returns true if the MWArray is empty.
IsLogicalArray (inherited from **MWArray**)	Returns true for an MWLogicalArray.
IsNumericArray (inherited from **MWArray**)	Returns true for an MWNumericArray.
IsStructArray (inherited from **MWArray**)	Returns true for an MWStructArray.
Item	Overloaded. The MATLAB structure array indexer
NumberofDimensions (inherited from **MWArray**)	Read only property returning the number of dimensions in the MWArray.
NumberOfElements (inherited from **MWArray**)	Read only property returning the number of elements in the MWArray.
NumberOfFields	Returns the number of fields in the MATLAB

	structure array.

Public Instance Methods

Clone	Makes a deep copy of an MWStructArray
Dispose (inherited from **MWArray**)	Releases resources of the MWArray and the native mxArray that it encapsulates.
Equals	Compares two MATLAB structure arrays for equality returning a boolean value.
GetField	Returns the contents of the specified field in the structure array.
GetHashCode	Returns the hashcode for the MWStructArray
GetObjectData (inherited from **MWArray**)	Serialization function.
GetType (inherited from **Object**)	Gets the Type of the current instance.
IsField	Returns true if the specified field is the name of a field in the structure array.
RemoveField	Removes the specified field from the structure array.
SetField	Sets the contents of the specified field in the structure array.
ToString	Returns a string representing the MATLAB structure array

MWArrayComplexity Enumeration

Members

Member Name	Description
Real	The numeric array is real.
Complex	The numeric array is complex.

MWArrayComponent Enumeration

MATLAB numeric array component enumerator.

public enum MWArrayComponent

Members

Member Name	Description
Real	Real Part.
Imaginary	Imaginary Part.

MWArrayType Enumeration

MMArray type enumeration.

public enum MWArrayType

Members

Member Name	Description
Array	Root array type.
Index	Cell array type.
Numeric	Numeric array type.
Logical	Logical array type.
Character	Character array type.
Cell	Cell array type.
Structure	Structure array type.

MWNumericType Enumeration

MATLAB numeric array data type enumerator.

public enum MWNumericType

Members

Member Name	Description
Double	Double array type.
Single	Single array type.
Int8	INT8 array type.
UInt8	UInt8 array type.
Int16	Int16 array type.
UInt16	UInt16 array type.
Int32	Int32 array type.
UInt32	UInt32 array type.
Int64	Int64 array type.
UInt64	UINT64 array type.

http://www.mathworks.com/access/helpdesk/help/toolbox/dotnetbuilder/