Cálculo de velocidades superficiales de una parcela de agua a partir de imágenes de satélite empleando la orientación a objetos

Universidad Nacional Autónoma de México
División de Estudios de Posgrado
Facultad de Ingeniería
Departamento de Ingeniería Informática
Maestría en Ingeniería Informática
Maestría en Ingeniería (Eléctrica)
Gabriel Alejandro López Morteo

Tutor: Dr. Román Alvarez Béjar

Febrero de 1999

Universidad Nacional Autónoma de México

División de Estudios de Posgrado

Facultad de Ingeniería

Departamento de Ingeniería Informática

Maestría en Ingeniería Informática

Ocean. Gabriel Alejandro López Morteo

Cálculo de velocidades superficiales de una parcela de agua a partir de imágenes de satélite empleando la orientación a objetos

Miércoles 10 de Febrero de 1999

Agradecimientos

Por regla general, un trabajo de investigación es el producto del trabajo en colaboración de una gran variedad de personas que aportan lo mejor de sí de manera desinteresada.

Es por ello que deseo agradecer muy fervientemente a las siguientes personas que hicieron que esta tesis fuera una realidad:

- A mi tutor el Dr. Román Alvarez Béjar por su comprensión y orientación durante los difíciles tiempos en que se desarrolló este trabajo. Gracias.
- A mis sinodales: Dr. Jesús Savage, Dra. Hanna Oktaba, Dra. Ana María Vázquez y el Ing. Sergio Beltrán por sus atinadas sugerencias y excelente disposición para la revisión de este trabajo. Gracias.
- Al Ing. Sergio F. Beltrán López, pues además de brindarme su apoyo y su confianza de que esta aventura de la Maestría llegaría a buen fin. Sin mencionar lo más preciado que me brindó en todo este tiempo: su amistad y respeto. Gracias.
- Al personal del Laboratorio de Sensores Remotos del Instituto de Geografía por su incondicional ayuda y las facilidades prestadas para el uso de la paquetería especializada. En especial deseo nombrar a la señorita Olivia por haberme facilitado las imágenes para esta tesis. A todos ellos Gracias.
- A mis compañeros de generación por su amistad y confianza y por haber crecido juntos como personas y académicos durante todos estos años. Gracias.
- A mis profesores que lejos de limitarse a la simple enseñanza de su asignatura, moldearon mi carrera y me motivaron a querer ser como ellos. Gracias
- A mi familia que con su ejemplo y ganas de superarse me enseñaron a creer en mi y a lograr los proyectos que me he propuesto en la vida. Gracias, muchas gracias.
- A mi esposa Mariana, quien es y ha sido el motor que me impulsó a continuar con nuestras aspiraciones a pesar de todas las adversidades que hemos encontrado en el camino. Gracias por estar siempre aquí.

Indice

1 INTRODUCCIÓN 1
1.1 OBJETIVOS 2
2 ANTECEDENTES 3
2.1 ANTECEDENTES GENERALES 3
2.2 ANTECEDENTES PARTICULARES 4
3 METODOLOGÍA 7
3.1 PARA EL CÁLCULO DEL CAMPO DE VELOCIDADES SUPERFICIALES 7
3.1.1 Obtención de las imágenes 7
3.1.2 Metodología del cálculo de velocidades superficiales 9
3.2 PARA EL ANÁLISIS Y DISEÑO DEL PROGRAMA EMPLEANDO LA ORIENTACIÓN A OBJETOS 14
3.2.1 La metodología orientada a objetos empleada en el análisis y diseño del programa . 14
3.2.2 El lenguaje de programación y sus requerimientos 14
RESULTADOS 16
3.2.3 Análisis y diseño 16
3.2.3.1 Los primeros pasos: definición del problema 16
3.2.4 Análisis y diseño del problema 18
3.2.4.1 Análisis y diseño de la aplicación 18
3.2.4.1.1 Diagrama de uso: principal 19
3.2.4.1.2 Diagrama de uso: Captura de la imagen 20
3.2.4.1.3 Diagrama de uso: Arreglo digital 21
Diagrama de uso: Cálculo de velocidades 22
3.2.4.1.5 Diagrama de uso: Presentación de resultados 23
3.2.4.1.6 Diagrama de secuencia: Selección de fecha y hora 24
3.2.4.1.7 Diagrama de secuencia: Captura y proceso 25
Diagrama de clases inicial 26
Diagrama de clases final 27
3.3 PROGRAMACIÓN 28
3.3.1 CLASE AVHRR_applet.java 28
3.3.2 CLASE Procesa_imagen.java 28
3.3.3 CLASE DebugWin.java 36
3.4 CÁLCULO DEL CAMPO DE VELOCIDAD SUPERFICIAL DEL AGUA DE MAR 37
3.4.1 Obtención de las imágenes 37
3.4.2 Interfase para el usuario 39
3.4.3 Resultados de la ejecución del programa 40
3.4.4 Desempeño del programa 48
4 DISCUSIONES 50
4.1 JAVA COMO LENGUAJE DE DESARROLLO 50
4.2 LA TECNOLOGÍA ORIENTADA A OBJETOS Y SUS APLICACIONES 52
4.2.1 Comparación práctica entre el análisis y diseño en la programación por procedimientos y la orientación a objetos 52
4.2.2 Estudios recientes relacionados con este trabajo 55
4.2.3 Conclusiones 57
5 REFERENCIAS 58

Indice de tablas

Tabla 1. Características de las bandas de recepción de la región espectral del sensor AVHRR-NOAA. Tomado de Chuvieco, 1990.....3

Tabla 2. Relación de archivos, fechas, horas y bandas de muestreo de las imágenes AVHRR seleccionadas para este estudio. Todas ellas corresponden a una región del pacífico occidental de la República Mexicana de aproximadamente 200 kilómetros por lado frente a las costas de Lázaro Cárdenas, Michoacán. Entre el 11 de diciembre de 1996 y el 24 de enero de 1997....38

Indice de figuras

Figura 1. Relación geométrica entre el mosaico patrón y el mosaico de búsqueda, en donde el centro del mosaico patrón corresponde al centro del mosaico de búsqueda en la subsección de la imagen 2.....11

Figura 2. Delimitación del área total de correlación. El mosaico patrón de la primer imagen se recorre una cierta cantidad de pixeles en la segunda imagen, formando así el área total de correlación.....13

Figura 3. Diagrama de uso principal que muestra los componentes principales del programa.....19

Figura 4. Diagrama de uso captura de la imagen.....20

Figura 5. Diagrama de uso Arreglo digital de las imágenes.....21

Figura 6. Diagrama de uso del cálculo de las velocidades superficiales.....22

Figura 7. Diagrama de uso Presentación de resultados.....23

Figura 8. Diagrama de secuencia selección de fecha y hora.....24

Figura 9. Diagrama de secuencia captura y proceso.....25

Figura 10. Diagrama de clases del sistema compuesta por 5 elementos principales.....26

Figura 11. Diagrama de clases final compuesto por 3 elementos principales.....27

Figura 12. Página HTML mediante la cual el usuario interactua con el sistema al seleccionar las imágenes a visualizar, así como, los parámetros específicos para dichas imágenes.....39

Figura 13. Ejecución del programa con el par de imágenes de control, con la longitud de la caja patrón de 11 pixeles y la longitud de la caja de búsqueda de 5 pixeles. Se observa que el programa detecta el desfasamiento de los cuadrados negros.....41

Figura 14. La misma corrida que la anterior pero invirtiendo el orden entre las imágenes. Se observa nuevamente que el sistema detecta el desfasamiento entre los cuadrados negros, pero registra solamente la dirección y no el sentido del mismo.....42

Figura 15 La longitud de la caja de búsqueda es de 21 pixeles mientras que la de la caja patron es de 7 pixeles. Se registra adecuadamente el desfasamiento pero se muestra un comportamiento errático al ser estos valores demasiado altos.....43

Figura 16 Se observa el mismo comportamiento al invertir el orden de las imágenes sin detectar aún el sentido del desfasamiento.....44

Resumen

Empleando una metodología de la orientación a objetos, se desarrolló un programa que calcula gráfica el campo de velocidades superficiales de una parcela de agua. Esta información se obtuvo a partir de un par de imágenes de satélite de temperatura superficial (SST), las cuales se presentan como mapas de colores de temperatura, que cumplieran con las características de cubrir exactamente la misma área geográfica; estar razonablemente libres de cobertura nubosa y no tener más de 12 horas de diferencia. El programa emplea la metodología conocida como correlación cruzada para encontrar un índice de varianza entre pequeñas ventanas de las dos imágenes, con el propósito de encontrar sus disimilitudes de color en el espacio cubierto por estas ventanas y de esta manera descubrir el movimiento de los esquemas de colores. Este movimiento indica el desplazamiento en espacio de una región de agua superficial con cierta temperatura. A partir de estos desplazamientos se encuentran la dirección y magnitud relativa del movimiento de la parcela de agua entre el tiempo transcurrido entre las dos imágenes y se grafican. Para ello se seleccionó el lenguaje de programación Java con el propósito de que la aplicación fuera accesible a cualquier usuario potencial que tenga acceso a Internet; que el cálculo de velocidades se realice en tiempo real en el equipo del usuario; y por supuesto, aprovechar un lenguaje completamente orientado a objetos y sus metodologías relacionadas.

1 INTRODUCCIÓN

Cada vez más, los sistemas y equipos de cómputo ofrecen una mayor capacidad de realizar operaciones por unidad de tiempo a un costo menor, lo que aumenta el radio de difusión y empleo de este tipo de sistemas fuera de las instituciones académicas.

De esta manera, se genera un número creciente de usuarios potenciales pertenecientes a diversos sectores productivos que se ven beneficiados por un acceso fácil y rápido a este tipo de recursos.

Pero para que esto ocurra de una manera eficiente, es necesario que los profesionales de la informática simplifiquen mediante software el acceso a los recursos del equipo. Proveyendo al usuario de herramientas simples y poderosas que sean fáciles de adquirir, rápidas de aprender y que tengan una capacidad de respuesta en tiempo real razonablemente buena de acuerdo a sus necesidades específicas.

Al aumentar el espectro de usuarios potenciales de un sistema informático, se contempla el acceso a usuarios que no se encuentren físicamente en donde esté ubicado el sistema de cómputo, sino que se localizan en una oficina alterna, un edificio distinto, una región o país diferente, en el aire entre México y Europa o a mitad del Océano Pacífico en busca de pesca. Siendo el reto el elaborar un sistema informático que pueda ser utilizado en un esquema remoto-local por cualquiera de estos usuarios, encontrando el equilibrio entre costo del equipo y software y el rendimiento obtenido.

Mediante esquemas tradicionales, para realizar lo anterior se requería de un equipo centralizado de gran capacidad (y por lo tanto precio), acceso remoto a éstos equipos mediante computadoras de mediana capacidad y sobre todo un conocimiento profundo en el empleo del equipo, el sistema operativo, el ambiente de trabajo, la utilería a emplear y las herramientas de redes.

Además de algo muy importante: un conocimiento experimentado de los resultados de la interacción con el sistema para su asimilación y empleo. Todo lo anterior a un costo que no cualquiera puede pagar.

Pero los tiempos cambian y las tecnologías no son la excepción.

Ahora se cuenta con un acceso a redes rápido que además de económico, es accesible al público en general. Los equipos y sus sistemas operativos tienden a disminuir su precio y complejidad de uso. Las herramientas para manipular lo anterior tienen una característica en común: cualquier persona, sí, cualquier persona puede utilizarlos con un mínimo de entrenamiento, agilizando así la comprensión de los resultados arrojados por el sistema.

Entonces, las circunstancias están dadas para desarrollar proyectos que a la par que empleen las nuevas tecnologías, son accesibles en concepto y forma al usuario final.

La visualización científica auxiliada por computadora presenta una alternativa viable para lograr lo anterior, exponiendo al usuario de la información a resultados visuales y comprensibles, productos de sistemas informáticos y de cómputo complejos.

En el campo de la visualización científica, el estudio de sistemas naturales requiere en gran medida, de una base metodológica y tecnológica completa para producir resultados confiables y precisos. Aunque no es el único reto que se debe cumplir, ya que también los resultados deben ser claros y comprensibles a la vez que accesibles a la comunidad científica y, ¿por qué no?, al público en general. Rompiendo antiguas y nuevas barreras que impiden que el nuevo conocimiento se divulgue y utilice fuera del ámbito académico e industrial a gran escala.

Tal es el caso del estudio de procesos naturales en el área de las Ciencias de la Tierra, específicamente en lo que se refiere a la oceanología aplicada a la percepción remota. Siendo ésta un área cuyo material de estudio proviene de imágenes y que a la vez, los resultados de las investigaciones son en su mayoría imágenes, existe un abismo entre la presentación de dichas imágenes y la comprensión plena de la información que contiene. Especialmente para aquellos que no son expertos en el tema ni mucho menos.

Aunado a esto, se requiere de una enorme especialización por parte de los investigadores del área en aspectos que van desde los métodos de obtención de las imágenes, el fundamente matemático para su procesamiento, hasta un fuerte conocimiento informático y de cómputo necesario para llevar a cabo la manipulación de la imagen, de la información numérica que ella contiene, de los modelos físicos o matemáticos que se deseen emplear, de la presentación final del resultado y hasta de los medios electrónicos que se deseen emplear para su publicación y distribución.

Afortunadamente, la tecnología informática se encuentra en un nivel en donde todo lo anterior puede ser realizado con relativa facilidad: el equipo de cómputo cada vez tiene mayores capacidades a un precio accesible; la metodología de diseño y evaluación de problemas complejos para la elaboración de software ha encontrado un gran aliado en la orientación a objetos; la codificación del diseño del software se apoya en lenguajes de programación robustos, de gran capacidad y cada vez más integrales como son el lenguaje C, C++ y Java; los medios de publicación, exposición y divulgación cada vez son más accesibles y cuentan con una cobertura regional, continental y hasta global como lo son la videoconferencia, los sistemas electrónicos de información y ese conjunto de herramientas para todo uso que es Internet.

Dadas estas características, se plantea el presente proyecto de investigación desde el punto de vista que integre lo mejor de diferentes áreas de la informática, partiendo del estudio matemático y metodológico de procesos naturales mediante sensores remotos, pasando por la comprensión del problema mediante la tecnología orientada a objetos, la optimización en el empleo de recursos de cómputo con herramientas que permitan emplear distintos equipos para el proceso de la información en tiempo real, los métodos de visualización que faciliten la comprensión de los resultados, hasta los mecanismos de distribución electrónica de todo lo anterior. Teniendo siempre presente que todo esto debe de ser comprensible y de fácil acceso para el usuario final: aquel o aquellos que pueden utilizar la información directamente en su entorno productivo.

1.1 Objetivos

- Obtener el campo de velocidades superficiales de una parcela de agua oceánica mediante imágenes de temperatura superficial provenientes de sensores remotos mediante un sistema orientado a objetos.

- Hacer que los resultados sean comprensibles a la vez que accesibles para el usuario final.

- Mostrar como la tecnología orientada a objetos maneja de una manera eficiente la complejidad del problema.

- Demostrar que la tecnología informática resuelve problemas complejos de una manera integral y práctica.

2 ANTECEDENTES

2.1 ANTECEDENTES GENERALES

Existen numerosos trabajos de investigación aplicados a la oceanología a partir de imágenes obtenidas con sensores remotos. Muchos de estos trabajos están basados en imágenes satelitales del tipo AVHRR (*Advanced Very High Resolution Radiometer*) obtenidos con satélites de la serie NOAA[1]. Cuyas características espectrales se muestran en la tabla 1.

Tabla 1. Características de las bandas de recepción de la región espectral del sensor AVHRR-NOAA. Tomado de Chuvieco, 1990.

Banda	Amplitud (μm)	Región Espectral
1	0.58-0.68	Rojo
2	0.72-1.1	Infrarrojo Cercano
3	3.55-3.93	Infrarrojo Medio
4	10.30-11.30	Infrarrojo Térmico
5	11.50-12.50	Infrarrojo Térmico

Estos satélites pueden obtener información tanto en la banda de la luz visible, como en diferentes tipos de infrarrojo, pudiendo calcular la temperatura superficial del agua de mar mediante metodología específica aplicada a imágenes de temperatura superficial del mar (SST).

Desgraciadamente su resolución espacial, aunque es grande, no es muy detallada con valores máximos de 1 km. x 1 km. en el nadir y una precisión de 0.5°C (Baransky y Mruglsky, 1989; Cantón y Hernández, 1991).

Las imágenes que van a emplear en este estudio, se forman a partir de las bandas 4 y 5, ya que éstas corresponden al espectro de transmisión de las frecuencias del infrarrojo térmico de la capa superficial del agua en general.

El resultado del procesamiento digital de esta señal es una imagen formada por un mapa de falso color, en donde cada tonalidad representa una temperatura en particular.

[1]National Oceanic and Atmospheric Administration

- Mostrar como la tecnología orientada a objetos maneja de una manera eficiente la complejidad del problema.
- Demostrar que la tecnología informática resuelve problemas complejos de una manera integral y práctica.

2 ANTECEDENTES

2.1 ANTECEDENTES GENERALES

Existen numerosos trabajos de investigación aplicados a la oceanología a partir de imágenes obtenidas con sensores remotos. Muchos de estos trabajos están basados en imágenes satelitales del tipo AVHRR (*Advanced Very High Resolution Radiometer*) obtenidos con satélites de la serie NOAA[1]. Cuyas características espectrales se muestran en la tabla 1.

Tabla 1. Características de las bandas de recepción de la región espectral del sensor AVHRR-NOAA. Tomado de Chuvieco, 1990.

Banda	Amplitud (μm)	Región Espectral
1	0.58-0.68	Rojo
2	0.72-1.1	Infrarrojo Cercano
3	3.55-3.93	Infrarrojo Medio
4	10.30-11.30	Infrarrojo Térmico
5	11.50-12.50	Infrarrojo Térmico

Estos satélites pueden obtener información tanto en la banda de la luz visible, como en diferentes tipos de infrarrojo, pudiendo calcular la temperatura superficial del agua de mar mediante metodología específica aplicada a imágenes de temperatura superficial del mar (SST).

Desgraciadamente su resolución espacial, aunque es grande, no es muy detallada con valores máximos de 1 km. x 1 km. en el nadir y una precisión de 0.5°C (Baransky y Mruglsky, 1989; Cantón y Hernández, 1991).

Las imágenes que van a emplear en este estudio, se forman a partir de las bandas 4 y 5, ya que éstas corresponden al espectro de transmisión de las frecuencias del infrarrojo térmico de la capa superficial del agua en general.

El resultado del procesamiento digital de esta señal es una imagen formada por un mapa de falso color, en donde cada tonalidad representa una temperatura en particular.

[1]National Oceanic and Atmospheric Administration

Desafortunadamente, este tipo de sensores no son capaces de atravesar la cobertura nubosa, por lo que la obtención de los datos de irradianza está sujeta al comportamiento del estado del tiempo regional.

Una tendencia que es claramente observable en los trabajos de oceanografía por satélite es que son en su mayoría, completamente descriptivos acerca de la distribución de procesos oceanográficos importantes como lo son las surgencias, corrientes superficiales a meso-escala y macro-escala, convergencias, etc. (v.g. Castagné, et. al., 1986; Szekielda y McGinnis, 1987; Szekielda, Degens y McGinnis 1987; Hernández, et. al., 1989; LLinas, Rueda y Perez-Martell, 1989; Baranski y Mrugalski, 1989; Maso, La Violette y Tintore, 1989; Huang y Robinson, 1995; entre otros).

Aunque en estos trabajos se hace un alto énfasis en las técnicas informáticas empleadas, no se hacen muchos estudios serios que presenten un punto de vista contrario, es decir ver a la informática como un área de estudio y desarrollo dentro de la oceanología.

La aplicación de elementos de la informática junto con la tecnología desarrollada sobre sensores remotos, ofrecen al investigador un medio más viable y rápido de obtener resultados diversos y confiables a partir de imágenes satelitales del océano y la zona costera en estudios temporales y multitemporales.

2.2 ANTECEDENTES PARTICULARES

El estudio y monitoreo de la temperatura superficial del agua de mar, empleando sensores remotos, no es un tema nuevo. De hecho, éste comenzó con el lanzamiento del primer satélite artificial de la serie NOAA, por parte de la National Oceanic and Atmospheric Administration en el año de 1978, con sensores capaces de percibir irradianza en el espectro del infrarrojo cercano (NOAA-6).

A partir de entonces, se realizaron varios intentos por obtener velocidades y direcciones de variación de las parcelas de agua o cobertura nubosa empleando al menos dos imágenes satelitales sucesivas (McMillin, 1975; Simpson y Gobat, 1994).

Uno de ellos data del año 1971 y fue elaborado por Lesse et.al., ellos desarrollaron un método automatizado para obtener el movimiento de nubes mediante el empleo de correlación cruzada en ventanas similares de dos imágenes sucesivas (con una diferencia de 24 minutos), tomadas por el satélite geoestacionario Applications Technology Satellite ATS-I, el cual obtiene imágenes pancromáticas del hemisferio oeste de la tierra. Básicamente, a cada par de imágenes les aplicaron técnicas de corrección geométrica para realizar un mapeo de la imagen en una proyección Mercator, después se aplicó el método de correlación cruzada como un reconocedor de patrones para obtener el desplazamiento y dirección de cada uno de ellos. Con estos resultados y el tiempo entre las imágenes, obtuvieron un campo de velocidades de desplazamiento de las nubes. Al comparar sus resultados con mediciones *in situ*, encontraron una correspondencia del 82% en la velocidad y del 72% en la dirección. Aunque el método requiere una gran capacidad de cómputo, empleando la transformada inversa de Fourier, puede reducirse este considerablemente. El mérito de este estudio, corresponde a que fue uno de los primeros en intentar esta metodología aplicada a una imagen obtenida por un sensor remoto. Aunque sus resultados no tienen una exactitud que sea muy confiable.

Por su parte, Emery *et.al.* (1992), emplearon secuencias de imágenes infrarrojas de temperatura superficial del mar (SST) de un sensor AVHRR. Ellos también ocuparon el método de

correlación cruzada máxima (MCC). Las imágenes, filtradas para cobertura nubosa, se utilizaron para encontrar los desplazamientos de patrones de temperatura superficial mediante la búsqueda de la máxima correlación cruzada en ventanas de cada par de imágenes. Encontrando una deficiencia en el método cuando hay presencia de fuertes corrientes superficiales, existen gradientes débiles de temperatura y, actúan giros ciclónicos o anticiclónicos (eddys) en el lugar, aunque proponen un método alternativo para solucionar este último problema. Además de que en principio se asume que el movimiento de los patrones termales obedece únicamente a procesos advectivos locales. Desechando la presencia de procesos geostróficos y residuales por ser de gran escala comparados con la escala de tiempo que se emplea en este tipo de estudio (aproximadamente 1 día), además del efecto de intercambio de calor entre la tierra y la atmósfera por considerarse mínimo en la misma escala de tiempo mencionada anteriormente.

En el año de 1992, Holland y Xiao-Hai elaboran un estudio para el reconocimiento de patrones termales, su discriminación y seguimiento en imágenes SST de la banda de 10 μm. Estos autores emplearon un método distinto al descrito anteriormente, el cual consiste en un algoritmo estadístico ordenado de detección de bordes que se emplea para seleccionar patrones termales mediante la detección de mapas de gradientes (OSEDA), y al mismo tiempo discriminar entre la superficie del agua, la parte correspondiente a la tierra[2] y las nubes. Mencionan que al emplear el gradiente como método de selección de patrones, se hace innecesaria la aplicación de filtros y máscaras para la remoción de nubes y tierra. La desventaja de este método, consiste en que depende únicamente de la magnitud de los gradientes de temperatura que se encuentren presentes en la imagen. Por lo que se ve limitado cuando los valores de temperatura son muy uniformes en el área estudiada, o bien, cuando la estructura de los patrones termales es muy compleja.

En Qing *et.al.* (1992), se empleó nuevamente el método de correlación cruzada máxima (MCC) para estimar las velocidades de desplazamiento de patrones termales empleando imágenes SST. La aportación de este estudio consistió en una metodología para identificar las áreas no aptas para la estimación del MCC, además de un método más riguroso para examinar la significancia de la correlación cruzada a tres niveles. Encontraron que hay dos factores importantes que limitan la efectividad del método MMC y son: la resolución de la calibración de la temperatura y, la resolución espacial de la imagen. Las fallas del método ocurren en regiones en donde la variación de la temperatura es muy pequeña; en áreas en donde existe movimiento rotacional ya que le método MCC únicamente busca dependencias lineales, cuando los patrones termales son muy grandes o muy pequeños para la resolución espacial de la imagen.

En 1994, Simpson y Gobat retomaron el método MCC para estudiar los errores de este método en relación a inexactitudes referidas al aumento en la magnitud y dirección en la derivación de la estimación de velocidades superficiales. Proponen un procedimiento numérico interactivo que combina métodos de análisis de imágenes y restricciones dinámicas para minimizar estas dificultades. Encontrando buena correlación entre los resultados estimados y las mediciones *in situ*, por lo que emplearon un modelo para simular el seguimiento de boyas de deriva para comprobar sus resultados, encontrando que el modelo se comporta de acuerdo a las mediciones reales y a los resultados.

[2]El termino "tierra" se refiere en este trabajo a la porción continental del planeta, así como a las islas, islotes o cualquier otra forma geológica que se encuentre sobre la superficie del mar.

Otro método para estimar las velocidades superficiales de patrones termales, fue desarrollado por Nan-Jung y Xiao-Hai en 1994. Ellos emplearon la similitud de formas aplicados para seguir patrones de bordes automáticamente en un par de imágenes SST. Para clasificar a las formas en el par de imágenes, emplearon el promedio pesado del radio y el centroide (MSSM). Antes procesaron las imágenes con un filtro de mediana de 3x3 para eliminar ruido y se aplicó un filtro de gradiente para encontrar los bordes de los patrones. Las componentes de la velocidad para ciertos puntos se calcularon mediante un procedimiento de prueba-error y descomposición de vectores. La desventaja de este método consiste en que las imágenes deben presentar gradientes fuertes para poder ser detectados por los filtros, además de que únicamente se puede aplicar en algunos puntos seleccionados de acuerdo al gradiente. Por el contrario, la gran ventaja del método es que es sensible a los movimientos rotacionales -el método MCC no lo es- por lo que se puede aplicar a áreas en donde existan este tipo de estructuras marinas (como por ejemplo en el Golfo de México, o durante el paso de un huracán). Por otra parte, aunque no lo mencionan en su artículo, el método de gradientes empleado no es sensible a la presencia de tierra o nubes -en donde el gradiente es máximo-, por lo que la imagen requiere de una evaluación subjetiva de la selección del borde a emplear, o bien, de un mayor preproceso para enmascarar la ocurrencia de estas estructuras.

Hasta el momento, se han analizado algunos métodos para el cálculo de velocidades de imágenes SST, que aunque son distintos, todos tienen algo en común: requieren una capacidad de cómputo muy grande, lo cual los hace muy difícil de implementar en sistemas automatizados en tiempo real. Esto se debe, a que los métodos -a excepción del MSSM-, requieren que se evalúe cada pixel de la imagen, así como sus vecinos más cercanos en una ventana de aproximadamente 30x30. Lo cual redunda en un mayor tiempo de cómputo, por lo que surge la siguiente pregunta: ¿existirá alguna manera de reducir el número de cálculos necesarios para la estimación del campo de velocidades?, o bien, ¿como eficientizar la manera de realizar todos estos cálculos?.

En este sentido, Kranspolsky y Breaker (1995) implementaron un método para corregir los errores en la localización terrestre. El método consiste en realizar la corrección geográfica basada en puntos de control terrestres y procedimientos de optimización de mínimos cuadrados, produciendo una función de corrección que se aplica únicamente a los vectores de desplazamiento como tales, y no al campo entero de pixeles contenidos en la imagen, como usualmente se ha realizado.

3 METODOLOGÍA

3.1 Para el cálculo del campo de velocidades superficiales

3.1.1 Obtención de las imágenes

Las imágenes utilizadas en el presente proyecto provienen del satélite artificial AVHRR-II, el cual obtiene imágenes dentro del espectro electromagnético que comprende del rojo hasta el infrarrojo térmico además del espectro visible.

Estas imágenes son capturadas automáticamente por una estación receptora situada en el Instituto de Geografía de la UNAM y almacenadas temporalmente en una estación de trabajo SUN Spark pasando posteriormente a cinta para su almacenamiento final.

El Instituto cuenta con un sistema de manipulación de estas imágenes llamado Terascan, el cual lee las imágenes en su formato original para ser presentadas en la pantalla de la terminal para su procesamiento. Aquí se le aplican filtros de realce de contrastes y aplicación de falso color (González y Woods, 1993; KHOROS, 1994; Rasure y Lotufo, 1994) con el propósito de ver claramente el contenido de las imágenes en cualquier rango del espectro electromagnético que captura el sensor del satélite.

Mediante el paquete Terascan y empleando las bandas 3 y 4 del infrarrojo cercano de las imágenes AVHRR seleccionadas, se obtuvieron los mapas de temperatura superficial del agua de mar. Se recortó la región de interés la cual comprende a la región oceánica frente a las costas de Lázaro Cárdenas, Michoacán. Se le aplicó un realce interactivo a las imágenes hasta se mostrara claramente y se le aplicó un esquema de falso color con el propósito de aumentar el contraste entre las áreas de temperatura. Paso seguido se georreferenció la imagen con la ubicación geográfica comprendida por la imagen. Posteriormente se calibraron los datos de irirradianza y se transformaron a temperaturas superficiales, con la cual se obtienen valores de temperaturas en grados centígrados o absolutos con una precisión de hasta medio grado. Se geocorrigió para eliminar el efecto de distorsión producido por la curvatura de la Tierra (Mitasova y Hofierka, 1994; TERASCAN, 1995).

La información de estas bandas permiten alcanzar una precisión de hasta 1 Km. por 1 Km. en el cenit de las mismas. Por lo que la cantidad de información contenida en las imágenes hacen que los archivos tengan un tamaño de hasta 70 megabytes completas. Para calcular el mapa de velocidades superficiales se necesita manipular el valor de temperatura de cada posición dentro de la imagen por lo que el hacerlo directamente de los datos de la imagen original hubiera implicado una gran capacidad de cómputo dadas las dimensiones del área de estudio. No obstante, los patrones de variación que se deseaban observar corresponden a un orden de magnitud tal que ese detalle es innecesario ya que se habla de eventos oceanográficos a nivel de mesoescala.

Por ello se opto por una opción que permitiera capturar la distribución de colores de la imagen ya desplegada en la pantalla y simplificar así el proceso de obtención de los colores asociados a temperaturas superficiales. Dado que los sistemas computacionales requieren de promediar los valores binarios comprendidos en cada byte de color asociado a cada pixel, la información presentada en pantalla se encuentra ya simplificada y al capturar dicha información directamente de la pantalla, se puede seleccionar la profundidad del mapa de colores que se desee guardar en un archivo gráfico. Así entonces, se seleccionó guardar los mapas resultantes en formato GIF con un tamaño estándar de aproximadamente 200 pixeles por 200 pixeles debido a

que con este tamaño la imagen es clara sin que haya necesidad de escalar la imagen y distorsionarla.

Se optó por capturar la imagen de la pantalla ya que dadas las características del método, no se consideró necesario trabajar directamente con los datos numéricos pues lo que se deseaba era encontrar la variación espacial de los mapas de colores en una escala regional por lo que el detalle fino no era necesario, puesto que si se tiene una imagen con una cantidad de pixeles adecuada se pueden realizar los cálculos de una manera confiable en términos de una mesoescala.

La estación gráfica de captura y manipulación de las imágenes de satélite cuenta con despliegue de 24 bits de profundidad, logrando desplegar hasta aproximadamente 17 millones de colores. Por lo cual, la información de la paleta "real" de colores del mapa queda registrada dentro de la imagen. Por su parte el formato GIF tiene la característica de contener exclusivamente un mapa de colores máximo de 256 colores (un byte) en lugar de los 17 millones de colores (3 bytes) que puede desplegar la estación gráfica. De esta manera se pueden manipular fácilmente los tres colores primarios del sistema RGB y mejorar el rendimiento de ejecución.

Debido a que el algoritmo de análisis de la imagen se desarrolló en el lenguaje de programación Java y que éste lee la información de la imagen directamente del archivo y lo manipula a nivel pixel, se logró mantener la integridad de la información que se capturó al momento de crear la imagen, de tal manera que en la computadora del usuario final se presente esta información sin modificarse considerablemente, sin importar su tipo y capacidad de despliegue ni la plataforma de la que se trate, por lo que los resultados del análisis de las mismas imágenes siempre serán los mismos para todos los usuarios que las utilicen.

Habrá que recordar que el formato GIF utiliza un algoritmo de compactación de datos del tipo LZW - por lo que no existe pérdida de los mismos a menos que se modifique la paleta de colores- evitando la degradación de la imagen además de que se obtiene un tamaño menor de archivo, por lo que se mejora la transmisión de la misma comparado al que tiene la imagen al ser desplegada por cualquier visualizador, logrando con ello un mejor aprovechamiento del ancho de banda de la red y por consiguiente, una mayor velocidad de transferencia hacia la computadora del usuario.

Mediante este sistema, se seleccionaron las imágenes a estudiar de acuerdo a las siguientes características:

- Que la cobertura regional comprendiera un área de estudio específica, en este caso, la región oceánica frente a las costas de Jalisco, Colima y Michoacán.
- Que la cobertura nubosa fuera nula en un área dentro de la región de estudio.
- Que en dicha área se contara con una imagen clara perteneciente al infrarrojo térmico (canal 4 y 5).
- Que de acuerdo a los puntos anteriores, esta área estuviera localizada cerca del centro de la imagen para evitar distorsión por la curvatura de la tierra.
- El último requerimiento, y quizás uno de los más importantes, es que se contara con dos imágenes de la misma zona con una diferencia de tiempo de muestreo no mayor a 12 horas (Emery *et.al.*, 1992).

Una vez que se obtuvieron al menos dos imágenes que cumplieran las características antes mencionadas, se capturó directamente de la pantalla el área seleccionada y se guardó con formato GIF.

3.1.2 Metodología del cálculo de velocidades superficiales

Para el cálculo de velocidades superficiales en la parcela de agua, se seleccionó la metodología desarrollada por Wahl y Simpson (1990) la cual se basa en la correlación cruzada sin tomar en cuenta efectos rotacionales con el objeto de simplificar el proceso de cálculo. Para comenzar, se describe la metodología basada en la descripción matemática, en donde se muestra un arreglo bidimensional de pixeles que representa la matriz de correlación entre dos imágenes.

Considerando un par de funciones reales, discretas y bidimensionales *s(m,n) y p(m,n).* En donde la autocovarianza y la covarianza cruzada están definidas por las ecuaciones 1 y 2 respectivamente.

$$Cov_{ss} = E\left[\{s(m,n)-\eta_s\}\{s(m+m_0,n+n_0)-\eta_s\}\right] \quad (1)$$

Ecuación 1. Definición de la autocovarianza de la función s(m,n)

en donde **E[.]** es el valor esperado de la covarianza, ***(m_0,n_0)*** es retraso espacial entre las funciones y η_0 , η_p están dadas por:

$$\eta_s = E\left[s(m,n)\right] \quad (3)$$

Ecuación 2. Valor esperado de los datos del primer arreglo

y para el arreglo p(m,n)

$$Cov_{sp}(m_0,n_0) = E\left[\{s(m,n)-\eta_s\}\{p(m+m_0,n+n_0)-\eta_p\}\right] \quad (2)$$

Ecuación 3. Definición de la covarianza cruzada entre las funciones s(m,n) y p(m,n)

Como puede observarse, la autocovarianza entre las dos funciones discretas corresponde a la diferencia entre los valores de los datos esperados de la segunda función y la diferencia entre los valores correspondientes de la primera función comparada con la segunda. Definiendo así una medida de similitud entre ambas funciones comparadas entre si.

$$\eta_p = E\left[p(m,n)\right] \quad (4)$$

Ecuación 4. Valor esperado de los datos del segundo arreglo

A partir de lo anteriór el coeficiente de correlación está definido como:

$$r_{sp}(m_0, n_0) = \frac{Cov_{sp}(m_0, n_0)}{\sigma_s \sigma_p} \quad (5)$$

Ecuación 5. Definición del coeficiente de correlación cruzada

en donde el valor de correlación antes de su valor máximo se define como:

$$|r_{sp}(m_0, n_0)| \leq 1 \quad (6)$$

Ecuación 6. Valor de la correlación cruzada antes del máximo

y

$$r_{ss}(0,0) = 1 \quad (7)$$

Ecuación 7. Valor máximo del índice de correlación cruzada

Delimitando así el límite máximo que puede tomar el coeficiente de correlación, el cual ocurrirá cuando las dos funciones presenten su máxima similitud.

Entonces, si la segunda señal corresponde a una versión exacta con retraso espacial de la primera señal se tiene:

$$s(m,n) = p(m + m_0, n + n_0) \quad (8)$$

Ecuación 8. Retraso espacial entre la segunda señal y la primera

De acuerdo con todo lo anterior, la ecuación 2 y la 7 requieren que el coeficiente de correlación presente un máximo absoluto en este retraso espacial o en $r_{sp}(m_0, n_0)=1$. Para cualquier señal física, la correlación máxima tendrá un valor menor a 1 ya que la segunda señal no es, necesariamente, una versión desfasada de la primera. El valor de la correlación disminuirá

conforme las señales sean cada vez más disimiles. Entonces, el valor de la correlación es una **medida de la disimilitud de las dos señales con un cierto desfase.**

Si se consideran dos imágenes de satélite del tipo AVHRR geocorrelacionadas en la misma malla espacial estas imágenes pueden verse como un par de funciones discretas bidimiensionales en las cuales sus primeros índices se relacionan con las líneas de las imágenes (renglones), y cuyos segundos índices se relacionan con las muestras dentro de la imagen (columnas). El origen de cada imagen se encuentra en la esquina superior derecha, en donde el índice de las líneas se incrementa hacia abajo y el índice de la muestra se incrementa hacia la derecha.

Dada una subsección *s(m,n)* tomada de la segunda imagen, el problema se limita a determinar si ésta contiene una región similar a una subsección *p(i,j)* de la primer imagen. La subsección *p(i,j)* se denomina como el **mosaico patrón**, mientras que *s(m,n)* se le denomina como el **mosaico de búsqueda** (figura 1).

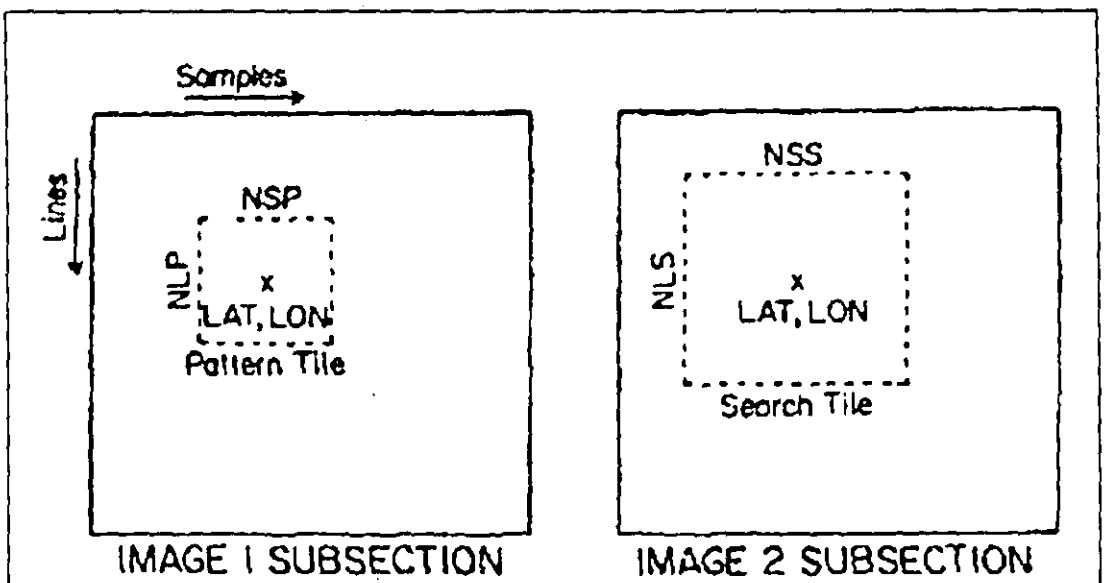

Figura 1. Relación geométrica entre el mosaico patrón y el mosaico de búsqueda, en donde el centro del mosaico patrón corresponde al centro del mosaico de búsqueda en la subsección de la imagen 2.

Así entonces, el mosaico patrón es una sección de la primer imagen que ocupa las mismas coordenadas espaciales en la región central del mosaico de búsqueda de la segunda imagen.

De esto se deriva que la matriz de correlación entre el mosaico patrón y el de búsqueda esta dado en función de la ubicación de cada pixel y su valor de color en ambos mosaicos como se indica en la siguiente ecuación:

$$r_{sp}(k,l) = \frac{\sum_i \sum_j [s(i+k, j+l) - \eta_s(k,l)][p(i,j) - \eta_p]}{D} \quad (9)$$

Ecuación 9. Relación entre el mosaico patrón y el mosaico de búsqueda

en donde el parámetro D se expresa como:

$$D = \left[\sum_i \sum_j [s(i+k, j+l) - \eta_s(k,l)]^2 \sum_i \sum_j [p(i,j) - \eta_p]^2 \right]^{1/2} \quad (10)$$

Ecuación 10. Definición del divisor del índice de correlación cruzada

En esta ecuación $\eta_s(k,l)$ es el valor promedio de *s(m,n)* en la subregión coincidente con *p(i,j)*, y las sumatorias se realizan en las coordenadas comunes para *s* y *p*. El valor de η_p se calcula una sola vez fuera de las sumatorias y está dada por la ecuación 4. El rango de valores de *k* y *l* corresponden a las regiones de correlación en donde *p(i,j)* esta completamente contenido en *s(m,n)* como se observa en la figura 2.

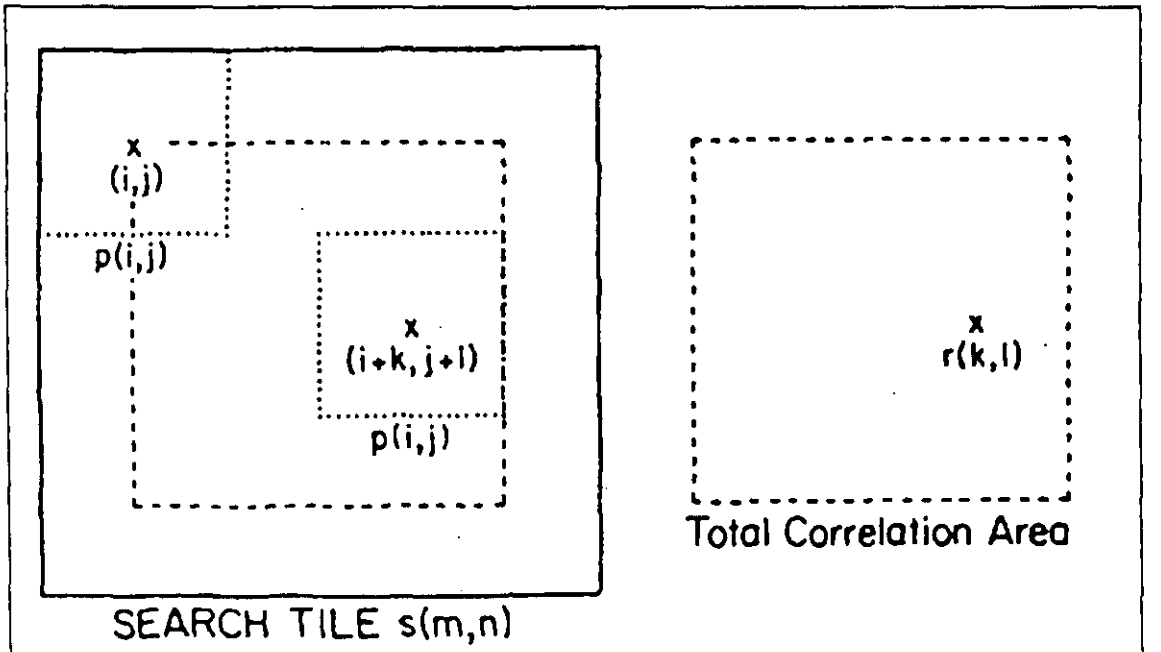

Figura 2. Delimitación del área total de correlación. El mosaico patrón de la primer imagen se recorre una cierta cantidad de pixeles en la segunda imagen, formando así el área total de correlación

De lo anterior se desprende que el método de correspondencia de patrones determina el desfasamiento espacial entre el mosaico patrón de la primer imagen y el mosaico de búsqueda de la segunda imagen al encontrar la localización de su correlación máxima. *Por lo que al obtener el desfasamiento espacial y el tiempo de muestreo entre las imágenes, se obtiene la velocidad promedio de las propiedades en el mosaico patrón.*

Explicado de otra manera, el procedimiento de búsqueda de patrones similares entre los dos mosaicos de las imágenes se da de la siguiente manera: el mosaico de búsqueda de la segunda imagen se sitúa en la primer imagen de tal manera que el mosaico *no rebase* la extensión de la misma, quedando acotado a los márgenes de la primer imagen. En este paso se calcula el valor de la correlación entre los pixeles comprendidos en el mosaico de búsqueda y los pixeles comprendidos en el mosaico patrón correspondiente. Una vez que se obtiene ese valor, el mosaico se corre una cantidad determinada de pixeles sobre el eje X y se vuelve a repetir el proceso. Como parámetro de restricción, se considera que el valor de correlación cruzada corresponde a un cambio significativo entre las dos imágenes, siempre y cuando se rebase un valor de umbral. El cual puede ser recorrido para la experimentación, con el consabido cambio en los resultados definiéndose así la sensibilidad de resolución del método. Es importante recalcar que la definición y selección de estos valores tiene una injerencia directa en la cantidad de cálculos que se van a llevar a cabo durante todo el proceso de búsqueda.

3.2 Para el análisis y diseño del programa empleando la orientación a objetos

3.2.1 La metodología orientada a objetos empleada en el análisis y diseño del programa

Una vez revisado y estructurado el problema matemático y operacional que implicó el cálculo de las velocidades superficiales, se prosiguió con el análisis y diseño orientado a objetos del programa para su posterior codificación en un lenguaje de programación.

Para ello se utilizó la metodología propuesta por Booch(1994) para el análisis y diseño orientado a objetos. En donde inicialmente debe definirse el problema y sus requerimientos de una manera clara y objetiva con el propósito de localizar los candidatos a clases, objetos y métodos. A partir de este análisis inicial se crean diccionarios de datos y métodos y se seleccionaron aquellos elementos que corresponden a una jerarquía mayor pudiendo localizar y diferenciar entre clases, objetos, métodos e instancias de clase.

Paso seguido se define la interacción que puede ocurrir entre ellos delimitando escenarios y casos de uso así como los diagramas de secuencia que muestre los métodos invocados para la ejecución de las diferentes tareas del sistema.

Posteriormente, se comienzan a definir las clases principales del sistema y a descomponer el problema en sus elementos principales. De acuerdo a la jerarquía resultante definida por las clases principales, se evaluó la necesidad de crear relaciones de herencia que fueran definiendo las características específicas que van a mostrar cada uno de los objetos en el sistema.

Todos estos pasos tienen que ser revalidados para ir depurando los esquemas iniciales y encontrar así un diseño final. Hasta que se logran tener finalmente los diagramas de clases y objetos que van a ser utilizadas en la codificación.

Para elaborar el análisis y crear todos los diagramas se utilizó la herramienta de tipo CASE Rational Rose ya que permite tener un mayor control de toda la información sin tener que tener siempre en mente toda la estructura del sistema, pues cada elemento del tipo clase u objeto se observa como un actor en un escenario y no como datos y métodos aislados.

3.2.2 El lenguaje de programación y sus requerimientos

Fueron varios los elementos que se tomaron en consideración como directivas primordiales que debía presentar el programa ya finalizado y son:

1. Escrita en un lenguaje orientado a objetos de fácil comprensión y corto tiempo de desarrollo.

2. Posibilidad de que el sistema sea utilizado prácticamente en cualquier plataforma con una configuración de hardware modesta, acercándose a las características de equipo que puede encontrarse normalmente en una institución educativa y empresas medianas.

3. Que el tiempo de ejecución sea relativamente aceptable y óptimo para no saturar las capacidades de un equipo servidor, de tal manera que pueda aumentar la cantidad de usuarios concurrentes que utilicen el sistema.

4. Que pudiera ser utilizado preferentemente en un ambiente de redes del tipo Internet.

Inicialmente la implementación del sistema se realizaría en C++, sin embargo las bibliotecas de clases para interactuar directamente con la estructura binaria de los archivos gráficos son muy complejas. El acceso a la información binaria se realiza mediante funciones que extraen una cierta cantidad de bits directamente del archivo para que posteriormente sean ordenadas por el programador en los correspondientes bytes. Requiriendo que el programador conozca a detalle la estructura del formato del archivo gráfico y pueda manejar variaciones entre sus diferentes versiones. Aún y cuando se resuelva este problema, al compilar el programa generaría un ejecutable en la plataforma en donde se trabaje por lo que sería necesario compilar nuevamente para cada plataforma para la cual se desee distribuirlo. Con ello el programador debe generar modificaciones al código fuente para asegurarse que el programa se ejecuta sin dificultades en cada plataforma. El siguiente problema ocurre cuando se involucró al ambiente gráfico pues habría que compilar una versión para Windows y sus derivados, y otra completamente distinta para Unix con un ambiente X.

En vista de estas problemáticas se resolvió emplear el lenguaje de programación Java para codificar el diseño. Se realizó esta elección en virtud de que las características propias del lenguaje llenan en gran medida los requerimientos descritos anteriormente: es completamente orientado a objetos; el código intermedio generado por el compilador puede ser ejecutado en cualquier plataforma sin la necesidad de equipo muy sofisticado y potente; el programa puede ser ejecutado total o parcialmente en el equipo del usuario; y especialmente en su potencial de distribución hacia el usuario final mediante Internet. Esta última característica representó una inquietud permanentemente presente durante toda la elaboración del proyecto.

Otra particularidad importante de este lenguaje de programación es su inherente funcionalidad en cualquier ambiente gráfico con un navegador con soporte para Java, por lo que no fue necesario codificar para plataformas específicas, sino heredar características gráficas de las clases correspondientes incluyendo la capacidad de leer los valores de cada pixel directamente de los archivos gráficos soportados.

RESULTADOS

Las pruebas a las que se expuso el programa del análisis de las imágenes satelitales de temperatura superficial del agua de mar, indican un gran potencial del lenguaje de programación Java como una alternativa viable y de rápido desarrollo para aplicaciones de investigación científica.

La naturaleza intrínseca del lenguaje completamente orientada a objetos, facilita el análisis y diseño de los algoritmos responsables del cálculo de los diferentes parámetros a medir en cualquier proyecto de investigación.

Aunque este lenguaje todavía se encuentra en una etapa de desarrollo continuo, ha probado ser consistente y confiable durante la ejecución en las diferentes pruebas a la que fue sometido. Las nuevas versiones que se anuncian de este lenguaje, prometen una cantidad de mejoras y aumentos, que harán del mismo una opción accesible a la comunidad científica, académica y de desarrollo a nivel mundial.

Por otra parte, dada la tendencia globalizadora de la compartición de la información, es de suma importancia contar con herramientas poderosas y flexibles que habiliten, de una manera rápida y económica, que los resultados de cualquier investigación puedan ser obtenidos por el usuario final sin importar su ubicación geográfica o física. En este contexto, Java introdujo un nuevo concepto de accesibilidad de la información sin contar además con los nuevos niveles de interactividad que se logran establecer entre el usuario, la información y los programas servidores de la misma.

Otra característica importante a denotar del lenguaje, es que, gracias a su enorme difusión mundial y la aceptación por parte de las empresas que fabrican los navegadores de páginas electrónicas pueden ejecutarse los programas hechos en Java prácticamente en cualquier plataforma. Con lo cual no se requiere, por parte del usuario final, adquirir y dedicar esfuerzos o recursos a la adquisición de computadoras caras dedicadas exclusivamente a la ejecución de algunos programas en específico: lo que ocurre comúnmente en cualquier institución, privada o pública, que desee utilizar paquetería especializada en su actividad.

3.2.3 Análisis y diseño

El análisis y diseño del sistema se realizó mediante el método de Booch (1994), empleando las herramientas de creación de diccionario de datos, objetos y clases; manipulación de escenarios de relación entre objetos y clases; diagramas de clase y objetos (Scott, 1998) que permitieron manejar la abstracción del problema.

A continuación, se muestran los resultados de las diferentes etapas del método y sus resultados, los cuales se utilizaron en todo el planteamiento del problema y que finalmente condujo a la implementación de la aplicación.

3.2.3.1 Los primeros pasos: definición del problema

La primer aproximación se realizó mediante una análisis del problema que se necesitaba solucionar y la delimitación de los diferentes componentes e interacciones entre los objetos y/o clases, o los candidatos a objetos y clases que se fueron presentando en esta etapa inicial.

El entendimiento de la funcionalidad del sistema se comenzó a definir de la siguiente manera:

"Se cuenta con imágenes de satélite de la temperatura superficial del agua de mar (SST) que están conformadas por mapas de colores que representan los valores de irirradianza de los primeros 10 centímetros de la superficie del mar. Estas imágenes pueden presentar diversas características cuando se reciben directamente del satélite: bajo nivel de contraste en la totalidad de la imagen, distorsión de la misma por el efecto de la curvatura de la tierra, cobertura nubosa que impide la obtención de los valores de irirradianza del agua. Debido a esto, las imágenes necesitan ser procesadas digitalmente para aumentar los valores de contraste entre las áreas coloreadas que representan las diferentes temperaturas, aplicar falso color para diferenciar claramente entre las temperaturas cálidas y templadas, geocorrelacionar la imagen para eliminar la distorsión, aplicar modelos a los valores de irirradianza para convertirlos a grados Kelvin y posteriormente a grados centígrados y por último guardar la imagen resultante en un formato que sea manipulable por el sistema. Este mismo procedimiento se aplica a otra imagen con la misma cobertura espacial pero con diferente cobertura temporal. Posteriormente, el par de imágenes deben de ser puestas a disposición del usuario mediante un servidor web para que un programa pueda transferirla a la computadora del usuario, sea desplegada y en tiempo real, sea procesada digital y numéricamente mediante los valores de colores de cada pixel para calcular la diferencia espacial entre los mapas de colores y con estas diferencias se calcule la velocidad de desplazamiento y dirección de dichos cambios. Por último, el sistema debe mostrar numéricamente o gráficamente dichos cambios para que el usuario pueda imprimirlo u obtener esta información directamente de la pantalla de su computadora."

Las palabras subrayadas, indican los candidatos a convertirse en clases u objetos dentro del sistema. Basado en esto, se crearon diccionarios de clases y objetos que tuvieron que ser revisados nuevamente para establecer si los candidatos se convertían ya en parte del diseño del sistema, o bien, pasaban a ser objetos o incluso métodos de clase o métodos de objetos.

Una vez que se definió lo anterior, se procedió a la codificación diagramática para delimitar gráficamente las estructuras de las clases y la relación entre ellas, tal y como se muestran en los diagramas de clase del sistema.

Dado que el problema que se estudia aquí involucra una capacidad y cantidad de cómputo considerable, se pretendió que el diseño final fuera lo más simple y claro posible. De tal manera que la petición de recursos y rendimiento de ejecución del programa, se concentrara en aquellos procedimientos de cálculo matemático, en lugar de los procedimientos de despliegue y presentación. Por lo que se simplificó mediante métodos ya implementados en Java, sacrificando un poco la ortodoxia del diseño en aras de una mejor ejecución durante la codificación.

Además se cuidó un aspecto más: la reusabilidad de código que ofrece la tecnología orientada a objetos. De tal manera que se opto por una clase dedicada exclusivamente a definir las propiedades de las imágenes y que contuviera los métodos y objetos que permiten el procesamiento digital de las mismas. Separando todo esto del cuerpo principal del programa.

Con lo anterior se logró de una manera muy sencilla, que la implementación y definición de nuevos métodos de procesamiento puedan agregarse en esta clase y aprovechen las características ya definidas para los objetos de esta clase. Por lo que el mantenimiento y crecimiento del sistema se vea reflejado en cambios exclusivos de la clase Procesa_imagen, sin que esto afecte al resto del código. Por lo que es posible, incluso, cambiar el método principal de cálculo del desplazamiento de las áreas de color, sin mayor esfuerzo que el de su codificación dentro de la clase. Mientras que los

métodos y objetos relacionados con la parte principal del sistema y la presentación de los resultados no se ven afectados en absoluto, siempre y cuando, se mantenga un mismo esquema de relación entre las clases involucradas.

3.2.4 Análisis y diseño del problema

El análisis y diseño del problema se realizó siguiendo el método propuesto por Booch (1994), en el cual se crearon diagramas de actores, escenarios y clases que permitieron delimitar las clases y objetos que estaban involucrados en el problema, así como visualizar sus relaciones y dependencias.

Esta metodología propone que el análisis del problema se fundamente en estrategias claras y redundantes para obtener los diagramas de relaciones que son la base para la codificación subsiguiente.

3.2.4.1 Análisis y diseño de la aplicación

Mediante el método de Booch (1994) y la ayuda del programa Rational Rose, se crearon los diagramas de actores, escenarios, estado y clases que fueron empleadas para la posterior codificación del sistema en el lenguaje Java. Con estos diagramas, se muestra y ejemplifica de una manera sumamente sencilla las estructuras de las clases y, sobre todo, las dependencias que se mantienen entre las diferentes clases.

Resultan especialmente útiles para comprender en su totalidad la abstracción del problema y su granularización, permitiendo que el programador enfoque su esfuerzo en crear un componente a la vez. El cual, una vez terminado, se mantiene prácticamente intacto durante la codificación de los restantes elementos del sistema. Una ventaja clara de la orientación a objetos.

3.2.4.1.1 Diagrama de uso: principal

En este diagrama se presenta el flujo de utilización del sistema. En él se definen cuatro elementos principales:

1. Captura de imágenes.
2. El arreglo digital de la información de colores contenida en ellas.
3. El cálculo de velocidades superficiales mediante la búsqueda de las disimilitudes entre las imágenes.
4. La presentación gráfica de los resultados

La captura de la imagen se realiza mediante equipo externo y con la intervención de personal calificado, es por esta razón que no se incluyó en el diseño final del sistema, sin embargo se presenta en este diagrama para una mayor claridad de los pasos a seguir.

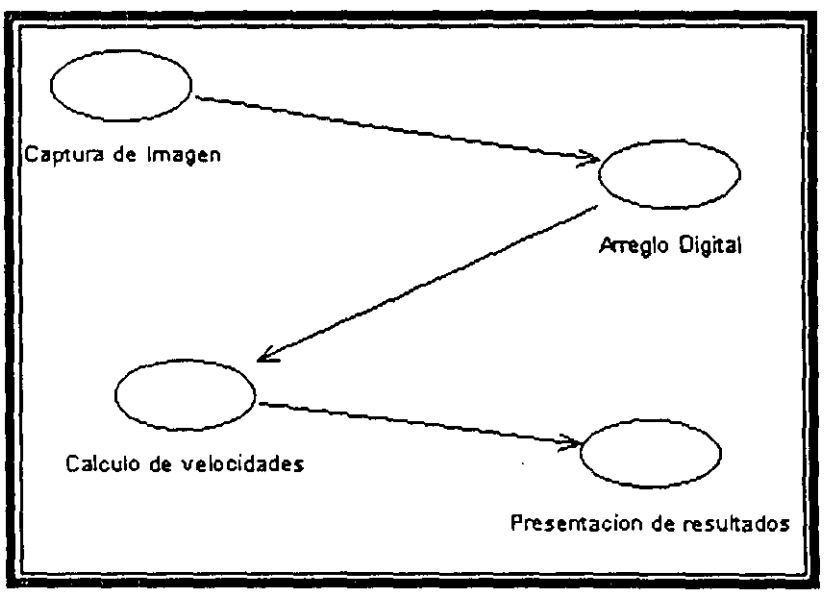

Figura 3. Diagrama de uso principal que muestra los componentes principales del programa.

3.2.4.1.2 Diagrama de uso: Captura de la imagen

En este diagrama se muestra los pasos a seguir para capturar la Imagen final a partir de los datos enviados a la estación receptora por el satélite AVHRR en turno. Como se puede observar, existe todo un procedimiento (definido como la elipse) que tiene que realizar la selección de la misma mediante técnicas visuales, ya que no se cuenta con un procedimiento automático para eliminar aquellas imágenes que presenten una cobertura nubosa tal, que impida ser utilizada para su estudio. Además se deben seleccionar aquellas imágenes que cubran el área de estudio.

Cada elemento funcional en el sistema se representa como un actor con características propias. En este nivel no es necesario conocer los detalles de la implementación de cada uno de ellos, sino su relación con el resto de los actores para llegar a un resultado final.

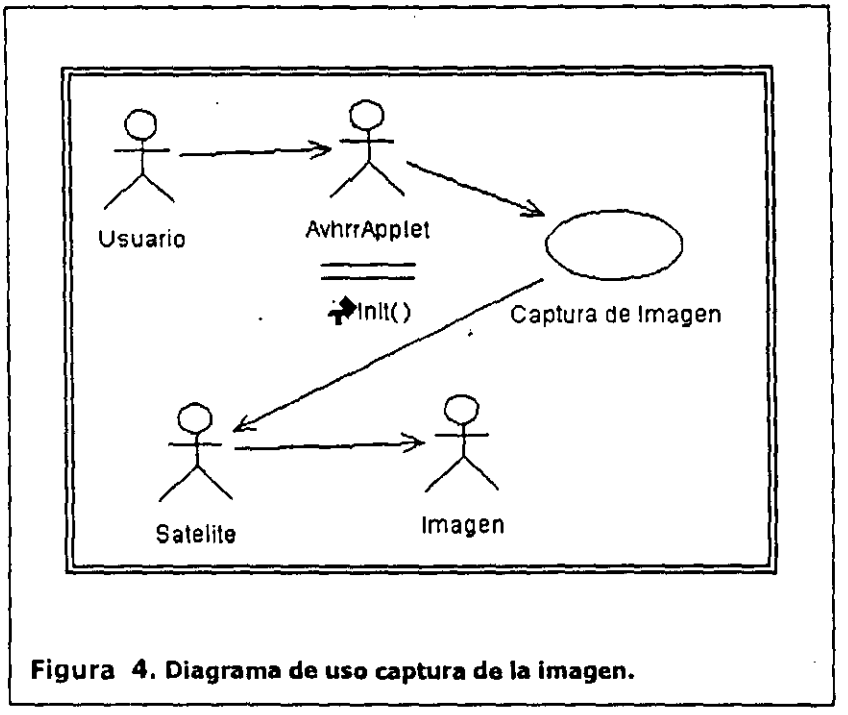

Figura 4. Diagrama de uso captura de la imagen.

3.2.4.1.3 Diagrama de uso: Arreglo digital

Aquí se observa el flujo a seguir para el cálculo de las velocidades superficiales. En donde el applet AVHRR_applet se inicializa a petición del usuario, obtiene las imágenes a estudiar y se manipulan digitalmente para el cálculo de las velocidades superficiales.

La manipulación digital comprende la captura de los valores numéricos de colores de cada pixel de las imágenes y se almacenan en un arreglo para su posterior utilización en el cálculo matemático.

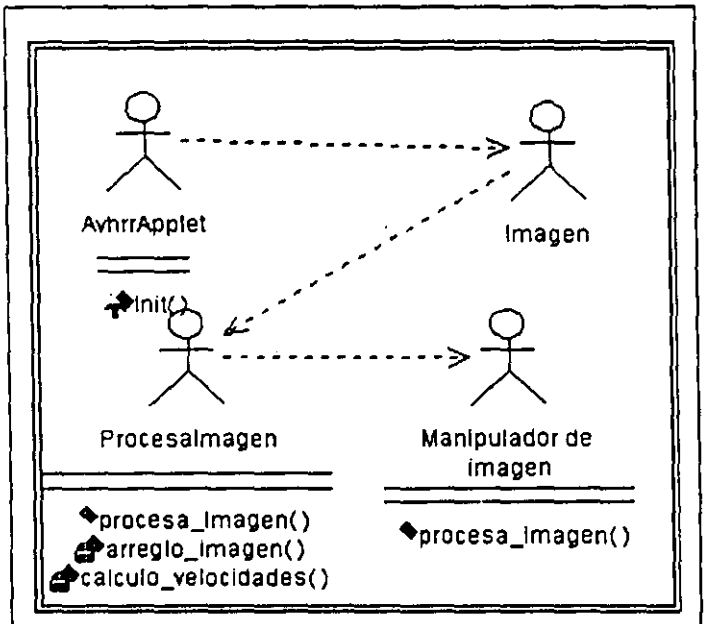

Figura 5. Diagrama de uso Arreglo digital de las imágenes.

3.2.4.1.4 Diagrama de uso: Cálculo de velocidades

En este diagrama se esquematiza el procedimiento para el cálculo de velocidades superficiales a partir de las imágenes que emplea el applet. La calculadora de velocidades se refiere, en este caso, a la clase Procesa_imagen quien es la responsable de llevar a cabo todos los cálculos necesarios para ello.

Es importante mencionar que el manipulador de la imagen y la calculadora de velocidades de alguna manera deben visualizarse como objetos o clases independientes para que cualquier modificación posterior al método del cálculo de velocidades pueda ser efectuado sin modificar significativamente la estructura y codificación del resto de los elementos.

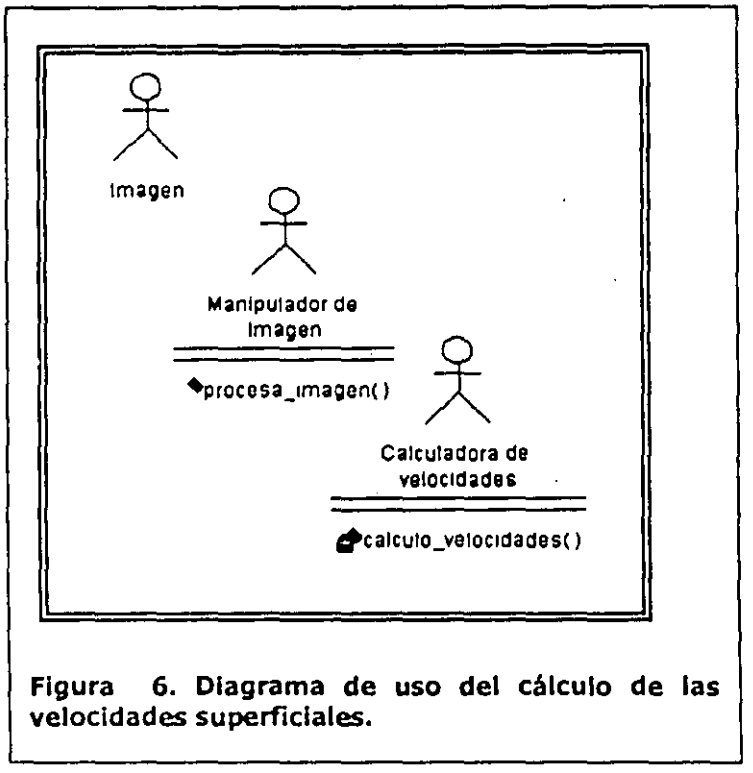

Figura 6. Diagrama de uso del cálculo de las velocidades superficiales.

3.2.4.1.5 Diagrama de uso: Presentación de resultados

Una vez que se realizaron los cálculos para las velocidades superficiales, estos tuvieron que ser presentados de alguna manera al usuario. En este caso, se emplea un graficador que muestra visualmente dichos resultados para una mejor comprensión de los mismos. Éste recibió los resultados de los cálculos directamente de la calculadora de velocidades y a la vez debe de interactuar con la clase Applet y las clases de gráficos de Java.

Si por alguna razón los métodos de presentación gráfica de los resultados son modificados, no será necesario modificar la calculadora de velocidades, manteniendo así la estructura de objetos y su encapsulación.

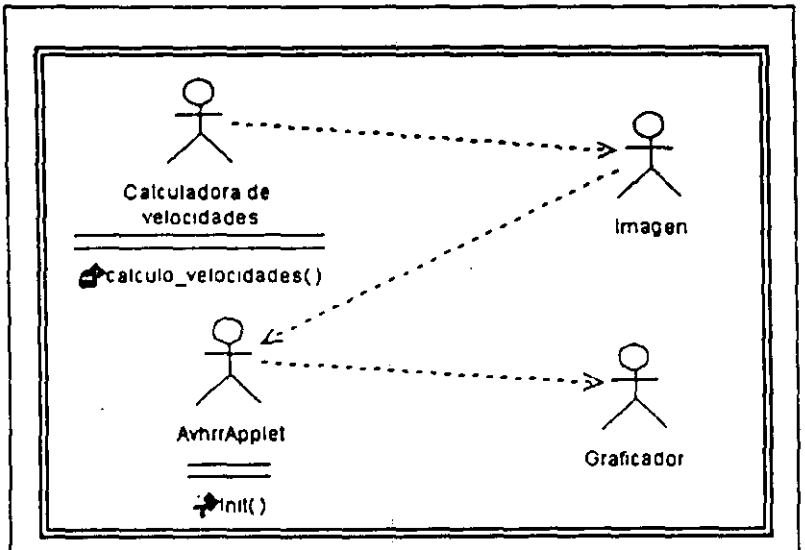

Figura 7. Diagrama de uso Presentación de resultados

3.2.4.1.6 Diagrama de secuencia: Selección de fecha y hora

Como un elemento adicional a la funcionalidad del sistema, se propone que el usuario seleccione una fecha y hora en particular para obtener las velocidades superficiales con esas referencias, siempre y cuando se cuente con las imágenes correspondientes en el servidor. La interfase del sistema se realizó a través de formas incluidas en páginas HTML.

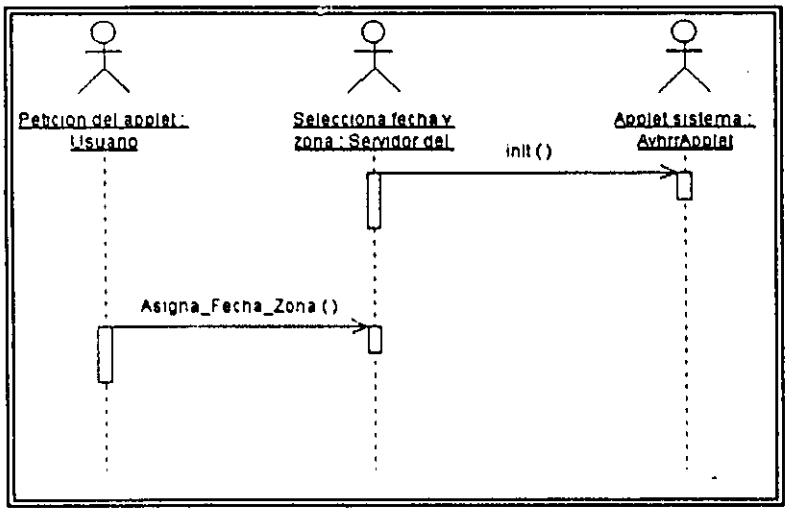

Figura 8. Diagrama de secuencia selección de fecha y hora.

3.2.4.1.7 Diagrama de secuencia: Captura y proceso

En este diagrama se muestran en conjunto todos los elementos involucrados en el sistema. Partiendo desde los elementos de captura inicial de los datos del satélite, pasando por el applet que toma las imágenes resultantes del primer elemento y las pone a disposición de la clase que realiza todo el cálculo. Hasta llegar a la presentación de los resultados al usuario final.

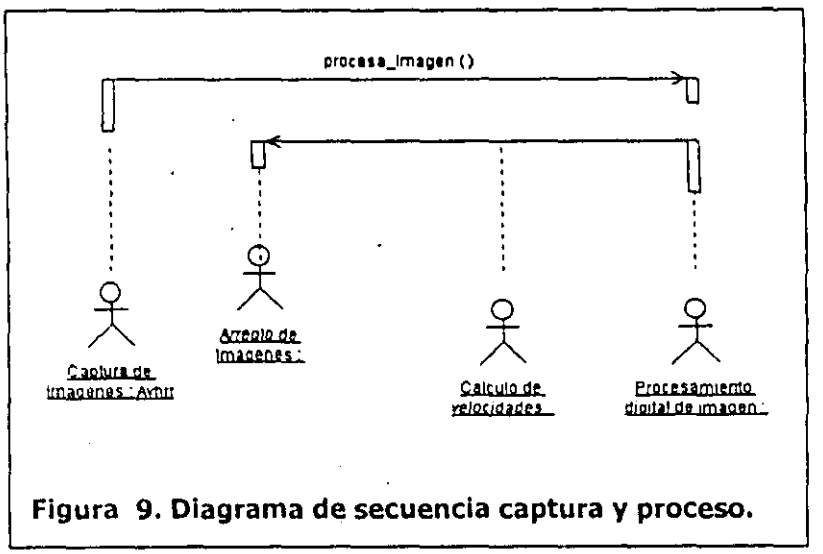

Figura 9. Diagrama de secuencia captura y proceso.

3.2.4.1.8 Diagrama de clases inicial

Después de analizar los diagramas anteriores y la funcionalidad que debería de presentar cada uno de los actores del sistema, se creó un diagrama de clases del sistema en donde se muestran 5 clases principales que van a interactuar con la clase AVHRR_applet, entre las que se cuentan: la clase procesa_imagen la cual se encarga del cálculo de las velocidades superficiales y la manipulación digital de las imágenes; la clase graficador que se encarga de la presentación gráfica de los resultados; la clase secuenciador quien mantiene el manejo de los hilos de la animación gráfica; y la clase DebugWin que se emplea para la depuración del sistema.

Como puede observarse, la clase AVHRR_applet es heredada directamente de la clase java.Applet la cual mantiene una relación de uso con el resto de las clases. En este caso no se consideró necesario aumentar aún más la jerarquía de clases en la clase procesa_imagen pues el método matemático es muy simple y se insertó como un método más de esta clase. En el caso en el que el método matemático para el cálculo de velocidades sea más sofisticado, será deseable aumentar la jerarquía y crear una subclase de procesa_imagen que se especialice en dicho procesamiento.

La figura 10 muestra este diagrama de clases. Aunque estas clases no fueron las que finalmente se codificaron en el programa, se muestra para dar una visión general y clara de los componentes del sistema y sus posibles mejoras.

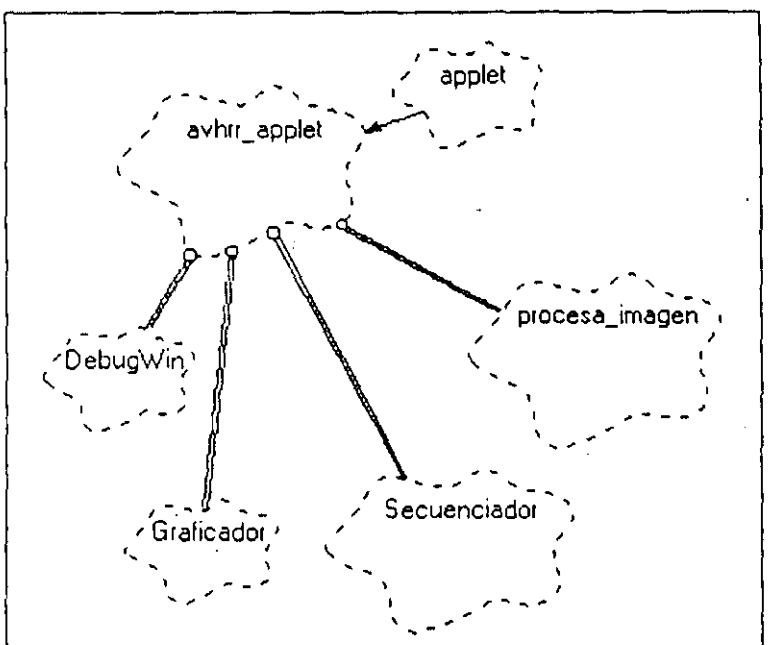

Figura 10. Diagrama de clases del sistema compuesta por 5 elementos principales.

3.2.4.1.9 Diagrama de clases final

Aunque durante el análisis se llegó a la necesidad de crear 5 clases para el sistema, la realidad es que hubo problemas que resolver durante la codificación que hicieron que este diseño fuera muy difícil de implementar.

Entre estos problemas se encontró la codificación del manejo de los hilos de control de la animación gráfica durante la presentaicón de los resultados, la cual resulta muy difícil de manejar si se incluye en una clase independiente de la clase de control principal. Por ello se optó por crear la clase principal implementando el tipo Runnable que implica un manejo intrínseco de multihilos.

Otro problema a resolver lo constituyó la presentación gráfica de los resultados, específicamente en lo que se refiere a mantener la sincronización entre el cálculo de los resultados y su presentación gráfica. Si se crea una clase especializada en la graficación se presentaba nuevamente el problema de la sincronización con la presentación y el refresco en la presentación necesario por parte del applet para redibujar los resultados en el navegador ya que el método update() es parte de la implementación de la clase applet y habrá que recordar que la clase AVHRR_applet cual se había dado de alta como Runnable. En otras palabras, la clase AVHRR_applet realiza la manipulación de los hilos múltiples de ejecución y es quien se tiene que encargar de la presentación gráfica para mantener su coordinación en el navegador.

De esta manera, el diagrama de clases se redujo a 3 entre las que se encuentran: la clase AVHRR_applet quien inicializa el ambiente gráfico, mantiene el hilo de ejecución para la animación y presenta los resultados en el navegador; la clase procesa_imagen quien manipula digitalmente las imágenes y realiza el cálculo de las velocidades superficiales; y la clase DebugWin que se utiliza para la depuración del código y el seguimiento de los resultados. El diagrama resultante se muestra en la siguiente imagen.

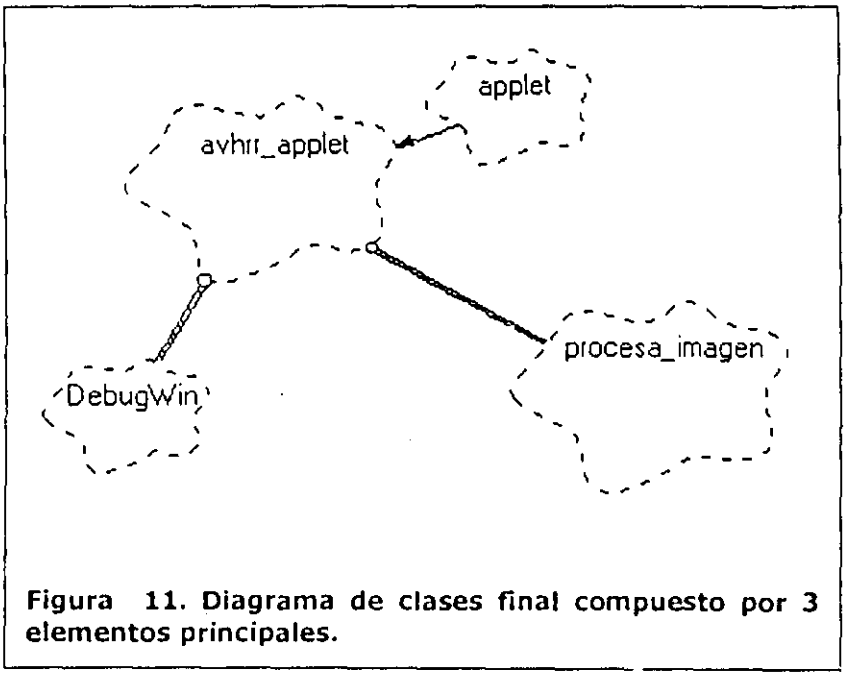

Figura 11. Diagrama de clases final compuesto por 3 elementos principales.

3.3 Programación

Como resultado de la codificación del problema de acuerdo al análisis y diseño orientado a objetos se crearon dos clases distintas: la primera de ellas (la principal: **AVHRR_applet.java**) es quien inicializa el programa y lleva el control del hilo de ejecución, delimitando el nombre de las imágenes que se van a trabajar, la creación de objetos que manipulen estas imágenes y las llamadas correspondientes a los objetos de la segunda clase (**Procesa_imagen.java**) para realizar el procesamiento de las imágenes así como los cálculos necesarios para la obtención del campo de velocidades. Por simplicidad, esta misma clase principal es quien se encarga de su presentación gráfica y la pseudoanimación de las imágenes y los resultados, dado que su presentación ocurre dentro de un visualizador gráfico de páginas HTML.

De acuerdo con esto, todo el problema se redujo, por simplicidad y rendimiento, a la creación de dos clases con una extensión baja de líneas de código dadas las bibliotecas de clases gráficas con las que cuenta el lenguaje Java.

3.3.1 CLASE AVHRR_applet.java

Esta clase es la principal y es quien inicializa el programa. Como se programó para funcionar como un applet de Java, esta deriva directamente de la clase java.applet.Applet heredando todas sus características con lo que se simplifica la programación al no tener que programar directamente la inicialización del ambiente gráfico ni la manera de dibujar en el visualizador, ya que estas características son conocidas por la clase Applet.

```
import java.awt.*;
import java.awt.image.*;
import java.lang.Thread;

public class avhrr_applet extends java.applet.Applet implements Runnable
{

// Ventana de seguimiento
//DebugWin dw = new DebugWin();
Thread animationThread=null;
int frameCounter;
Image avhrr_uno,avhrr_dos,avhrr1_inv,avhrr2_inv,alpha,borde;
int x;
int y;
int limitad;

int cajav;
int subvertice1,subvertice2;
int ciclo;
Procesa_imagen avhrr1_procesa,avhrr2_procesa;
Image[] frame1,frame2;
int frames=100000;
int delay=1000;
int frameCounter1=0;
int frameCounter2=0;
int cuentacoord=0;
int[] vertice1,vertice2;
int[] x1,y1,x2,y2;
int coord=100000;

// Tamaño de la caja
//int  caja=7;
```

```
double umbral=0.7;
//int  longitud=20;

int contador=0;

public void init() {

}

   // Este metodo se llama cuando el applet se hace visible
   public void start()
   {
      // Crear el hilo y correrlo
      if (animationThread == null)
      {
         animationThread = new Thread(this);
         animationThread.start();
      }
   }

   // Este metodo se llama cuando el applet se hace invisible
   public void stop()
   {
      // Se detiene el hilo de animacion
      animationThread = null;
   }

   public void run()
   {
      long time = System.currentTimeMillis();
      while (animationThread != null)
      {
         // se utiliza el reloj del sistema
         // para asegurarse que el tiempo entre cuadros es constante
         try
         {
            time += delay;
            Thread.sleep(Math.max(0,time - System.currentTimeMillis()));
         }
         catch (InterruptedException e){;
         // Se inicia la actualizacion de la pantalla
         repaint(); }
      }
   }

   // Graficacion de las imagenes utilizadas y generadas
   public  void paint(Graphics g)
   {


   // Inicializacion de los datos de la subventana y su longitud
   x=0;
   y=0;

   // Parametros obtenidos de la pagina html
   //int caja = Integer.parseInt(getParameter("caja"));
   //int longitud = Integer.parseInt(getParameter("longitud"));
   //String avhrr_uno_gif= getParameter("avhrr_uno");
   //String avhrr_dos_gif= getParameter("avhrr_dos");
   //String borde_gif= getParameter("borde");

   int caja=5;
   int longitud=51;
```

```
String avhrr_uno_gif="avhrr_uno.gif";
String avhrr_dos_gif="avhrr_dos.gif";
String borde_gif="borde.gif";
Font f = new Font("Helvetica",Font.BOLD,10);
g.setFont(f);
g.drawString("Resultados",5,220);
g.drawString(avhrr_uno_gif,200,220);
g.drawString(avhrr_dos_gif,400,220);

// Inicializando el arreglo de imagenes para la animacion
frame1 = new Image[frames];
frame2 = new Image[frames];
vertice1 = new int[frames];
vertice2 = new int[frames];
x1=new int[coord];
x2=new int[coord];
y1=new int[coord];
y2=new int[coord];


// Se capturan las dos imagenes a trabajar
showStatus("Espera mientras se cargan las imágenes ....");
avhrr_uno = getImage(getCodeBase(),avhrr_uno_gif);
avhrr_dos = getImage(getCodeBase(),avhrr_dos_gif);
borde= getImage(getCodeBase(),borde_gif);


// Se obtiene una subcaja de la primer imagen
avhrr1_procesa = new Procesa_imagen(avhrr_uno,x,y,longitud);
int base=avhrr1_procesa.ancho;
int altura=avhrr1_procesa.alto;
limitad=Math.round(longitud/2)-1;
//("base,altura = " +base+","+altura);
int currentFrame=0;

g.drawImage(borde,0,0,this);
g.drawImage(avhrr_uno,avhrr1_procesa.ancho+5,0,this);
g.drawString("altura"+altura,50,240);
g.drawString("base"+base,5,240);

for (int b=0; b < altura-longitud; b+=longitud, contador++)
{

 for (int a=0; a < base-longitud; a+=longitud, contador++)
 {

   avhrr1_procesa = new Procesa_imagen(avhrr_uno,a,b,longitud);

   // Se obtiene el pixel del centro de la subventana
   int ancho = avhrr1_procesa.ancho;
   // Pixel del vertice superior izquierdo de la caja en la subventana
   cajav=avhrr1_procesa.pixel_corrido(caja);
   subvertice1=(limitad+a)-cajav;
   subvertice2=(limitad+b)-cajav;
   //dw.print("subvertice1 = " + subvertice2);
   //dw.print("subvertice2 = " + subvertice2);
   // Se crea una imagen a partir del arreglo de pixels de la subventana
   avhrr1_inv=avhrr1_procesa.crear_imagen();
   // Se crea el arreglo de pixeles de la caja de la segunda imagen
   // y se crea la imagen
   avhrr2_procesa = new Procesa_imagen(avhrr_dos,subvertice1,subvertice2,caja);
   avhrr2_inv=avhrr2_procesa.crear_imagen();

   g.drawImage(avhrr_dos,avhrr1_procesa.ancho*2+5,0,this);
```

```
currentFrame=contador+1;
frame1[contador]=avhrr1_inv;
frame2[contador]=avhrr2_inv;
vertice1[contador]=a;
vertice2[contador]=b;


showStatus("Se inicializa la correlacion cruzada en las subregiones");
// Se inicializa la correlacion cruzada en las subregiones

double menor=1.0;
int w=0,z=0;
for (int j=0; j < longitud; j++)
{

 for (int i=0; i < longitud; i++)
 {
    double promedio1=avhrr1_procesa.promedio(i,j,ancho,caja);
    double promedio2=avhrr2_procesa.promedio(x,y,ancho,caja);

    ////dw.print("prom1 = " + promedio1);
    ////dw.print("prom2 = " + promedio2);

    showStatus("Correlación cruzada "+j+" de la caja número " +(i+1)+" de "+longitud+"...");

    double corr=avhrr1_procesa.correlacion(avhrr1_procesa,avhrr2_procesa,ancho,x,y,longitud,promedio1,promedio2);

         if (Math.abs(corr) < Math.abs(menor))
    {
     menor=corr;
     w=i;
     z=j;
     x1[contador]=a;
     y2[contador]=b;
     x2[contador]=w;
     y2[contador]=z;
     }

 }
}

if (menor <= umbral)
{
    //dw.print("Correlacion = " + menor);
    //dw.print("x , y = " +w+","+z);
    //dw.print("x1,y1,x2,y2 = " + x1[contador]+","+y1[contador]+","+x2[contador]+","+y2[contador]);
    if (x2[contador] < x1[contador] || y2[contador] < y1 [contador])
    {
    g.drawLine(subvertice1,subvertice2,x2[contador]+a,y2[contador]+b);
    g.fillOval(x2[contador]+a-2,y2[contador]+b-2,4,4);
    }
    else
    {
    g.drawLine(x2[contador]+a,y2[contador]+b,subvertice1,subvertice2);
    g.fillOval(subvertice1,subvertice2,4,4);
    }
}
//dw.print("a,b = " +a+ "," +b);


 // Se dibuja el cuadro
int n=(frameCounter1++)%frames;
int m=(frameCounter2++)%frames;
```

```
    int o=(cuentacoord++)%coord;
    ////showStatus("Ahora se dibuja la imagen " +n);
    //g.drawImage(frame2[n],avhrr1_procesa.ancho+longitud+10,0,this);
    //g.drawImage(frame1[n],vertice1[n],vertice2[m],this);
    }
  }
    g.drawImage(borde,0,0,this);
    g.drawString("Ejecución terminada",100,240);
  }

  public void update(Graphics g)
  {
     paint(g);
  }

   public boolean handleEvent(Event evt)
  { if (evt.id == Event.MOUSE_DOWN && loaded)
    { if (runner != null && runner.isAlive())
      { if (stopped)
        { showStatus("Click to stop");
          runner.resume();
        }
        else
        { showStatus("Click to restart");
          runner.suspend();
        }
        stopped = !stopped;
      }
      else
      { stopped = false;
        current = 0;
        runner = new Thread(this);
        runner.start();
      }
    }
    else return super.handleEvent(evt);
    return true;
  }

  public synchronized boolean imageUpdate(Image img, int infoflags,
    int x, int y, int width, int height)
  { if ((infoflags & ImageObserver.ALLBITS) != 0)
    { // Image is complete
      imageWidth = image.getWidth(null);
      imageHeight = image.getHeight(null);
      showStatus("Click to stop");
      loaded = true;
      notify();
      return false;
    }
    return true; // want more info
  }

  private Image image;
  private int imageCount;
  private int imageWidth = 0;
  private int imageHeight = 0;
  private String imageName;
  private Thread runner = null;
  private int current = 0;
  private boolean loaded = false;
  private boolean stopped = false;
}
```

3.3.2 CLASE Procesa_imagen.java

Esta clase es la encargada de la manipulación de las imágenes y contiene además los métodos para el cálculo de la variación en el espacio de los mapas de colores. Al ser una clase especializada solamente en la manipulación de imágenes, se cuidó de que no contuviera ningún método de graficación.

La clase principal necesita instanciar objetos de esta clase para poder acceder a los métodos de procesamiento de las imágenes.

```
// Clase que realiza la lectura de la imagen y guarda una copia de cada pixel en
// un arreglo

// Clase que realiza la lectura de la imagen y guarda una copia de cada pixel en
// un arreglo

import java.awt.*;
import java.awt.image.*;

public class Procesa_imagen extends avhrr_applet {
int ancho;
int alto;
int centrox;
int centroy;
int cajav;
//DebugWin dw = new DebugWin();


int[] pixels;

Procesa_imagen(Image imagen,int x_ventana,int y_ventana,int ancho_ventana)
{
  // Primero debemos esperar a que toda la imagen se lea
  espera_imagen(imagen);

  // Se captura el ancho y alto de la imagen
  ancho=imagen.getWidth(this);
  alto=imagen.getHeight(this);
  centrox=Math.round(ancho/2);
  centroy=Math.round(alto/2);

  // Se asigna la memoria para capturar todos los pixeles de la imagen
  pixels= new int[ancho*alto];

  // Se capturan todos los pixeles

  PixelGrabber pg = new PixelGrabber(imagen,x_ventana,y_ventana,ancho_ventana,
  ancho_ventana,pixels,0,ancho);
    try {
    pg.grabPixels();
    } catch (InterruptedException e){
      System.err.println("Se interrumpio la espera por los pixeles");
      return;
    }

  }  // fin del metodo procesa_imagen()


// Subrutina que establece el valor del recorrido del pixel
// del vertice de la subventana
int pixel_corrido(int caja)
  {
```

```
if (caja == 11) cajav=5;
if (caja == 9)  cajav=4;
if (caja == 7)  cajav=3;
if (caja == 5)  cajav=2;
if (caja == 3)  cajav=1;
return cajav;
}


// esta rutina asegura que se leyeron todos los pixeles de la imagen
void espera_imagen(Image imagen)
{
  MediaTracker seguidor = new MediaTracker(this);
  seguidor.addImage(imagen,0);
  try {
    seguidor.waitForID(0);
  } catch (InterruptedException e){};
}

// Transparencia de la imagen

Procesa_imagen transparencia (Procesa_imagen img, int alto, int longitud)
 {
    for (int i=0; i < longitud*alto; i++)
     {
      // Codificación de los componentes de cada pixel
          int c=img.pixels[i];
          int alpha=(c&0x00000000)>>24;
          int r=(c&0xff0000)>>16;
          int g=(c&0x00ff00)>>8;
          int b=c&0xff;
     }
     return img;
 }



// rutina crea otra imagen al vuelo
Image crear_imagen()
{
  Image imagen=createImage(new MemoryImageSource(ancho,alto,pixels,0,ancho));
  int[] orgpix=pixels;
  pixels= new int[alto*ancho];
  System.arraycopy(orgpix,0,pixels,0,alto*ancho);
  return imagen;
}

    // primera parte de la correlacion

double promedio(int x, int y, int ancho, int caja)
{
       double suma=0;
       //int[] srcpix=pixels;
       //pixels= new int[ancho*alto];


       for (int z=0; z < caja; z++, x++)
       {

          for (int w=0; w < caja; w++, y++)
          {
            int c=pixels[y*ancho+x];
            int r=(c&0xff0000)>>16;
            int g=(c&0x00ff00)>>8;
```

```
            int b=c&0xff;
            suma+=((double)r+(double)g+(double)b)/3;
          }
        }

        double prom=suma/(caja*caja);
        //dw.print("promedio = " + prom);
        return prom;
  }

// Funcion que calcula la correlacion cruzada

 double correlacion(Procesa_Imagen Img1, Procesa_Imagen Img2, int ancho, int x, int y, int longitud, double media1, double
media2)
   {
      double a=0.0,b=0.0,c=0.0,d=0.0,e=0.0,f=0.0;

      //segunda imagen, segundo indice
      int c2=Img2.pixels[y*ancho+x];
      int r2=(c2&0xff0000)>>16;
      int g2=(c2&0x00ff00)>>8;
      int b2=c2&0xff;
      //dw.print("r2 = " + r2);
      //dw.print("g2 = " + g2);
      //dw.print("b2 = " + b2);
      int indice=0;

       for (int l=0; l < longitud; l++)
       {
         //dw.print("l = " + (l));
         for (int k=indice; k < longitud+indice; k++)
         {
            // Codificación de los componentes de cada pixel
            // primer imagen, primer indice
            int c1=Img1.pixels[k];
            int r1=(c1&0xff0000)>>16;
            int g1=(c1&0x00ff00)>>8;
            int b1=c1&0xff;
            //dw.print("k =  " +  k);
            //dw.print("r1 = " + r1);
            //dw.print("g1 = " + g1);
            //dw.print("b1 = " + b1);
            // primer Imagen, segundo indice
            //int c11=Img1.pixels[j*ancho+(i+1)];
            //int r11=(c11&0xff0000)>>16;
            //int g11=(c11&0x00ff00)>>8;
            //int b11=c11&0xff;
            //dw.print("r11 = " + r11);
            //dw.print("g11 = " + g11);
            //dw.print("b11 = " + b11);

            // Numerador de la division de la correlación
            a=(((double)r1+(double)g1+(double)b1)/3) - media1; // primer termino
            b=(((double)r2+(double)g2+(double)b2)/3) - media2; //segundo termino
            c+=(a*b);
            // Denominador de la division de la correlacion
            d+= a*a; // primer termino
            e+= b*b; // segundo termino
            //dw.print("a = " + a);
            //dw.print("b = " + b);
            //dw.print("c = " + c);
            //dw.print("d = " + d);
            //dw.print("e = " + e);
```

```
        }
        indice+=ancho;

      }

        f= Math.sqrt(d*e); //raiz cuadrada del denominador
        double r=c/f; //correlacion cruzada
        //dw.print("a = " + a);
        //dw.print("b = " + b);
        //dw.print("c = " + c);
        //dw.print("d = " + d);
        //dw.print("e = " + e);
        //dw.print("f = " + f);
        //dw.print("r = " + r);
      return r;
    }
  }
}
```

3.3.3 CLASE DebugWin.java

Esta clase crea una ventana de visualización de los resultados numéricos de todos los cálculos que realiza la clase Procesa_Imagen. Su empleo, además de ser útil para la depuración del código, se fundamenta en que el usuario pueda obtener información precisa de lo que esta observando gráficamente en su pantalla. Solamente se empleó en el proceso de depuración pero no se incluyó en el sistema final.

```
/*
 * Gary Cornell and Cay S. Horstmann, Core Java (Book/CD-ROM)
 * Published By SunSoft Press/Prentice-Hall
 * Copyright (C) 1996 Sun Microsystems Inc.
 * All Rights Reserved. ISBN 0-13-565755-5
 *
 * Permission to use, copy, modify, and distribute this
 * software and its documentation for NON-COMMERCIAL purposes
 * and without fee is hereby granted provided that this
 * copyright notice appears in all copies.
 *
 * THE AUTHORS AND PUBLISHER MAKE NO REPRESENTATIONS OR
 * WARRANTIES ABOUT THE SUITABILITY OF THE SOFTWARE, EITHER
 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE, OR NON-INFRINGEMENT. THE AUTHORS
 * AND PUBLISHER SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED
 * BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING
 * THIS SOFTWARE OR ITS DERIVATIVES.
 */

/**
 * @version 1.00 07 Feb 1996
 * @author Cay Horstmann
 */

import java.awt.*;

class DebugWin extends Frame
{  public void print(Object ob)
   {  output.appendText("\n" + ob);
   }

   public DebugWin()
   {  setTitle("Resultados");
```

```
    output.setEditable(false);
    output.setText("[Resultados de la correlación]");
    add("Center",output);
    resize(300, 200);
    show();
  }

  public boolean handleEvent(Event e)
  { if (e.id == Event.WINDOW_DESTROY)
      hide();
    else return super.handleEvent(e);
    return true;
  }
  private TextArea output = new TextArea();
}
```

3.4 Cálculo del campo de velocidad superficial del agua de mar

3.4.1 Obtención de las imágenes

Las imágenes que se emplearon en este estudio se obtuvieron mediante una estación receptora de imágenes, instalada en el Instituto de Geografía de la Universidad Nacional Autónoma de México. Esta estación recibe continuamente imágenes de los satélites de la NOOA con sensores del tipo AVHRR.

Las imágenes utilizadas se seleccionaron en base a las siguientes características:

- Cobertura espacial amplia
- Cobertura nubosa baja
- El tiempo de muestreo entre imágenes sucesivas menor a 12 horas
- El área de estudio situada al cenit de la cobertura espacial, esto es, al centro de la imagen

A continuación, se muestra la relación de las imágenes que cumplieron con estos requisitos:

Tabla 2. Relación de archivos, fechas, horas y bandas de muestreo de las imágenes AVHRR seleccionadas para este estudio. Todas ellas corresponden a una región del pacífico occidental de la República Mexicana de aproximadamente 200 kilómetros por lado frente a las costas de Lázaro Cárdenas, Michoacán. Entre el 11 de diciembre de 1996 y el 24 de enero de 1997.

Nombre del archivo	Fecha de muestreo	Hora	Banda
96121108.gif	11-12-96	8:00	5
96121208.gif	12-12-96	8:00	5
96121308.gif	13-12-96	8:00	5
97011913.gif	19-01-97	13:00	5
97012013.gif	20-01-97	13:00	5
97012113.gif	21-01-97	13:00	5
97012313.gif	23-01-97	13:00	5
97012413.gif	24-01-97	13:00	5
97012513.gif	25-01-97	13:00	5
97012913.gif	29-01-97	13:00	5
97013013.gif	30-01-97	13:00	5
980329.mov	29-03-98	Película GOES sobre el movimiento de nubosidad sobre IOWA	

La selección de las imágenes a utilizar involucró un conjunto de problemas operativos, ya que dependen de una serie de variables atmosféricas que pueden hacer que una imagen sea completamente inutilizable.

Debido a que la diferencia de tiempo entre imágenes no debe ser mayor a 12 horas, fue frecuente desechar imágenes que sobrepasaran este tiempo, pues las trayectorias de los diferentes satélites que censan el área de estudio, a veces presentaban bitácoras de paso mayores a este tiempo límite.

La obtención y selección de la imagen a emplear, requiere de una gran intervención por parte del técnico encargado del manejo y almacenamiento de las mismas, por lo que la automatización de todo este proceso resulta hasta el momento imposible de implementar. Esta situación hace que el análisis de las imágenes y su posterior exposición al usuario final no pueda hacerse en tiempo real. Limitando seriamente el alcance de los resultados del campo de velocidades extraído de las propias imágenes.

3.4.2 Interfase para el usuario

Un aspecto que se cuidó en la elaboración de este estudio fue la accesibilidad del sistema en múltiples plataformas siendo ésta una de las razones para seleccionar a Java como plataforma de desarrollo.

En este sentido, se optó por que la interfase con el usuario fuera lo más simplificada posible por lo que se generó una página electrónica del tipo HTML para ello (figura 12). En ésta, el usuario puede seleccionar varios parámetros con los cuales se alimenta al applet para su ejecución y son los siguientes:

- Tamaño en pixeles de la caja de búsqueda
- Tamaño en pixeles de la caja patrón
- Primer imagen de la secuencia
- Segunda imagen de la secuencia
- Archivo de referencia geográfica en los resultados
- Alto y ancho del applet

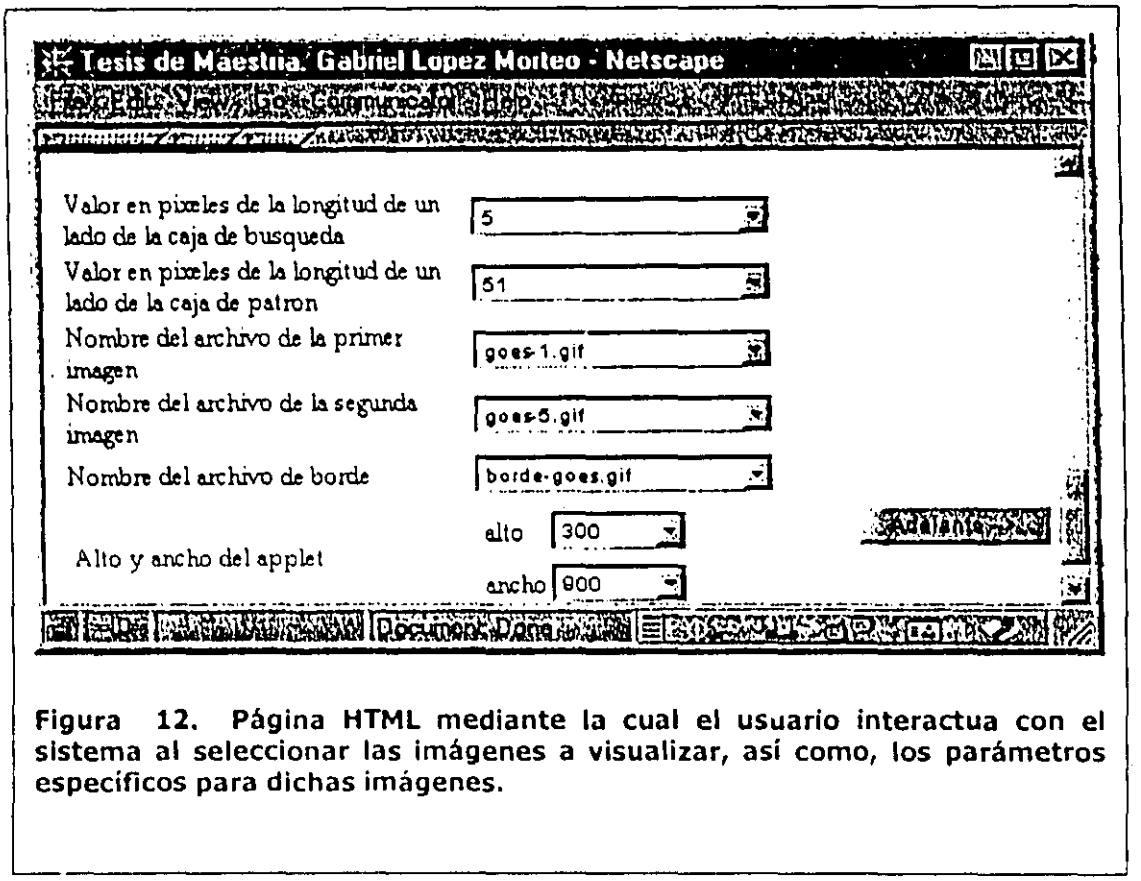

Figura 12. Página HTML mediante la cual el usuario interactua con el sistema al seleccionar las imágenes a visualizar, así como, los parámetros específicos para dichas imágenes.

Cuando el usuario envía la forma para su ejecución, se corre un programa del tipo CGI (Interfase común en la compuerta o Common Gateway Interface) en el servidor que crea una

página HTML dinámica en donde se incluye al applet y se le definen los valores de los parámetros seleccionados que van a ser leídos por el applet y comenzar la ejecución.

El programa CGI está escrito en el lenguaje de guión PERL garantizando con ello que se puede ejecutar prácticamente en cualquier plataforma para el que exista el interprete de dicho lenguaje. Tanto las páginas electrónicas, las imágenes y las clases de Java pueden situarse en cualquier servidor WEB para su empleo. El código de dicho CGI se muestra a continuación:

```
#!/usr/bin/perl
#
# Gabriel Alejandro López Morteo

# Declaración de librerias

use DBI;
use CGI qw/:standard :HTML3 -no_debug/;
use DBI::DBD;

# Se instancia un documento CGI

$query=new CGI;

print $query->header(-expire=>'0');
print $query->start_HTML(-title=>'AVHRR_applet.',-BGCOLOR=>'#000099',
-TEXT=>'#FFFFFF',-LINK=>'#FFCC00',-VLINK=>'#51188E', -ALINK=>'#FFFF00');

#Declaración de variables

$longitud=$query->param("longitud");
$caja=$query->param("caja");
$AVHRR_uno=$query->param("AVHRR_uno");
$AVHRR_dos=$query->param("AVHRR_dos");
$borde=$query->param("borde");
print $anchoapplet=$query->param("anchoapplet");
print $altoapplet=$query->param("altoapplet");

print "<HR><center>Campo de velocidades superficiales de una parcela marina a
 partir de imagenes del tipo AVHRR</center><hr>";
print "<APPLET CODE='AVHRR_applet.class' WIDTH='$anchoapplet' HEIGHT='$altoapplet'>";
print "<PARAM NAME=caja VALUE='$caja'>";
print "<PARAM NAME=longitud VALUE='$longitud'>";
print "<PARAM NAME=AVHRR_uno VALUE='$AVHRR_uno'>";
print "<PARAM NAME=AVHRR_dos VALUE='$AVHRR_dos'>";
print "<PARAM NAME=borde VALUE='$borde'>";
print "</APPLET>";

print $query->end_HTML;
```

Este guión toma los valores de los campos de la página HTML y crea una página nueva dinámica que se regresa al navegador. En esta página se llama a ejecución el applet AVHRR_applet.class, se define el alto y el ancho del área de trabajo (datos que se obtuvieron de la página HTML anterior) y se declaran etiquetas <PARAM> que definen los valores de los parámetros que emplea el applet y que van a ser leídos por el mismo.

3.4.3 Resultados de la ejecución del programa

El programa se ejecutó en una secuencia de imágenes de tres tipos: la primera de ellas corresponde a un par de imágenes de control completamente en blanco pero con un cuadrado

página HTML dinámica en donde se incluye al applet y se le definen los valores de los parámetros seleccionados que van a ser leídos por el applet y comenzar la ejecución.

El programa CGI está escrito en el lenguaje de guión PERL garantizando con ello que se puede ejecutar prácticamente en cualquier plataforma para el que exista el interprete de dicho lenguaje. Tanto las páginas electrónicas, las imágenes y las clases de Java pueden situarse en cualquier servidor WEB para su empleo. El código de dicho CGI se muestra a continuación:

```
#!/usr/bin/perl
#
# Gabriel Alejandro López Morteo

# Declaración de librerias

use DBI;
use CGI qw/:standard :HTML3 -no_debug/;
use DBI::DBD;

# Se instancia un documento CGI

$query=new CGI;

print $query->header(-expire=>'0');
print $query->start_HTML(-title=>'AVHRR_applet.',-BGCOLOR=>'#000099',
-TEXT=>'#FFFFFF',-LINK=>'#FFCC00',-VLINK=>'#51188E', -ALINK=>'#FFFF00');

#Declaración de vanables

$longitud=$query->param("longitud");
$caja=$query->param("caja");
$AVHRR_uno=$query->param("AVHRR_uno");
$AVHRR_dos=$query->param("AVHRR_dos");
$borde=$query->param("borde");
print $anchoapplet=$query->param("anchoapplet");
print $altoapplet=$query->param("altoapplet");

print "<HR><center>Campo de velocidades superficiales de una parcela marina a
 partir de imagenes del tipo AVHRR</center><hr>";
print "<APPLET CODE='AVHRR_applet.class' WIDTH='$anchoapplet' HEIGHT='$altoapplet'>";
print "<PARAM NAME=caja VALUE='$caja'>";
print "<PARAM NAME=longitud VALUE='$longitud'>";
print "<PARAM NAME=AVHRR_uno VALUE='$AVHRR_uno'>";
print "<PARAM NAME=AVHRR_dos VALUE='$AVHRR_dos'>";
print "<PARAM NAME=borde VALUE='$borde'>";
print "</APPLET>";

print $query->end_HTML;
```

Este guión toma los valores de los campos de la página HTML y crea una página nueva dinámica que se regresa al navegador. En esta página se llama a ejecución el applet AVHRR_applet.class, se define el alto y el ancho del área de trabajo (datos que se obtuvieron de la página HTML anterior) y se declaran etiquetas <PARAM> que definen los valores de los parámetros que emplea el applet y que van a ser leídos por el mismo.

3.4.3 Resultados de la ejecución del programa

El programa se ejecutó en una secuencia de imágenes de tres tipos: la primera de ellas corresponde a un par de imágenes de control completamente en blanco pero con un cuadrado

negro desfasado espacialmente con un tamaño de 200 por 200 pixeles; la segunda secuencia corresponde a una secuencia de imágenes del tipo AVHRR de la costa frente a Lazaro Cárdenas, Michoacán; y la tercera es una secuencia de imágenes climatológicas del tipo GOES correspondiendo a las bandas del infrarrojo cercano del movimiento de nubes.

En el primer caso, al ejecutarse el programa encuentra de una manera correcta el movimiento producto del desfasamiento de los cuadrados negros. Sin embargo se observó una característica importante: el movimiento se detecta pero sin registrar el sentido del movimiento, registrando exclusivamente la dirección del mismo (figuras 13 y 14).

Este comportamiento puede deberse a cambios en el signo del resultado producto de las operaciones matemáticas que se realizan en el cálculo del desplazamiento ya que no existe codificación matemática, en el modelo empleado, que delimite el sentido del cambio sino únicamente la diferencia de la posición de los valores de colores.

Se seleccionó este tipo de figura pues representa el contraste máximo entre el cuadrado negro y el fondo blanco en cualquier esquema de colores pues el valor promedio de cada pixel en el esquema rgb siempre será 0 para el color negro y 255 para el blanco.

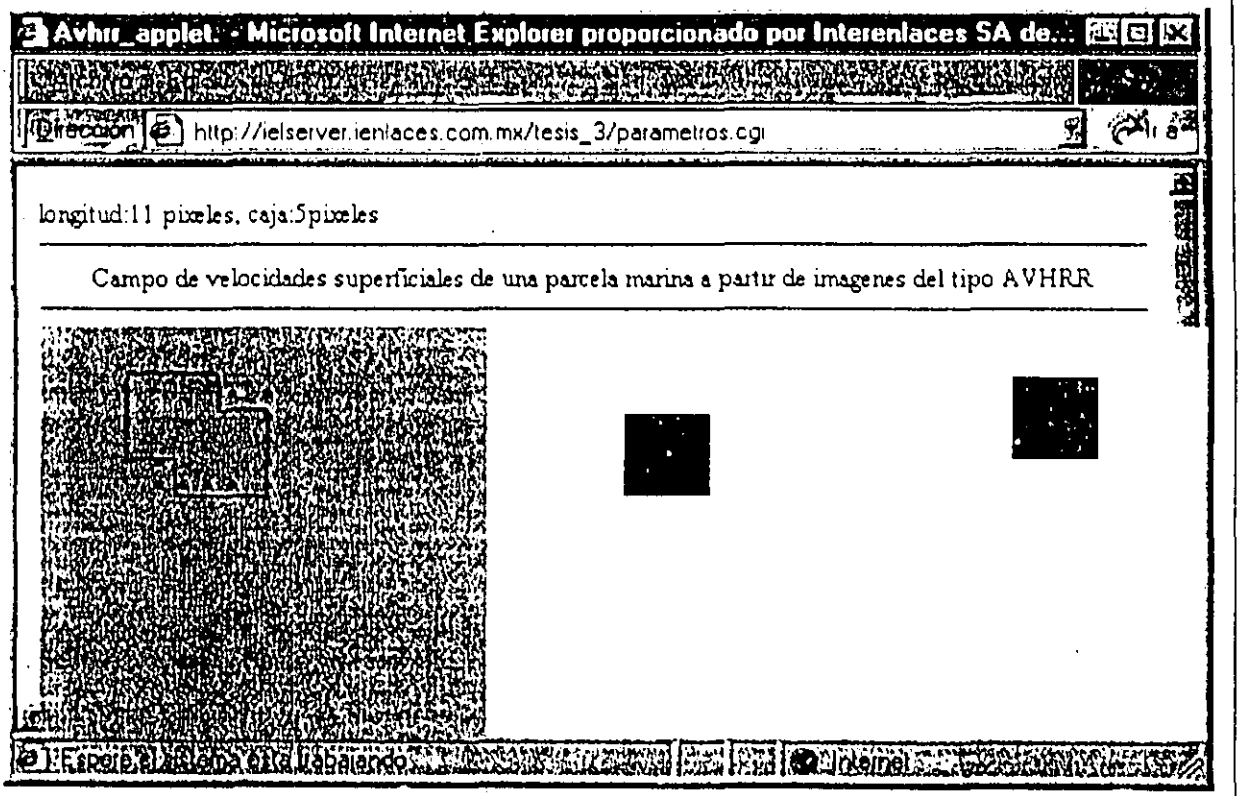

Figura 13. Ejecución del programa con el par de imágenes de control, con la longitud de la caja patrón de 11 pixeles y la longitud de la caja de búsqueda de 5 pixeles. Se observa que el programa detecta el desfasamiento de los cuadrados negros.

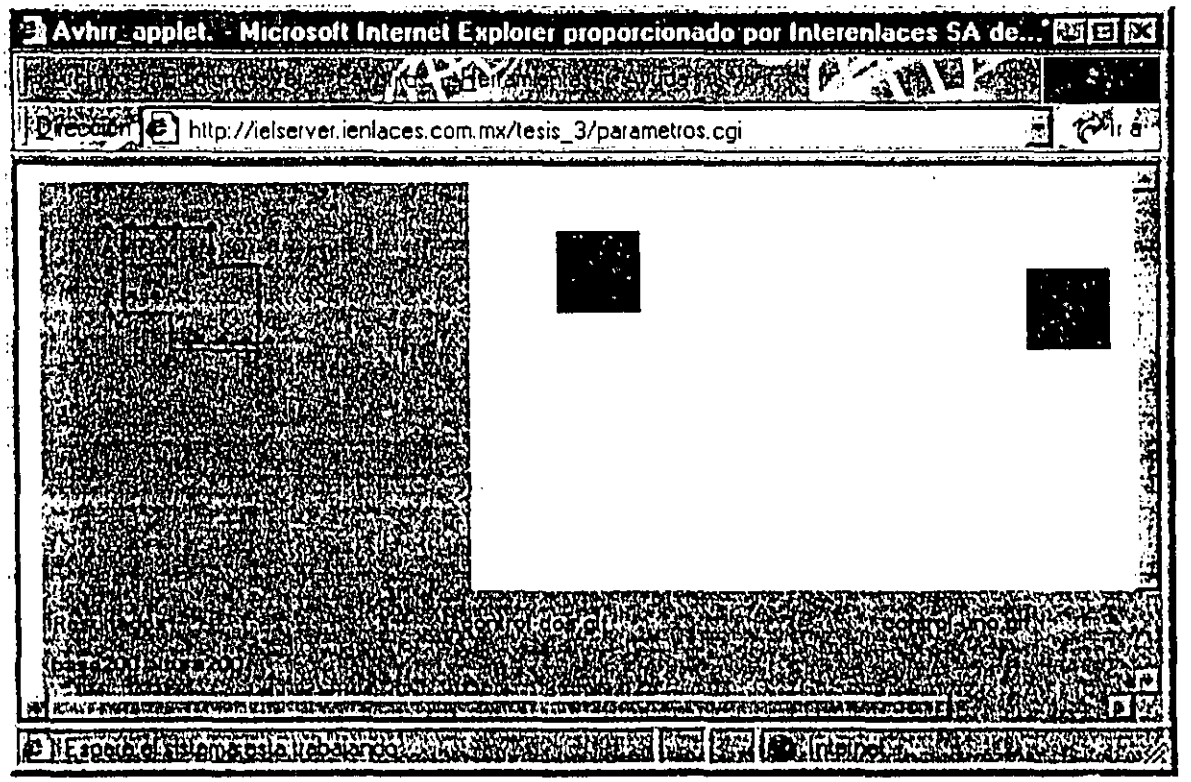

Figura 14. **La misma corrida que la anterior pero invirtiendo el orden entre las imágenes. Se observa nuevamente que el sistema detecta el desfasamiento entre los cuadrados negros, pero registra solamente la dirección y no el sentido del mismo.**

En imágenes compuestas por colores este esquema consiste en una limitante importante en la funcionalidad del programa pues al tener cada pixel un valor producto de la combinación de tres valores de colores (rojo, verde y azul), se requirió encontrar el valor promedio de los mismos como el representante de cada pixel por lo que no es exactamente el valor real del pixel.

Por otro lado, el sistema es sensible a la selección inicial de los tamaños de las cajas de búsqueda y la caja patrón pues si el valor es demasiado grande, se toman en cuenta más valores de los pixeles vecinos con lo cual el valor promedio del color representativo de la caja puede variar demasiado de la realidad. En el caso de este par de imágenes, al tener solamente dos valores de colores y al aumentar los valores de las cajas ya mencionadas se registra bien el desfasamiento espacial, pero esto no necesariamente ocurrirá en imágenes más complejas (figuras 15 y 16).

El funcionamiento del programa en imágenes complejas como las del tipo AVHRR, presenta un patrón de comportamiento más intrincado, sin embargo se puede observar que se continua detectando los desfasamientos entre los mapas de colores en secciones específicas de la imagen, especialmente en aquellas zonas que realmente son representativas. Es decir, en aquellas áreas en donde no existe cobertura nubosa, representada por las áreas coloreadas, como la que se encuentra en la esquina inferior izquierda de la imagen (figuras 17 y 18).

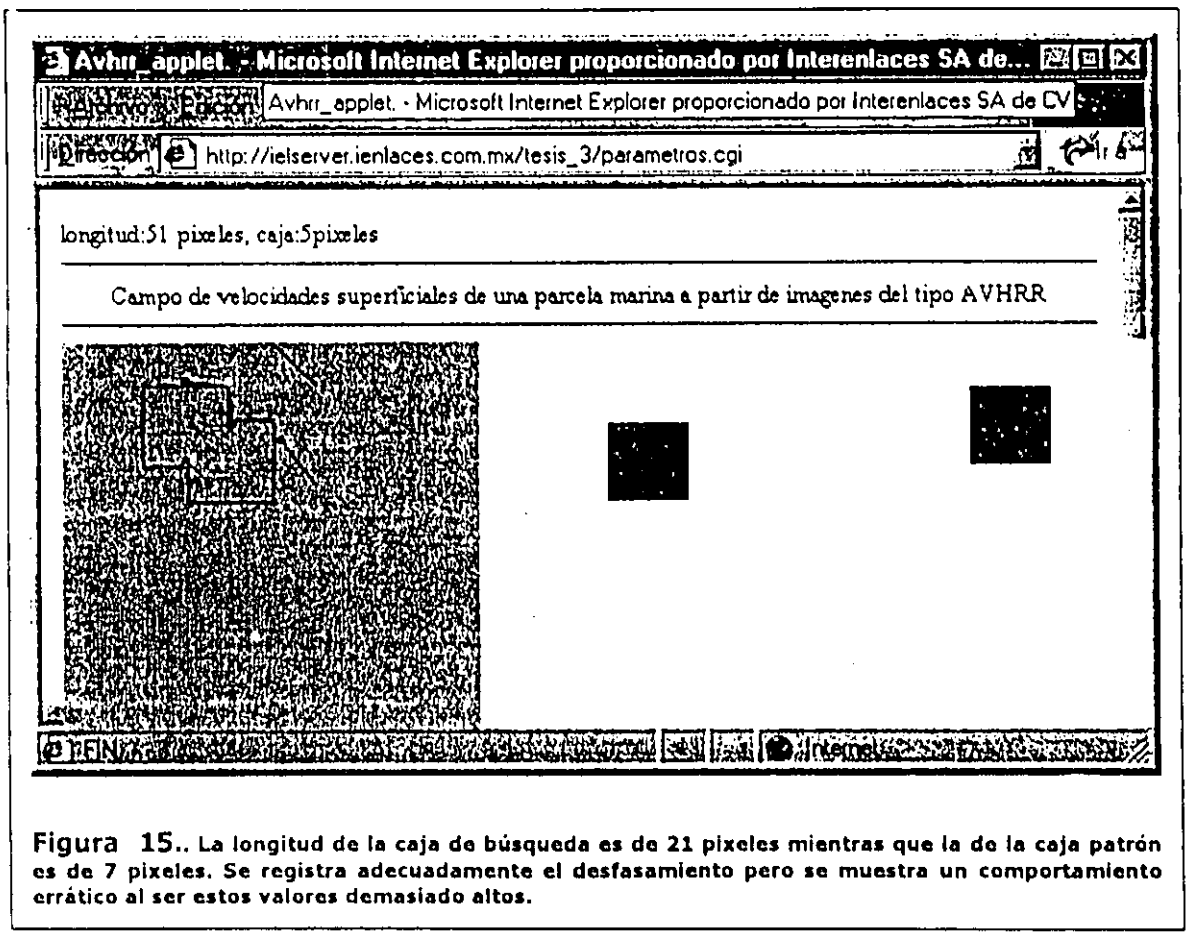

Figura 15.. La longitud de la caja de búsqueda es de 21 pixeles mientras que la de la caja patrón es de 7 pixeles. Se registra adecuadamente el desfasamiento pero se muestra un comportamiento errático al ser estos valores demasiado altos.

Lamentablemente la estructura de las imágenes de este tipo dista mucho de presentar un patrón ideal, además de que resulta particularmente difícil encontrar una secuencia de imágenes completamente adecuadas para ser utilizadas abiertamente por el programa. En este sentido, deben de tomarse con reserva los resultados del programa considerando la introducción de "ruido" producido por las regiones cubiertas por nubes y que necesariamente son incluidas dentro de las cajas de búsqueda y la patrón.

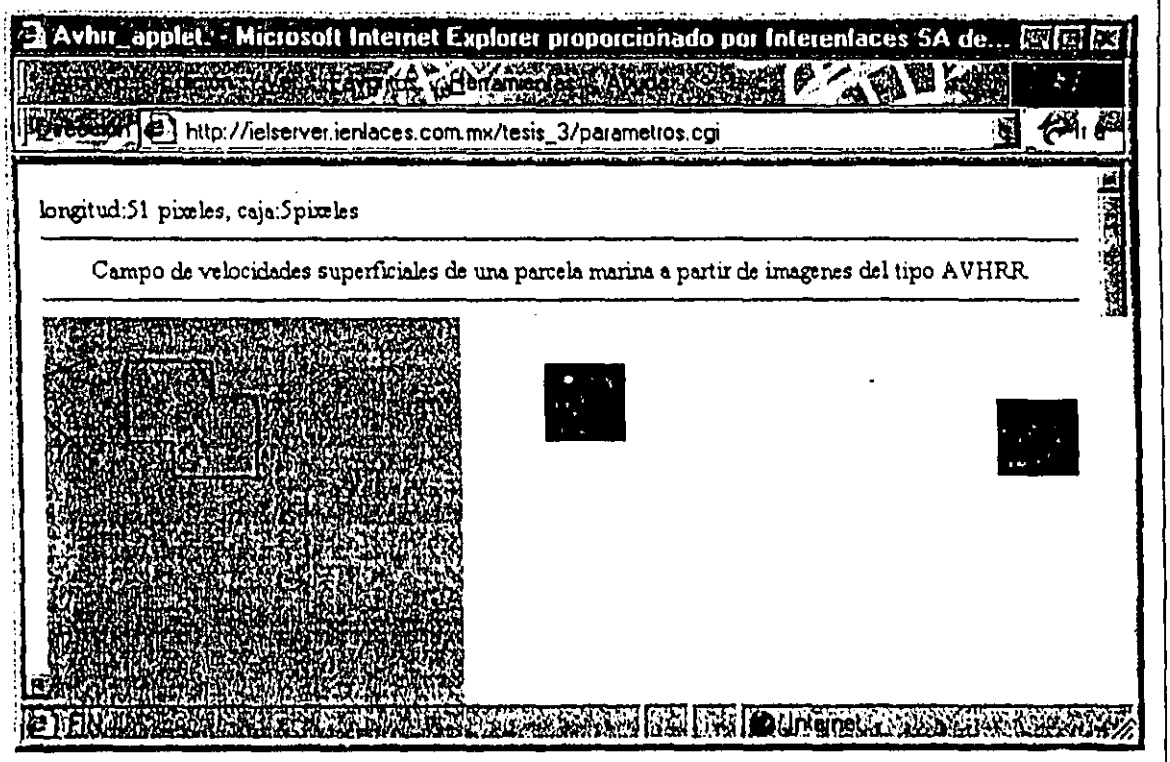

Figura 16. **Se observa el mismo comportamiento al invertir el orden de las imágenes sin detectar aún el sentido del desfasamiento**

Este problema puede ser disminuido mediante un procesamiento adicional en las imágenes originales al seleccionar exclusivamente las áreas que presenten condiciones propicias para su estudio y al aumentar el contraste entre los mapas de colores, además de disminuir la paleta de colores al promediar los valores de color de los pixeles vecinos para simplificar la complejidad de las estructuras de colores de la imagen.

Figura 17. El mismo programa aplicado a imágenes complejas del tipo AVHRR. En la parte más representativa de la imagen situada en la esquina inferior izquierda de la misma, se observa un área de coloración roja, la cual el sistema representa con un desplazamiento mínimo. Aunque estas imágenes estan desfasadas en tiempo por un periódo considerable, se seleccionaron por presentar patrones similares en el mapa de colores.

Figura 18. Se presentan las mismas imágenes pero invertidas en su orden. El comportamiento del programa se muestra similar a la anterior en la región representativa de las imágenes.

El tercer juego de imágenes corresponde a una secuencia meteorológica de la serie GOES en el infrarrojo cercano de la temperatura de la cobertura nubosa sobre el estado de Iowa en los Estados Unidos de Norteamérica (figuras 19 y 20)

Los resultados obtenidos con estas imágenes son muy semejantes a las obtenidas con la serie AVHRR pero con una gran ventaja: en esta secuencia, la estructura de las nubes es más regular y menos compleja que la de temperatura superficial de agua de mar, por lo que el análisis de los resultados se vuelve más claro.

3.4.4 Desempeño del programa

La promesa que Java introdujo en el mundo de la informática en donde se "codifica una vez y se corre en donde sea" aumenta de manera significativa la penetración de la tecnología en áreas y campos en las que anteriormente se consideraba como una herramienta para el laboratorio pero difícilmente útil para la vida diaria. En este sentido hay mucho que decir para que esa promesa se convierta en una realidad palpable para la mayoría de los usuarios.

Uno de los problemas que se detectaron durante las corridas del programa, corresponde al empleo de los recursos del sistema por parte de la máquina virtual de Java, especialmente en lo que se refiere a la utilización de la memoria del equipo. Java es un tremendo devorador de la misma y más aún cuando requiere hacer una cantidad tan importante de operaciones de punto flotante conforme se seleccionan valores más pequeños en las cajas de búsqueda y patrón (figura 21). Pudiendo llegar hasta los 10 millones de operaciones cuando se analizan imágenes de 400 pixeles por 400 pixeles.

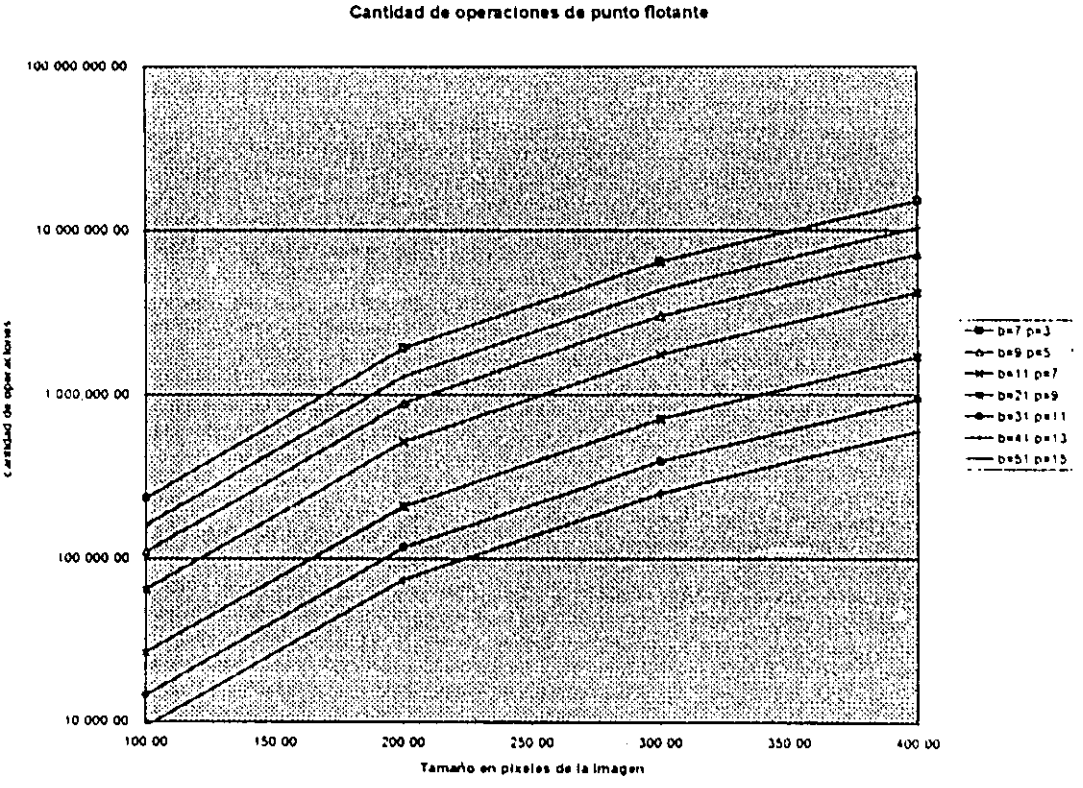

Figura 21. Cantidad de operaciones de punto flotante que se realizan dependiendo del tamaño de las imágenes, el valor de la caja de búsqueda (b) y de la caja patrón (p).

Al agotar los recursos en RAM del equipo, se comienza a utilizar el área de memoria virtual del disco (swap) para continuar trabajando. Al parecer, el recolector de basura de Java no es capaz de realizar una liberación exitosa de memoria cuando existe una ejecución intensiva de los recursos del sistema, por lo que ésta no se libera realmente, sino que es almacenada en swap para que un nuevo objeto ocupe su lugar en RAM. Por lo que se observa un aumento muy significativo en la utilización de esta área llegando a ocupar hasta 450 megabytes en la ejecución de las imágenes más grandes. Sin embargo es notable que mientras no se acabe el espacio en disco, la ejecución del programa continúa hasta terminar aunque con una notable degradación de la ejecución. Desgraciadamente esto no ocurre en todos los sistemas operativos. Los sistemas operativos Windows (95,98 y NT) tienen la capacidad de aumentar dinámicamente el espacio de swap conforme se va requiriendo pues emplean un archivo guardado en el sistema de archivos, pudiendo éste crecer hasta que se agote el espacio físico. En el caso de los sistemas UNIX, éstos tienen reservada una partición en disco para swap de tamaño finito con el propósito de hacer más eficiente la escritura hacia el disco de ésta memoria virtual. Así cuando la memoria RAM se agota y comienza la escritura en el área de swap, la ejecución del programa se degrada y continúa hasta que ésta se llene. Entonces se dispara una excepción de Java con el mensaje de recursos agotados.

Este comportamiento refleja una desventaja considerable del lenguaje con respecto a los lenguajes compilados en el manejo dinámico y manual de la memoria. De hecho, si es posible manipular manualmente el empleo de la memoria en Java, pero representa una sobrecarga en la ejecución del programa y es muy complicada su codificación con lo que la facilidad de escribir código transportable fácilmente se pierde.

Aún existe otro problema más: las máquinas virtuales de Java varían dependiendo del tipo de navegador de páginas electrónicas que se emplee e incluso entre las diferentes versiones del mismo navegador, especialmente en lo que se refiere a las operaciones de punto flotante. Esto dificulta tener resultados completamente consistentes pues la capacidad instalada de diferentes versiones y tipos de navegadores es muy amplia como para estandarizar los resultados. Sin embargo, el resultado global sigue siendo correcto.

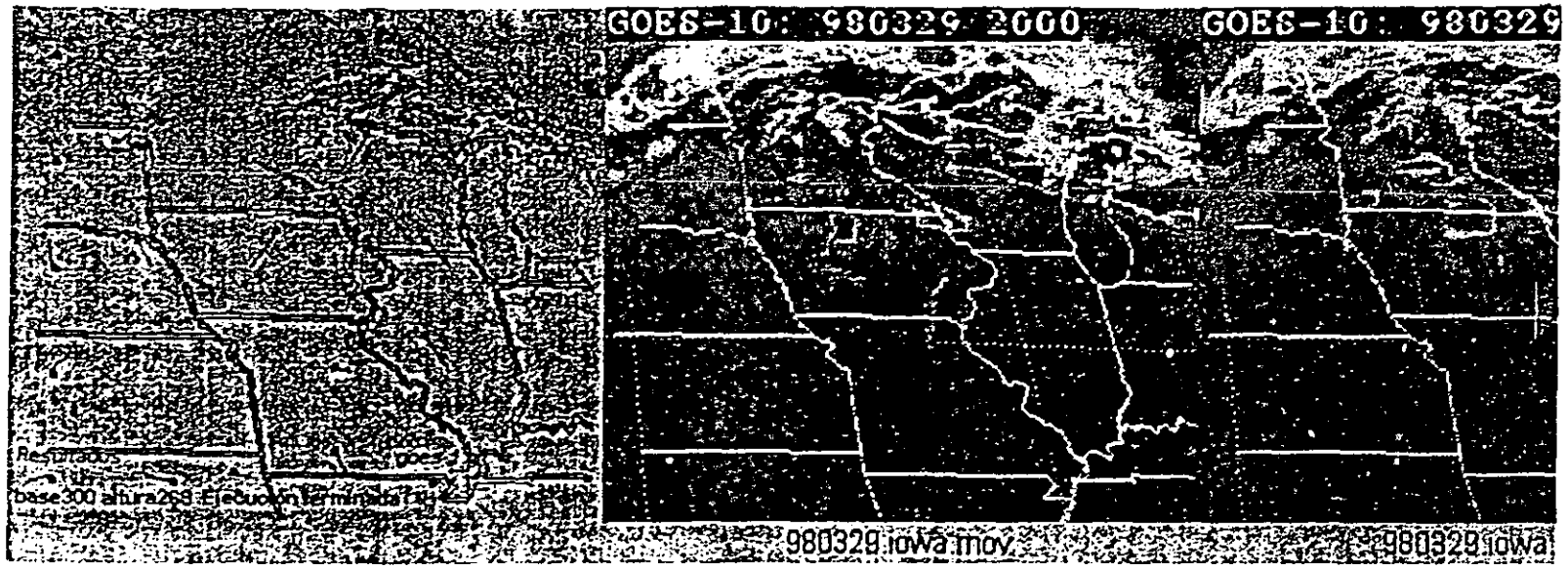

Figura 19. Serie de imágenes del tipo GOES representando temperaturas en la cobertura nubosa. Nótese el gran contraste en el mapa de colores. El valor de la caja de búsqueda es de 5 pixeles, mientras que el tamaño de la caja patrón es de 51 pixeles.

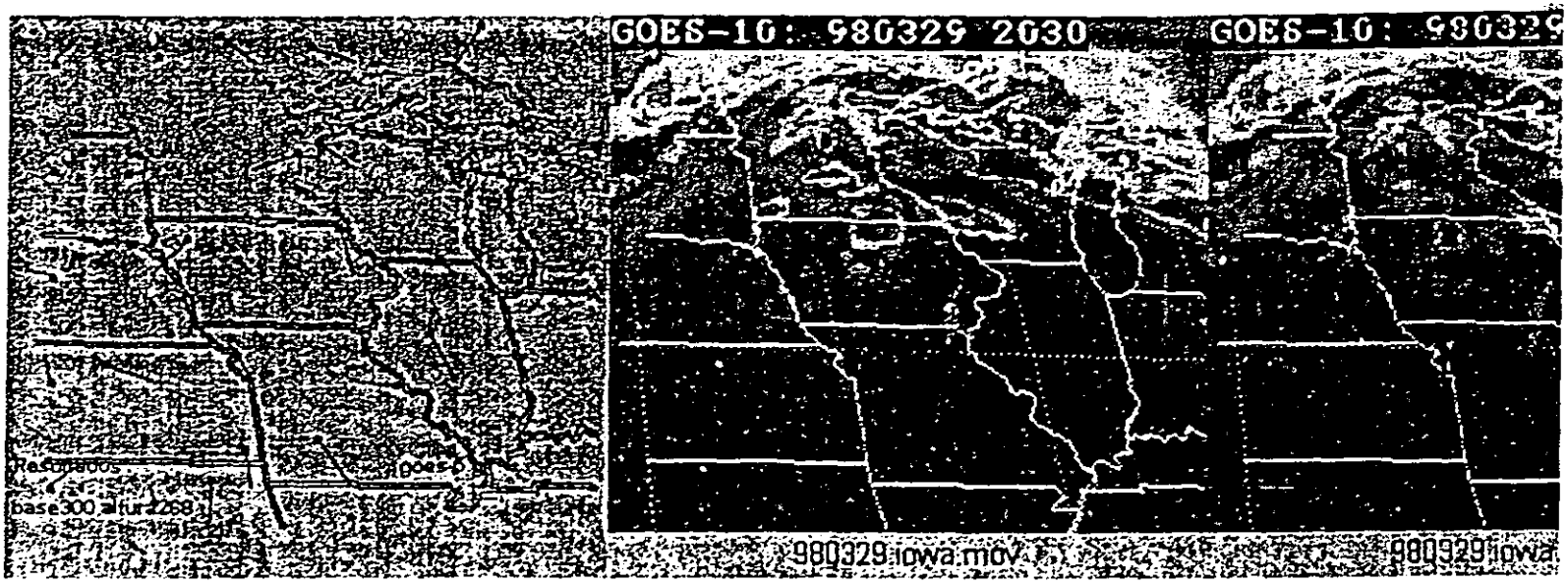

Figura 20. La misma secuencia anterior pero ahora invirtiendo el orden de las imagenes. El valor de la caja de búsqueda es de 5 pixeles, mientras que el tamaño de la caja patrón es de 51 pixeles.

4 DISCUSIONES

Los resultados obtenidos durante el presente proyecto mostraron ser consistentes con el método empleado, aunque éstos no fueron relacionados con mediciones reales para su corroboración, lo cual se propone como una segunda etapa de este estudio para ser desarrollada en conjunción con profesionales del área de la oceanografía. Sin embargo, el método empleado arrojó resultados satisfactorios a primera vista además de que simplifica en mucho el empleo e interpretación de los mismos por parte de usuarios expertos y no expertos en el área.

Por su parte, la metodología orientada a objetos representa una alternativa real y viable para el investigador de cualquier área, para desarrollar rápidamente aplicaciones que sean funcionales y que tengan un componente tecnológico informático de gran calidad. Además, el análisis y diseño empleado ayuda en gran medida a manejar no solamente la complejidad del problema computacional, sino que también, es un invaluable auxiliar en el entendimiento y granularización del problema oceanográfico en si, al permitir delimitar claramente a todos los actores del problema y enfocar la atención debida a la comprensión y desarrollo de cada uno de ellos. Manipulando adecuadamente la relaciones e interdependencias que existen o pudiera existir entre cada uno de los componentes del problema.

4.1 Java como lenguaje de desarrollo

La tecnología orientada a objetos provee al programador e investigador de una herramienta invaluable que abre la posibilidad de crear aplicaciones, de corte científico y tecnológico, que sean más confiables y de rápido desarrollo.

Por su parte, Java como lenguaje de desarrollo y como filosofía de programación, resultó ser todo un ambiente de desarrollo muy poderoso y versátil. Ya que fue posible implementar fácilmente las clases de manipulación de las imágenes derivando directamente de las clases **Applet** y **AWT** todas las características de manejo de los gráficos, su despliegue y métodos de actualización en pantalla, sin que haya existido la necesidad real de crear manejadores propios que realizaran dichas tareas. Lo cual permitió que el esfuerzo real y de mayor tiempo se centrara en la parte más fundamental del trabajo, que fue la comprensión e implementación de la metodología para el análisis de las imágenes satelitales y el posterior cálculo de las velocidades superficiales.

La capacidad de difusión de este tipo de sistemas cuyos resultados son necesarios en la toma de decisiones en la industria, no solamente se ve auxiliada por lenguajes y tecnologías como Java, sino que se soluciona completamente esta problemática al tener una audiencia potencial del 100% de las personas que tengan acceso a Internet y que cuenten con algún navegador capaz de ejecutar programas en Java. Resultando esto especialmente significativo si se toma en cuenta que los navegadores lideres para realizar estas tareas se distribuyen gratuitamente para prácticamente cualquier plataforma de hardware y sistemas operativos que funcionen en ellas.

Además, se debe tomar en cuenta una característica muy importante de la tecnología envuelta en Java, y es la capacidad de ejecutar programas completos o bien, la parte cliente de un sistema directamente en la computadora del usuario, aprovechando la potencia y eficiencia de las modernas computadoras personales. Marcando una gran diferencia: este tipo de sistemas que anteriormente podían ser ejecutados solamente en computadoras centrales o bien en equipo con características de estación de trabajo, ahora son susceptibles de ser distribuidos, empleados y ejecutados de principio a fin en computadoras personales incluso con capacidades modestas de cómputo. Siendo esto una gran posibilidad para la comunidad científica de desarrollar nuevas ideas

y proyectos de investigación que sean realmente utilizados por una mayor cantidad de personas en todo el mundo.

Dada su estructura completamente orientada a objetos y la amplia librería de clases con las que actualmente cuenta, facilita en gran medida la tarea del desarrollador al reutilizar sus clases especialmente las referidas a la manipulación de gráficos en su totalidad y sobre todo, a nivel pixel.

En un principio se pensó en desarrollar el trabajo en el lenguaje C++, pero éste representó una gran problemática en la manipulación binaria de archivos ya que no presenta una estructura clara para ello y sobre todo, no contiene clases especializadas en archivos gráficos por lo que el programador debe profundizar demasiado en la estructura binaria del archivo y en el lenguaje para obtener algún resultado. Lo cual hace que la manipulación de estos tipos de archivos resulte penosa y dilatada. Por su parte, Java cuenta con clases especializadas en el manejo de gráficos y su presentación logrando con ello que el desarrollo sea rápido y relativamente sencillo para el programador. Estas clases (Applet. AWT,. Graphics e Image) proveen de los métodos y objetos especializados para ser adoptados mediante la herencia en el programa y solucionar un problema complejo en cuestión de días. De hecho, la estructura principal de la clase **Procesa_imagen**, proviene de un desarrollador especializado en gráficos (Espeset, 1996) y pudo reutilizarse satisfactoriamente su trabajo en el presente ya que esta clase tiene una buen diseño orientado a objetos.

Sin embargo no todo es tan sencillo como parece. La metodología de evaluación y diseño arroja un esquema de clases y objetos hasta cierto punto ideal, pues aunque se cumpla con toda las condiciones de la teoría de objetos, en el momento de codificar este diseño en el lenguaje se encontraron serias trabas que implicaron cambiar el diseño final en la estructura del código fuente.

Como ejemplo particular de lo anterior cabe mencionar la metodología de graficación y el manejo de multihilos necesarios para la presentación animada de los resultados. Como puede verse claramente en el código de la clase **AVHRR_applet**, se implementó aquí el código necesario para la graficación (en este caso los métodos **paint** y **update**) aún y cuando esta clase es la principal. Esto fue necesario para aprovechar la implementación del manejo de multihilos que permite la interfase **Runnable**, la cual se encarga de que se puedan ejecutar recursivamente el cálculo de las velocidades y la graficación en el navegador en *tiempo real*. De no haberlo hecho así, se hubieran requerido de al menos dos clases extras: la primera de ellas debería de implementar el manejo de los hilos y sería la responsable del control y sincronización de la ejecución de esa parte del programa; y la segunda sería una subclase especializada exclusivamente en la presentación gráfica de los resultados. El hacer uso de estas clases nuevas hubiera significado seguir completamente con el diseño original de acuerdo con la teoría, pero hubiera acarreado una mayor complejidad en la codificación y se hubieran creado clases demasiado especializadas en el problema como para ser reutilizadas fácilmente en otro programa. Además de que representaría el no emplear las facilidades que ofrece el lenguaje en aras de simplificar la codificación y su empleo en múltiples plataformas. Sobre todo porque se buscaba que el usuario pueda ver los resultados gráficos y numéricos conforme estos se van realizando ya que se considero importante para su comprensión. Recordando que una característica que se deseaba implementar era el que el programa fuera lo más simple posible ya que deberían de evitarse retrasos en la ejecución por elementos "secundarios" dada la gran cantidad de cálculos matemáticos que deben de realizarse.

Debido a la formación de los investigadores de las ciencias de la tierra, éstos cuentan con una gran capacidad de abstracción que les permite descomponer un problema en sus partes elementales. Esto les facilita sobremanera el plasmar sus conceptos directamente en la metodología de objeto, pero resulta difícil para estos investigadores el siguiente paso, que es el codificar su

abstracción para obtener un programa de computadora. En este caso, Java representa una alternativa viable para que los ingenieros especialistas en informática desarrollen las clases y objetos adecuados que puedan ser utilizados y reutilizados fácil y eficazmente por los primeros.

4.2 La tecnología orientada a objetos y sus aplicaciones

La tecnología orientada a objetos significó además de una herramienta para el diseño del sistema, una manera distinta de concebir el problema y sus posibles soluciones.

El concepto del objeto significa una concepción distinta de la informática en las ciencias naturales al acercar el modelo computacional a una realidad constituida por entes independientes y colaboradores entre si que, vistos como un todo, representan la complejidad de un sistema que bien puede ser simulado y modelado a través de una computadora.

4.2.1 Comparación práctica entre el análisis y diseño en la programación por procedimientos y la orientación a objetos

Para ejemplificar la problemática envuelta en esta tarea se plantea lo siguiente: imagine un problema en el área de las ciencias de la tierra con cierta complejidad. Tal vez desee conocer como interactuan las diferentes comunidades en un ecosistema de playa y a la vez como influye el medio físico (p.e. el sustrato) en dicho ecosistema. Esta interacción debe ser modelada en el tiempo.

La cantidad de información involucrada requerirá en cierto momento el empleo de computadoras para realizar diferentes cálculos (p.e. estadísticos) que le permitan reconocer los procesos más influyentes en el sistema.

Y ¿cómo comenzar?. Analizando primero los elementos involucrados encontraremos un diccionario de actores y procesos bastante grande y variada que contendrá a especies competidoras y colaborantes, tipo de sustrato, nutrientes, energía en el medio, flujo energético entre comunidades... en fin, la lista crece y crece.

Ahora quizás decida separar el problema en partes y comenzar a solucionar aspectos específicos codificando cada segmento en la computadora. Para ello puede crear **diagramas de flujo** en donde se muestra de manera secuencial los **procedimientos** o procesos que modelen la interacción entre algunos elementos. Después creará más **diagramas de flujo** que detallen los subsistemas que usted haya identificado señalando los **procedimientos** de estos subsistemas y así sucesivamente hasta que tenga un esquema diagramático de todo su problema.

A partir de este momento, debe seleccionar el lenguaje en el cual va a codificar su diseño (tal vez el lenguaje C) y comience la programación. Durante toda la codificación del programa deberá tener mucho cuidado de mantener la integridad de todas las variables (como el número de individuos de una población o el tamaño de los sedimentos) para evitar que accidentalmente se cambien estos valores por algún procedimiento "escondido" en el programa. Conforme aumenten las líneas de código, los programadores deben manejar más información y adentrarse en detalles más finos aumentando la complejidad de la programación sumándose a la complejidad del problema.

Por fin, un día llega a buen término el programa y el sistema funciona perfectamente. Pero a raíz de la resolución del problema original, ahora se plantea una nueva problemática: incluir en la solución la injerencia humana en el ecosistema. Debido a la estructura por **procedimientos** de su

programa deberá revisar todo su diseño original y todo el código para encontrar la manera de incluir esta nueva característica en el sistema. Lo cual puede resultar frustante por todo el tiempo que deberá emplear solamente en determinar si su diseño original soporta esta inclusión, y en el peor de los escenarios, quizá deba recodificar todo o gran parte del sistema para esta nueva inclusión o bien si desea modificar un procedimiento de los ya existentes pudiera presentarse la situación en la que tenga que recodificar una buena parte del programa para que el sistema funcione bajo el nuevo esquema. Por lo que la limitante ahora se encuentra del lado de la programación y no en la problemática original la cual es la más importante *ya que la complejidad del problema se mantiene durante toda la implantación de la solución y en las subsecuentes revisiones del programa* sin contar que tal vez su sistema no le funcione para otros problemas que tengan elementos comunes con su problemática actual.

Entonces, si esta tratando de modelar un proceso del mundo real el cual se compone de entes funcionales ¿por qué no utilizar una metodología de programación que haga un mapeo del mundo real a la computadora?, ¿es posible pensar que estos entes sean vistos como objetos con características y comportamientos propios en lugar de procedimientos aislados y sin una entidad propia?.

De acuerdo con lo anterior, si se deja de pensar en **cómo** se realiza una acción y se piensa en **quién** realiza la acción se obtiene la posibilidad de modelar el mundo real tal y como lo analizamos comúnmente. En la playa no arriba un conjunto de partículas de agua con cierta densidad y con un movimiento definido, a la playa arriba una ola o la marea o brisa. Cada una de ellas con un comportamiento propio y características únicas y todas ellas son partículas de agua con cierta densidad y con cierto movimiento. Es muy distinto definir un ente solamente por sus datos y no por su comportamiento.

Retomando el ejemplo inicial, podemos definir a cada uno de los elementos del problema como **objetos** con características y comportamiento propios. Existen características comunes entre estos objetos que pueden definir **tipos de objetos** o **clases de objetos**, por lo que el esfuerzo se centraliza en la codificación de estas características comunes las cuales van a ser **heredadas** a **objetos** o **clases** que requieran una definición más detallada de sus características propias. Por ejemplo: el sustrato puede estar compuesto por rocas, estas a su vez pueden dividirse en rocas volcánicas, sedimentarias o metamórficas y estas a su vez se dividen en sedimentos mecánicos o químicos y así sucesivamente. Se parte de características generales hasta llegar al detalle fino del objeto que hereda características comunes. El agregar una nueva clasificación de sustrato será una tarea que solamente herede las características generales (si es el caso) y se agreguen únicamente las nuevas propiedades del objeto *sin que necesariamente se altere el resto de la codificación del sustrato*. Al hacer esto se **reutiliza el código** ya existente y puede ocurrir que el desarrollo de la solución sea más fácil de encontrar pues la complejidad del problema se esta granularizando en sus componentes elementales. Ya que se codificó el sustrato, se toma otro elemento como las comunidades bióticas en las cuales se definen características que pueden ser heredadas a comuninades bentónicas, planctónicas o pelágicas, y a la vez heredarán características a las comunidades bentónicas sésiles y así sucesivamente.

Al trabajar con una orientación a objetos y ver a cada componente como un ente, cuando se desee que algún objeto interactue con otro, solamente se deberán definir las **llamadas** a ejecución de los diferentes objetos, los cuales saben ya como comportarse de acuerdo al tipo de llamada recibida pues sus métodos y datos se encuentran **encapsulados** dentro de la definición del objeto.

Aunque este mecanismo quizá parezca más complicado que el manejo del flujo de información entre procedimientos, su entendimiento permite resolver la problemática original por partes, y una

vez realizado esto, enviar mensajes de entrada y salida de información a cada objeto conforme se vaya utilizando.

Para ejemplificar de una manera más clara las diferencias se presenta la siguiente figura en donde se muestra simplificado como se maneja mediante diagramas la complejidad del problema mediante el método por procedimientos y la orientación a objetos.

Una de las contribuciones de la orientación a objetos radica en la manipulación de la complejidad del problema al definir actores que realizan funciones propias que son "escondidas" de los modelos iniciales pero que se encuentran implícitas en cada uno de los objetos.

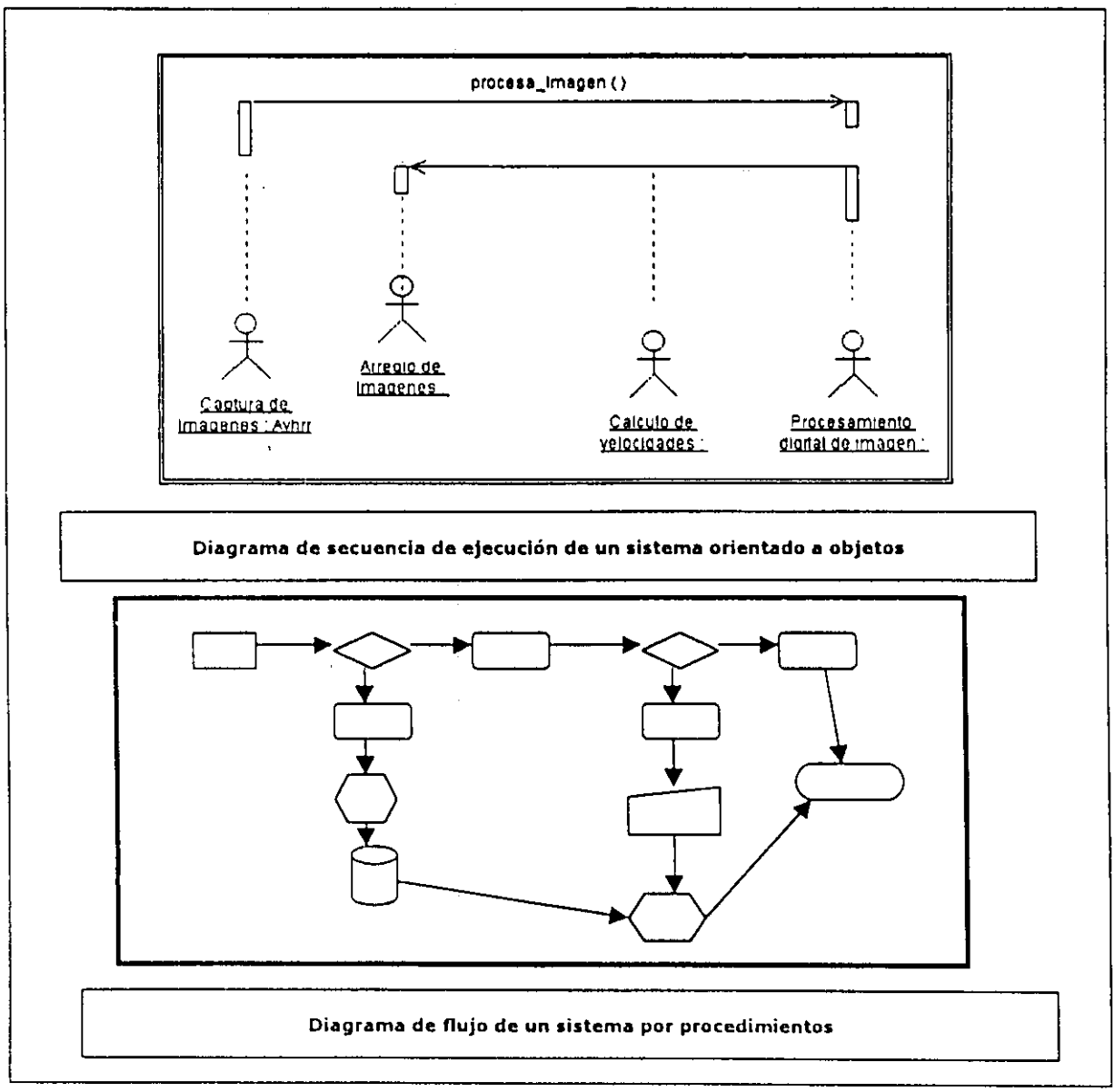

Figura 22. Comparación de los diagramas de secuencia de ejecución entre un sistema orientado a objetos y un sistema orientado a procedimientos. Mientras que en el primero los detalles finos están "escondidos" en cada uno de los objetos, en el segundo estos detalles se manejan todos a la vez dificultando la comprensión del problema. En el primer diagrama la complejidad del problema se observa fácilmente.

Al tener entes colaborantes en lugar de métodos ordenados secuencialmente, se gana en un manejo más cercano a la realidad del problema en cuestión, ya que en el diagrama de programación por procedimientos el programador necesita tomar en cuenta todas las salidas y entradas del sistema desde su concepción inicial (en el análisis) por lo que si el problema es muy complejo esta complejidad se hereda al programador. Ahora bien, si por algún motivo se desea hacer cambios en cualquier parte del programa, habrá que asegurarse de modificar también todas las dependencias entre los datos a partir del punto que se va a modificar hacia el final del programa.

Por su parte, en el diagrama de uso de objetos, se observa que cada ente es responsable de ejecutar ciertas tareas específicas. El programador debe tener en mente las macro tareas lo cual logra disminuir la complejidad del problema. Si por algún motivo se requiere modificar cualquier parte del programa, el programador deberá modificar solamente el objeto en cuestión sin preocuparse del resto del programa, pues existe el concepto de entes colaborantes que saben ya que hacer para cumplir con su tarea específica.

Resumiendo, la principal diferencia radica en que en el diagrama por procedimientos se utilizan solamente métodos que aparecen aislados, mientras que en el diagrama por objetos se emplean entes colaborantes a los cuales se les pide ejecutar tareas.

4.2.2 Estudios recientes relacionados con este trabajo

La comunidad científica de las ciencias naturales a continuado estudiando el movimiento de las capas superficiales de agua de mar mediante sensores remotos, y en especial, empleando imágenes SST para obtener el campo de velocidades superficiales del agua de mar.

Resulta importante denotar que a diferencia de los primeros trabajos en este campo, en donde se intentaba probar la factibilidad de emplear estas imágenes para mediciones reales y confiables, los nuevos estudios sostienen sus conclusiones basándose en la manipulación digital de las imágenes SST y generan ya nuevo conocimiento oceanográfico de sus resultados (Nishimura et.al., 1998).

Uno de estos trabajos fue desarrollado por Dominguez et. al. en 1998 en donde emplearon el ambiente de programación MATLAB para generar mapas de velocidades de corrientes superficiales mediante el método de correlación cruzada logrando resultados muy certeros en cuanto a la identificación de la dirección y magnitud de dichas corrientes, indicando la dificultad de encontrar resultados coherentes en las cercanías de las costas y las nubes y cuando las ventanas de muestreo contienen información concerniente a la tierra firme, cuando existe contaminación a nivel de subpixel por parte de las nubes y cuando en las ventanas de búsqueda no se encuentra la frontera entre patrones de colores de gran escala. Estos problemas de interpretación oceanográfica fueron resueltos en parte mediante la implementación de valores de umbral para los coeficientes de correlación, verificación de valores de pixeles invariantes (correspondiendo a la tierra), un valor de umbral para la velocidad máxima, aplicación de filtros espaciales, calculo del promedio de varios campos de velocidades y la determinación de funciones empíricas ortogonales complejas (método EOF por sus siglas en inglés). Y la ayuda de un operador que seleccionaba interactivamente las áreas dentro de las imágenes susceptibles para aplicarles el método.

Esta aproximación es muy interesante pues identifica varias estrategias que pueden implementarse para mejorar y automatizar la aplicación de este tipo de estudios, entre ellas se identifican:

- Un análisis a los datos de la imagen empleando redes neuronales como seleccionadores de patrones oceanográficos que las identifiquen y las cataloguen para la aplicación de un método en específico, como puede ser el caso de patrones circulares (eddys) para los cuales deben emplearse metodologías distintas debido a su naturaleza no lineal.

- Un análisis estadístico, por ejemplo el EOF o la descomposición de eigenvectores aplicados en el eje X y Y (Huang y Robinson, 1995) para identificar las componentes principales en las variaciones espaciales en cada imagen y entre las imágenes además de funcionar como una medida de la varianza entre los datos.

- La aplicación de filtros espaciales y de gradientes para identificar las zonas no computables y eliminar ruido en las imágenes.

- Un estudio de los valores de umbral adecuados para cada área de estudio con el propósito de que se adapten estos valores dependiendo de la complejidad de la distribución de los patrones térmicos.

Por su parte, Labuto y Valentini (1998) realizaron una compilación de lo que se ha realizado en los últimos cinco años en materia del empleo de imágenes SST en la oceanografía separando claramente cada una de las aplicaciones en las que se han empleado en diferentes estudios en todo el mundo. En este trabajo indican que actualmente la discusión de este tipo de imágenes se ha centrado en problemas de calibración y no en el potencial de nuevos estudios en los cuales estas imágenes pueden apoyar e influir considerablemente en los resultados y conclusiones obtenidas.

De la serie de trabajos revisados, se pudo observar con satisfacción que la informática va integrándose como un componente esencial en el estudio de las ciencias de la tierra, presentando trabajos que emplean manipulación digital de las imágenes para extraer patrones e información estadística que apoya directamente a la interpretación oceanográfica. Un comentario interesante dentro del trabajo radica en que la Oceanografía y los sensores remotos ya están íntimamente ligados en el trabajo de los investigadores y que deben trabajar juntos en la investigación con el fin de lograr conclusiones más extensas. Tal vez, a este pensamiento habrá que agregar un tercer actor que hasta el momento se ha mantenido "tras bambalinas" pero que es indispensable: la informática y computación como aliados necesarios y de primera mano en la investigación científica y tecnológica del mar.

Resulta inobjetable la participación de los profesionales de la computación e informática en este tipo de investigaciones, y en beneficio de todas las áreas del conocimiento humano involucradas estos profesionales deben formar un equipo interdisciplinario e integral junto con los oceanógrafos, geográfos y demás profesionales involucrados para que juntos identifiquen el problema, lo descompongan en sus componentes principales y elaboren el proyecto conjunto que involucre la sapiencia de cada uno con un fin común y único reflejado en los objetivos del estudio. De esta manera la informática y computación se verían beneficiadas con las experiencia de dichas áreas y puedan generar nuevas líneas de investigación derivadas de las necesidades combinadas e individuales del grupo.

4.2.3 Conclusiones

Las imágenes obtenidas a partir de sensores remotos constituyen una herramienta invaluable en el estudio de las ciencias de la tierra al proporcionar información práctica, económica y de fácil acceso tanto a los investigadores como a los usuarios finales quienes se benefician directamente de ella. Sin embargo, estas imágenes contienen información extra que puede ser extraída mediante su manipulación digital y procesada en tiempo real por prácticamente cualquier computadora moderna de una manera eficiente.

En este sentido, el lenguaje de programación Java constituye una plataforma de desarrollo viable y accesible para el programador para crear soluciones sofisticadas y acertadas pues se constituye como un lenguaje maduro y potente.

Los campos de velocidades obtenidos representaron adecuadamente a un nivel cualitativo los cambios en las posiciones de las áreas de colores en las imágenes analizadas siempre y cuando los cambios en dichos mapas de colores no sean demasiado drásticos. La sensibilidad de los resultados depende de una variedad de factores como son: la diferencia temporal entre las imágenes, la hora del día en la que se obtuvo, la cobertura nubosa, variaciones en la asignación del mapa de colores empleado en el falso color, eventos "instantáneos" o de muy corta duración y los errores de redondeo y acarreo producidos por la gran cantidad de operaciones matemáticas que se deben realizar en el método seleccionado.

Por su parte la tecnología orientada a objetos provee al investigador informático y al de las ciencias de la tierra de una manera eficiente y real de pensar y analizar la complejidad del problema en turno pues crea la gran expectativa de obtener soluciones integrales a problemas sumamente complejos mediante una abstracción que refleja en un programa para computadora la estructura del mundo real, logrando ampliar el potencial del alcance del proyecto de investigación a un plano más amplio e integral y por lo tanto a conclusiones más completas y específicas.

Cualquier sistema natural es infinitamente más complejo que cualquier sistema humano, por ello es necesario que el investigador de las ciencias de la tierra cuente con la metodología necesaria que le permitan comprender y abstraer los elementos fundamentales de dicho sistema y poder crear modelos que sean representativos de dicha complejidad y no solamente una abstracción parcial del problema.

Indice

1 INTRODUCCIÓN 1
1.1 OBJETIVOS 2
2 ANTECEDENTES 3
2.1 ANTECEDENTES GENERALES 3
2.2 ANTECEDENTES PARTICULARES 4
3 METODOLOGÍA 7
3.1 PARA EL CÁLCULO DEL CAMPO DE VELOCIDADES SUPERFICIALES 7
3.1.1 Obtención de las imágenes 7
3.1.2 Metodología del cálculo de velocidades superficiales 9
3.2 PARA EL ANÁLISIS Y DISEÑO DEL PROGRAMA EMPLEANDO LA ORIENTACIÓN A OBJETOS 14
3.2.1 La metodología orientada a objetos empleada en el análisis y diseño del programa . 14
3.2.2 El lenguaje de programación y sus requerimientos 14
RESULTADOS 16
3.2.3 Análisis y diseño 16
3.2.3.1 Los primeros pasos: definición del problema 16
3.2.4 Análisis y diseño del problema 18
3.2.4.1 Análisis y diseño de la aplicación 18
3.2.4.1.1 Diagrama de uso: principal 19
3.2.4.1.2 Diagrama de uso: Captura de la imagen 20
3.2.4.1.3 Diagrama de uso: Arreglo digital 21
Diagrama de uso: Cálculo de velocidades 22
3.2.4.1.5 Diagrama de uso: Presentación de resultados 23
3.2.4.1.6 Diagrama de secuencia: Selección de fecha y hora 24
3.2.4.1.7 Diagrama de secuencia: Captura y proceso 25
Diagrama de clases inicial 26
Diagrama de clases final 27
3.3 PROGRAMACIÓN 28
3.3.1 CLASE AVHRR_applet.java 28
3.3.2 CLASE Procesa_imagen.java 28
3.3.3 CLASE DebugWin.java 36
3.4 CÁLCULO DEL CAMPO DE VELOCIDAD SUPERFICIAL DEL AGUA DE MAR 37
3.4.1 Obtención de las imágenes 37
3.4.2 Interfase para el usuario 39
3.4.3 Resultados de la ejecución del programa 40
3.4.4 Desempeño del programa 48
4 DISCUSIONES 50
4.1 JAVA COMO LENGUAJE DE DESARROLLO 50
4.2 LA TECNOLOGÍA ORIENTADA A OBJETOS Y SUS APLICACIONES 52
4.2.1 Comparación práctica entre el análisis y diseño en la programación por procedimientos y la orientación a objetos 52
4.2.2 Estudios recientes relacionados con este trabajo 55
4.2.3 Conclusiones 57
5 REFERENCIAS 58

Indice de tablas

Tabla 1. Características de las bandas de recepción de la región espectral del sensor AVHRR-NOAA. Tomado de Chuvieco, 1990.....3

Tabla 2. Relación de archivos, fechas, horas y bandas de muestreo de las imágenes AVHRR seleccionadas para este estudio. Todas ellas corresponden a una región del pacífico occidental de la República Mexicana de aproximadamente 200 kilómetros por lado frente a las costas de Lázaro Cárdenas, Michoacán. Entre el 11 de diciembre de 1996 y el 24 de enero de 1997....38