28.

Trabajo de Grado

Métricas para Análisis Textural de Imágenes

Alumnos

Raimondi, Gustavo

Mulinaris, Pablo

Champredonde, Jorge

Director

Agr. Platzeck, Gabriel

Co-Director

Ing. De Giusti, Armando

Departamento de Informática - Facultad de Ciencias Exactas

Universidad Nacional de La Plata

1997

Indice

Capítulo 3: Herramientas Utilizadas

3.1 - El Lenguaje de Programación C++
- **3.1.1** - Reseña Histórica
- **3.1.2** - Algunas Restricciones
- **3.1.3** - Límites de C++
- **3.1.4** - Soporte de la Programación Orientada a Objetos
  - **3.1.4.1** - Clases
  - **3.1.4.2** - Objetos
  - **3.1.4.3** - Encapsulamiento
  - **3.1.4.4** - Herencia
  - **3.1.4.5** - Herencia Múltiple
  - **3.1.4.6** - Polimorfismo en C++
  - **3.1.4.7** - Mecanismos de Llamado
  - **3.1.4.8** - Verificación de Tipos
- **3.1.5** - Model / View / Controller

3.2 - Khoros
- **3.2.1** - Introducción
- **3.2.2** - Servicios de Programación de Khoros
  - **3.2.2.1** - Servicios Fundamentales
  - **3.2.2.2** - Servicios de Datos
  - **3.2.2.3** - Servicios GUI y Visualización
- **3.2.3** - Herramientas de Programación
  - **3.2.3.1** - El Sistema de Programación Visual
  - **3.2.3.2** - El lenguaje Visual Cantata
  - **3.2.3.3** - La Herramienta de Diseño GUI de Cantata
  - **3.2.3.4** - Herramientas de Desarrollo de Software

3.3. - Evaluación Comparativa de las Herramientas de Desarrollo
- **3.3.1** - Experiencia de Desarrollo en C++
- **3.3.2** - Experiencia de Desarrollo en el Ambiente Khoros
- **3.3.3** - Conclusiones de la Comparación entre las Experiencias de Desarrollo

Capítulo 4: Métricas: Estudio y Utilización

4.1 - Evaluación
4.2 - Suavizado HS
4.3 - Clasificación

Capítulo 5: Apreciación Final y Líneas de Trabajo Futuras

5.1 - Conclusiones
5.2 - Aplicaciones y Líneas de Trabajo Futuras

Anexo A: Bibliografía

Agradecimientos

A los integrantes del LIDI que nos facilitaron todo el material disponible, conocimientos y equipos para el desarrollo.

A LBS Informática por habernos facilitado su tiempo y sus equipos.

Al Turco por su paciencia inagotable.

A la DAIS por el material y conocimientos facilitados.

A Pablo.

A Las Balinas, por su colaboración en Khoros.

A cada uno de los que, de una u otra manera, participaron en este trabajo.

Dedicatorias

A nuestros padres, familiares y amigos, que son los que dieron contenido a este *trabajo*.

A mi Señora y a mi Hijo. Colo.

A Paula. Mula.

A mi Perrita. Jorge.

A Los Malditos.

CAPÍTULO 1

1 - Introducción

1.1 - Definición del problema

Las tareas intelectuales humanas incluyen realización, evaluación e interpretación de impresiones sensoriales, lo que podríamos llamar percepción. A pesar de que todo ser humano es capaz de percibir una gran cantidad de impresiones sensoriales, no resulta sencillo establecer en forma precisa cuál es el mecanismo que realiza tal percepción.

La idea central en el reconocimiento de patrones es la simulación de la percepción humana, lo que no implica reconstruir en forma exacta los algoritmos utilizados, sino que simplemente se intenta alcanzar resultados similares.[5].

Una caracterización convencional en el procesamiento de imágenes, es realizada en base a la firma espectral de los objetos, la cual proporciona información sobre las capacidades de reflexión versus absorción de los objetos, en función de la longitud de onda, lo cual se traduce en un vector de amplitudes. En determinadas ocasiones este tipo de características son insuficientes, es por eso que se hace necesario investigar nuevos parámetros para mejorar la capacidad de discriminación.

Una posibilidad es extraer características que involucren propiedades de superficies o áreas.

La textura es uno de los elementos utilizados en la interpretación visual de imágenes de sensores remotos y ha sido utilizada experimentalmente en análisis automatizado de imágenes. En particular el reconocimiento de patrones espaciales en base a textura, es realizado con cierta facilidad por el experto, pero es difícil de automatizar sobre una computadora.

En la actualidad los métodos estadísticos de clasificación automatizada aplicados a imágenes multiespectrarles, basados en las firmas espectrales o amplitudes, pueden mejorar su capacidad de discriminación cuando se incorpora información textural.

1.2 - Motivación y objetivos

Este trabajo tuvo lugar en el marco de un proyecto de colaboración entre el LIDI (Laboratorio de Investigación y Desarrollo en Informática) del Departamento de Informática preteneciente a la Facultad de Ciencias Exactas de la UNLP y la Dirección de Análisis de Imágenes Satelitarias (DAIS) dependiente del Ministerio de Obras y Servicios Públicos de la Provincia de Buenos Aires. Contando con la colaboración de los expertos de la DAIS, se intentó complementar las capacidades de las herramientas para fotointerpretación asistida utilizadas por dicha Dirección, concluyendo que sería de suma utilidad la incorporación de técnicas de Análisis Texural de Imágenes.

De esta manera, nos propusimos evaluar la capacidad de discriminación de texturas que proporcionan las características espaciales de los niveles de gris en una imagen.

Luego intentamos desarrollar una aplicación que incorpore estas técnicas para la fotointerpretación asistida, la que permitiría al usuario experto identificar visualmente una textura determinada y señalarla, a partir de lo cual sería posible localizar tal patrón de textura en la imagen completa.

Las aplicaciones concretas de este tipo de herramienta son numerosas tanto desde el punto de vista teórico como aplicado. En este último aspecto se destacan la optimización de métodos para la estimación de cosechas, crecimiento de zonas desérticas y la delimitación de áreas afines en imágenes de alta resolución de zona urbana.

CAPÍTULO 2

2 - Procesamiento de imágenes

2.1 - Introducción

La frase *procesamiento digital de imágenes* se refiere frecuentemente al procesamiento de figuras bidimensionales mediante una computadora digital. Generalizando esta idea, podríamos decir que abarca al procesamiento de cualquier conjunto de datos n-dimensional. Una imagen digital estará representada por un arreglo de números reales o complejos, cada uno de ellos representado a su vez por un número finito de bits[1].

El procesamiento digital de imágenes abarca un amplio espectro de aplicaciones tales como remote sencing mediante satélites, transmisión y almacenamiento de imágenes para aplicaciones comerciales, procesamiento de imágenes médicas, de radar, acústicas, de robots, e inspección automatizada en la industria, entre otras.

Una posible clasificación de los problemas a resolver durante el procesamiento de imágenes sería:

- Modelización y representación de imágenes
- Mejorado de imágenes
- Restauración de imágenes
- Análisis de imágenes
- Compresión de imágenes

2.1.1 - Modelización y representación de imágenes

Durante la representación de las imágenes se determinará qué cantidad representará cada pixel (*picture-element*). Una imagen podría representar la luminancia (luminances) de los objetos en una escena (cámara común), las características de absorción del tejido de un cuerpo (rayos X), la respuesta a las ondas de radar (radar cross section) de un objetivo o bien el perfil de temperatura de una región (infrarrojos), entre otros. En general cualquier función n-dimensional que contenga información, puede ser pensada como una imagen.

Una consideración importante en la representación de imágenes es la determinación de un criterio de fidelidad, con el fin de medir la calidad de una imagen. El conocimiento de un criterio de fidelidad será útil en el diseño de sensores, ya que permitirá determinar qué variables deberían ser medidas con mayor o menor precisión.

El requerimiento fundamental en el procesamiento digital es que las imágenes deben ser muestreadas y cuantizadas. El número de pixels por unidad de área, es decir, la frecuencia de muestreo a utilizar, debe ser suficiente para evitar la pérdida de información útil de una imagen.

Un método clásico para representación de señales es la utilización de una expansión de series ortogonales, como son las series de Fourier. Para el caso particular de imágenes, una representación análoga es posible mediante funciones ortogonales bidimensionales, llamadas *basis images*. Cualquier imagen podrá ser entonces representada mediante sumas pesadas de éstas. Diversas características de las imágenes, tales como su contenido en frecuencia espacial, ancho de banda, etc. y sus aplicaciones en el diseño de filtros y extracción de características, pueden ser estudiados mediante tales expansiones.

Los modelos estadísticos describen a una imagen como miembros de un ensamble, generalmente caracterizado por sus funciones de media y covarianza. Esto permite el desarrollo de algoritmos útiles para una clase completa o un ensamble de imágenes. Con frecuencia, dicho ensamble se supone estacionario, de manera que las funciones de media y covarianza pueden ser fácilmente determinadas. Los modelos estacionarios son útiles en problemas de compresión, restauración y otras aplicaciones donde las propiedades globales del ensamble sean suficientes.

Con el fin de caracterizar propiedades locales de los pixels, una alternativa es la caracterización de cada pixel, mediante una relación con sus pixels vecinos. Estos métodos son útiles para el desarrollo de algoritmos que tengan diferentes implementaciones en hardware.

2.1.2 - Mejorado de imágenes

El objetivo aquí es acentuar determinadas características de la imagen para que sean más adecuadas para subsecuentes análisis, o simplemente para su posterior visualización. Algunos ejemplos podrían ser mejoras de bordes y contraste, pseudocoloreado, filtrado de ruido, realce (sharpening), entre otros.

La mejora de imágenes será de utilidad en tareas tales como extracción de características, análisis de imágenes y visualización de información.

El mejorado de imágenes por sí mismo no incrementa el contenido de información de los datos, sino que simplemente enfatiza ciertas características. Los algoritmos de este tipo suelen presentarse como interactivos y dependientes de la aplicación.

Algunas técnicas para mejorado de la imagen mapean un nivel de gris en otro nivel de gris o color, mediante una transformación predeterminada. Otras técnicas realizan operaciones sobre vecindades locales como en el caso de convolución, o bien operaciones de transformación como la transformada discreta de Fourier.

2.1.3 - Restauración de imágenes

La restauración de imágenes consiste en minimizar o bien remover las posibles degradaciones que una imagen puede sufrir. Estas técnicas incluyen la eliminación del borroneado de imágenes degradadas por limitaciones en los sensores utilizados o su medio ambiente, filtrado de ruido y corrección de distorsiones geométricas o no lineales, causadas por los sensores.

Un resultado fundamental en la teoría de filtrado utilizado frecuentemente en restauración es el llamado *filtro de Wiener*, el cual es la mejor estimación cuadrática media lineal de los objetos en observación.

Existen otros métodos de restauración tales como mínimos cuadrados y mínimos cuadrados restringidos, que pueden ser clasificados como filtros de Wiener. Otras técnicas tales como máxima verosimilitud, máxima entropía y máximo a posteriori son técnicas no lineales de restauración que requieren soluciones iterativas.

2.1.4 - Análisis de imágenes

El análisis de imágenes involucra la toma de medidas cuantitativas de una imagen con el fin de producir una segmentación o una descripción de la misma.

En su forma más simple, esta tarea podría involucrar el ordenamiento de diferentes partes en una línea de ensamble, o la determinación de tamaño y orientación de células de sangre en imágenes médicas. En sistemas de análisis de imágenes más avanzados, se mide información cuantitativa, la cual es utilizada para tomar decisiones sofisticadas, tales como el control de los brazos de un robo con el fin de mover un objeto luego de identificarlo, o navegar un avión con la ayuda de imágenes adquiridas a través de su trayectoria.

Las técnicas aquí requieren la extracción de ciertas características que asistan en la identificación de los objetos. Se utilizan entonces técnicas de segmentación, con el fin de aislar los objetos deseados de la escena de manera de obtener luego las medidas de interés sobre éstos. Las medidas cuantitativas de las características de los objetos permitirán finalmente la clasificación y/o descripción de la imagen.

2.1.5 - Compresión de imágenes

La cantidad de datos asociados a la información visual suele ser excesivamente grande, y por tanto su almacenamiento requiere enormes capacidades. A su vez el tiempo de acceso a la información es directamente proporcional a la capacidad de almacenamiento requerida. A modo de ejemplo, las imágenes televisivas generan datos a razón de más de 10 millones de bytes por segundo.

Asimismo, el almacenamiento y/o transmisión de tales datos requiere grandes capacidades y/o ancho de banda, lo cual podría resultar muy costoso. Las técnicas de compresión de datos de imágenes intentan reducir el número de bits requeridos para almacenar o transmitir imágenes evitando una pérdida substancial de información.

Debido a la gran cantidad de aplicaciones que involucra, la compresión de datos es de gran importancia en el procesamiento digital de imágenes.

2.2 - Análisis de Imágenes

El análisis de imágenes se relaciona fuertemente con la extracción de información o datos a partir de imágenes mediante sistemas o dispositivos automáticos o semiautomáticos. [2][3]

El análisis de imágenes se distingue de otros tipos de procesamiento de imágenes, tales como codificación, restauración y mejorado, en que usualmente el producto final de un sistema de análisis de imágenes será una salida numérica, en lugar de otra imagen.[4]

El objetivo en el análisis de imágenes es relacionar objetos de una escena, con niveles de gris en la imagen. Este trabajo generalmente involucra el *matching* de información derivada de la imagen con modelos de objetos con determinadas restricciones conocidas a priori. [3]

Cuando el análisis de las imágenes se restringe a la clasificación de diversas subregiones en un número fijo de categorías, estamos en presencia de un sistema clásico de reconocimiento o análisis de patrones.[4][5]

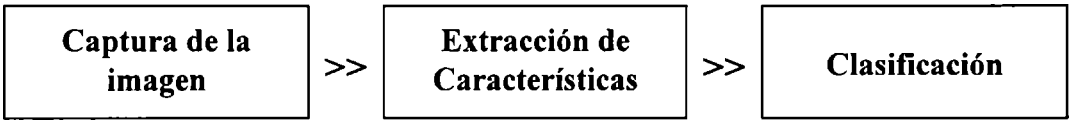
El reconocimiento de patrones es la ciencia que trata la descripción o clasificación de medidas, usualmente basado en un modelo. Las dos aproximaciones principales al reconocimiento de patrones son la estadística y la sintáctica o estructural, a las que podemos agregar una tercera surgida posteriormente, la cual se basa en la utilización de redes neuronales. [2]

Aquí aparecen dos tópicos de interés.

- Para el procesamiento de imágenes, será de utilidad reducir una aproximación basada en estadística para clasificación de características o basada en patrones, a una formulación práctica.
- La selección de características o primitivas en problemas de reconocimiento de patrones generales, se acerca más al arte que a la ciencia.

La estructura de un sistema de reconocimiento de patrones típico, consta de:

un sensor, un mecanismo para preprocesamiento y/o mejorado, un mecanismo de extracción de características, y finalmente, un algoritmo de clasificación o de descripción, dependiendo de la aproximación utilizada. [5][2]



A su vez, en muchos casos se asume que algunos datos ya clasificados o descriptos, se encuentran disponibles con el fin de entrenar al sistema.

2.2.1 - Características (Features)

A modo de definición podemos pensar a las características como cualquier medida que sea posible extraer y a su vez pueda prestar alguna utilidad en procesamientos posteriores. Algunos ejemplos de características son la intensidad de los pixels o distancias geométricas entre pixels.

Las características pueden también resultar de la aplicación de un algoritmo de extracción de características u operador, a los datos de la imagen.

Podemos destacar aquí que un esfuerzo computacional significativo puede ser requerido durante la extracción de características de la imagen.

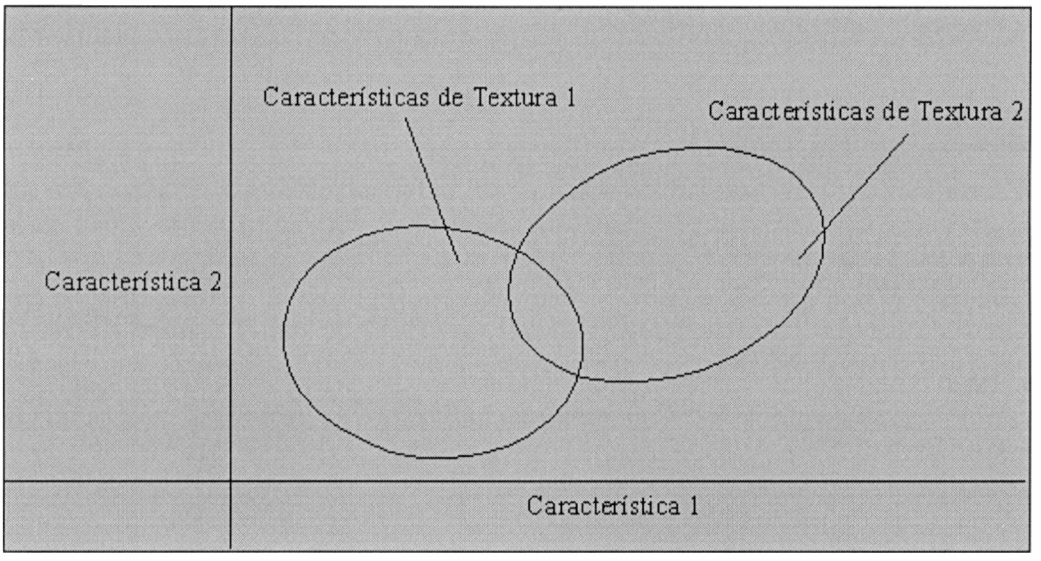
Las características pueden provenir de diversos dominios de información, que detallamos:

- *El dominio espectral:* Tomemos a modo de ejemplo los espectros de luz visible e infrarrojo, los cuales pueden ser utilizados para obtener características de un objeto, independientes entre ellas. Esta independencia en las características se debe al origen independiente de las energías tomadas en cuenta, dado que la primera se basa en la apariencia visible de los objetos, mientras que la última es una función de su temperatura.
- *El dominio espacial:* Un ejemplo aquí sería la distribución espacial de un patrón de intensidades sobre una región, la cual puede ser utilizada como una característica para distinguir regiones particulares.
- *El dominio temporal:* Por ejemplo, el cambio o movimiento de atributos de la imagen es una característica dinámica que varía en función del tiempo, lo cual para el caso de sensores estacionarios, provee un método de discriminación entre entidades que se mueven en la escena y aquellas que permanecen estacionarias.

Generalmente, las características son representadas mediante un vector de características, el cual conduce a un espacio de medidas multidimensional. La gran dimensionalidad es un rasgo distintivo en la extracción de características y durante la utilización de reconocimiento de patrones en las aplicaciones de procesamiento de imágenes.

2.2.2 - Reconocimiento estadístico de patrones. Teoría elemental

Supongamos que debemos distinguir entre dos clases de objetos w_1 y w_2, cada una de las cuales posee dos características distintivas x_1 y x_2. A modo de ejemplo podemos considerar a las características distribuidas de acuerdo a la siguiente *figura*. [3]



De acuerdo a esto, si tomamos un elemento perteneciente a la clase w_1, esperamos valores pequeños para x_1 y x_2, mientras que si el objeto en observación pertenece a la clase w_2 , esperaremos valores más grandes para x_1 y x_2.

Posee una importancia particular la región donde los valores de las características se superponen, ya que en ellas son probables los errores en la clasificación.

Si tomamos una muestra particular, las características extraídas serán colocadas en un vector de características $x^T = [\ x_1\ ,\ x_2\ ]$. Supongamos disponible la función de densidad de probabilidad condicional para el vector de características $p(x_1\ ,\ x_2 \mid w_i\ )$ o simplemente $p(\underline{x}|w_i)$. Asumamos finalmente que se dispone de alguna información anterior a la medición de las características de los objetos, de la probabilidad de ocurrencia de las clases w_1 y w_2 , es decir que se conocen las probabilidades a priori $P(w_i\ )$, i = 1,2. En ausencia de dicha información, una suposición razonable es tomar $P(w_1\ ) = P(w_2\ )$.

El problema de clasificación mediante reconocimiento estadístico de patrones puede ahora ser descripto brevemente como la determinación de una estrategia para clasificar diversas muestras, en base a las mediciones de $\underline{x}$, de manera de minimizar los errores durante la clasificación.

Utilizando el teorema de Bayes, la estimación de la probabilidad a priori de una clase se convierte en la probabilidad a posteriori de la siguiente forma:

$$p(w_i \mid \underline{x}\ ) = p(\underline{x} \mid w_i\ ) \,.\, P(w_i\ ) \,/\, p(\underline{x}\ )$$

donde

$$p(\underline{x}\ ) = \sum_i p(\underline{x} \mid w_i\ )$$

Aunque se omiten algunos detalles aquí, la forma de minimizar el error de clasificación requiere que, dada una muestra con vector de características $\underline{x}$, se elija la clase w_i para la cual $P(w_i\ )$ sea máxima, lo cual es también intuitivamente razonable.

Podemos notar aquí que $p(\underline{x})$, por ser común a todas las probabilidades condicionales de las distintas clases, resulta en un factor de escala que puede ser eliminado.

De esta manera, el método de clasificación estará dado por decidir

- w_1 cuando $p(\underline{x} \mid w_1) \cdot P(w_1) > p(\underline{x} \mid w_2) \cdot P(w_2)$
- w_2 cuando $p(\underline{x} \mid w_2) \cdot P(w_2) > p(\underline{x} \mid w_1) \cdot P(w_1)$

Puede notarse también que es posible utilizar cualquier función monótona no-decreciente de $P(w_i \mid \underline{x})$ para este método.

2.2.3 - Aplicaciones de reconocimiento estadístico de patrones utilizando imágenes satelitales.

Los sensores basados en satélites en órbita alrededor de la tierra son una de las principales fuentes de datos cuantitativos de imágenes. Las imágenes adquiridas de la superficie terrestre, las cuales a su vez pueden estar distribuidas en diversos espectros, contienen información que suele ser de utilidad en aplicaciones militares, económicas y humanitarias, entre otras. A modo de ejemplo podemos mencionar que es posible identificar áreas de la superficie terrestre que contengan depósitos minerales, o bien la clasificación de áreas en base a su vegetación.

Una serie de satélites utilizados para sensado remoto de la Tierra es el programa Landsat, cuyos satélites están equipados por sistemas de scanner multiespectrales, los cuales generan un vector de características para cada pixel, cuya i-ésima componente representa la respuesta en intensidad en un espectro en particular de una posición dada.

Consideremos como ejemplo la siguiente tabla:

Característica	Longitud de onda	Banda
x_1	0.5 - 0.6 μm	visible
x_2	0.6 - 0.7 μm	visible
x_3	0.7 - 0.8 μm	infrarrojo
x_4	0.8 - 1.1 μm	infrarrojo

2.2.4 - Reconocimiento / Segmentación

El resultado de la segmentación estará dado por un conjunto de *objetos de segmentación*, donde cada uno de ellos contará con determinados atributos.

A su vez, para que sea posible llevar a cabo esta etapa, se requiere de algunos cambios o fluctuaciones útiles de alguna de las propiedades de los patrones. [5]

El proceso de segmentación agrupa los pixels de una imagen de manera de obtener estructuras de imágenes que contengan regiones, las cuales a su vez pueden estar relacionadas con entidades en un modelo de más alto nivel. El objetivo último de la segmentación es particionar la imagen con el fin de determinar qué representa cada región en la escena real en tres dimensiones. El éxito del algoritmo de segmentación generalmente determina el éxito o fracaso del sistema de análisis de imágenes completo. El problema general de segmentación puede ser considerado en el contexto de un problema de reconocimiento de patrones, donde las entidades a ser clasificadas pueden ser tanto pixels como regiones [3]. Generalmente, las aproximaciones de *clustering* o clasificación no supervisada son utilizadas bajo determinadas condiciones que se detallan a continuación.

- Cada región resultante es tan homogénea como sea posible respecto de alguna medida de similitud de características.
- Los pixels que se encuentran en diferentes regiones son no-homogéneos.
- El agrupamiento resultante mantiene algún significado en función del procesamiento posterior; es decir que no son agrupamientos artificiales sino que, por el contrario, constituyen subregiones de la imagen cuyas interpretaciones son significativas respecto de un concepto de más alto orden o nivel de abstracción, como pueden ser calle, árboles, agua, pasto, etc.

Existen al menos dos formas de realizar segmentación. En la segmentación *no-contextual*, las relaciones entre características de pixels o regiones, son ignoradas, mientras que en la *contextual* se utilizan relaciones entre cacterísticas en las vecindades. Esta última suele ser más exitosa que la primera, ya que la información de lugar en una imagen puede ayudar en la decisión de clasificación, pero desde el punto de vista teórico es dificultoso para cuantificar, como así también resulta compleja su implementación.

Un buen ejemplo de la aproximación no contextual es la clasificación de pixels mediante reconocimiento estadístico de patrones, donde el vector de características se basa en la intensidad. Una posible aplicación de esta metodología es la utilización de thresholding simple sobre el valor de un pixel con el fin de asignarlo a una de las clases ON u OFF. En contraste, la detección de bordes y el crecimiento de regiones, ejemplificarían una aproximación de clasificación contextual, donde la base para las decisiones en la segmentación, proviene de datos tomados sobre la región de un pixel y no de un pixel individual.

2.2.4.1 - Segmentación basada en bordes

De acuerdo a lo mencionado anteriormente, todos los algoritmos de segmentación de bajo nivel, se basan en el concepto de similitud (o bien, disimilitud) de atributos. Una posible aproximación a la detección de disimilitudes es intentar identificar discontinuidades en la intensidad de los niveles de gris (bordes).[3]

En muchos casos, los bordes proveen información importante acerca de los objetos o de la escena. Existen numerosos métodos para extracción de contornos, ya que la reducción de imágenes de niveles de gris a contornos, puede resultar en una

significativa reducción de información y, a su vez, podría facilitar posteriores procesamientos. [5]

Debido a la presencia de ruido superpuesto en las imágenes, no será posible la detección de contornos ideales y como consecuencia, la extracción de contornos se realizará en al menos dos etapas: en la primera de ellas se determinarán los puntos o segmentos de línea donde ocurran cambios en los niveles de gris, y en segundo lugar, se intentará encadenarlos mediante líneas rectas o curvas.

Entre los métodos utilizados en la primer etapa podemos mencionar: Operadores gradiente, tales como Roberts, Prewitt, Sobel, Isotropic, Laplaciano, Zero-crossing, operadores de compás y detección de líneas y spots, entre otros.

Entre los métodos utilizados en la segunda etapa se encuentran: Conectividad 4 u 8, Seguimiento de contornos, Encadenamiento de bordes, búsqueda en grafos y la transformada de Hough.

El problema de segmentación basada en bordes se torna más complejo si se requiere que una imagen sea segmentada a partir de determinar regiones con bordes que formen contornos cerrados en la imagen plana. [3]

A su vez, la representación de los bordes es importante en las aplicaciones donde se realice análisis y síntesis de formas. Entre los métodos desarrollados para resolver esta etapa, podemos contar a los Códigos de Cadena, Ajuste de segmentos de línea y representación B-Spline entre otros.

2.2.4.2 - Segmentación basada en regiones

La aproximación *regional* intenta realizar la segmentación de la imagen, considerando la similitud entre los datos de una imagen, como alternativa al mejorado y encadenado de bordes en la imagen.[3]

La extracción de regiones aparece debido a que la detección de contornos es muy dificultosa en presencia de ruido, y que además existen casos donde los bordes solos no contiene información suficiente. El rasgo fundamental de las regiones es la *homogeneidad* respecto de propiedades o parámetros elegidos en forma adecuada.[5]

Algunas condiciones en los resultados de la segmentación estarán dados por:

- Cada uno de los pixels de una imagen deben pertenecer a una y sólo una región.
- Cada una de las regiones debe ser homogénea respecto del criterio adoptado.
- Si intentáramos mezclar dos regiones que compartan un límite en común, incurriríamos en una violación al criterio de homogeneidad adoptado.

De aquí en más consideraremos que se aplica un operador de segmentación a una subregión de la imagen relativamente pequeña lo cual implica que:

- Desde el punto de vista teórico, desearemos una resolución tal, que la imagen segmentada en base a las características extraídas, nos permita determinar de manera confiable el tamaño de una región, su forma y sus límites. Esto sugiere que los resultados que se obtengan variarán en función del tamaño de las regiones que se utilice para extracción de características. Es decir que regiones

muy pequeñas, sobre todo si se trata de imágenes con ruido, pueden conducir a operadores con grandes errores (o varianzas) en las características extraídas, mientras que regiones excesivamente grandes pueden extraer características que representen variaciones de nivel de gris tomadas de regiones distintas.

- Desde el punto de vista práctico, el esfuerzo computacional requerido para calcular características de regiones, suele ser significativo. Como cada característica resulta de una secuencia de operaciones sobre una subregión de una imagen, es deseable mantener estas regiones pequeñas.

Aunque no existe una definición de textura estricta o aceptada universalmente, la noción de textura en una imagen es familiar a la mayoría de las personas.

La textura es una característica importante para el análisis de diversos tipos de imágenes, pero a pesar de su importancia, no existe una definición formal o precisa de la misma. Las técnicas de discriminación de texturas son en su mayoría Ad-hoc. [6]

Típicamente, la textura se refiere a variaciones que se perciben en el nivel de gris de una subregión de la imagen. Resulta claro que en el espectro visible, imágenes de un desierto serán notablemente diferentes en las características espaciales, de aquellas que representen forestación.

En algunos casos también, ciertos objetos pueden presentar una textura que manifieste una determinada direccionalidad, lo cual facilitaría su discriminación respecto de otras subregiones de la imagen, aunque esto último no suele ocurrir en imágenes naturales. Más adelante ampliaremos estos conceptos referidos a texturas (Seción 2.3).

2.2.4.2.1 - Características de regiones

Podemos definir a las características de las imágenes como rasgos distintivos o atributos de un campo de la imagen. Algunas de ellas son naturales ya que tales características están definidas por la apariencia visual de la imagen, mientras que las llamadas artificiales resultan de manipulaciones específicas o de ciertas mediciones de la imagen. Entre las primeras podemos contar el brillo o contraste en regiones de una imagen, bordes de objetos y texturas. Entre las características artificiales podemos mencionar amplitudes en histogramas, espectros en frecuencias espaciales, entre otras.

- **Características de amplitud**

Una de las posibles características naturales de una imagen, es la medida de amplitudes en términos de valores espectrales, luminancia, valores *tristimulus*, u otras unidades. Estas mediciones pueden tomarse sobre puntos específicos de las imágenes, o bien, sobre vecindades.

Existen muchos grados de libertad al establecer características de amplitud de las imágenes. Los valores de la imagen pueden ser utilizados en forma directa o bien, es posible realizar transformaciones lineales, no lineales e incluso no inversibles, con el fin de obtener un nuevo espacio de amplitudes en la imagen.

Las medidas de amplitud en las imágenes son de gran utilidad al intentar aislar objetos en una figura, y durante el etiquetado de tales objetos.

- **Características de histograma**

Estas técnicas estiman las distribuciones de probabilidades en la imagen en términos de mediciones de características de amplitud. [1][4]

La forma del histograma de una imagen suele proveer indicios acerca del carácter de una imagen. Por ejemplo un histograma poco distribuido, indica una imagen de bajo contraste, mientras que un histograma bimodal sugiere la presencia de dos regiones de diferente brillo.

- **Características de transformación**

Las transformaciones de las imágenes proveen la información del dominio de las frecuencias que hay en los datos. Estas características son extraídas mediante un filtrado por zonas de la imagen, en el espacio transformado seleccionado. Los filtros zonales, también llamados *máscaras de característica,* son simplemente pequeñas ranuras o aperturas. [1]

Las características de alta frecuencia pueden ser utilizadas para extracción de bordes o límites, mientras que las ranuras angulares pueden utilizarse en la detección de orientaciones.

Una combinación de ranuras angulares con un filtro pasa-bajos, o pasa-banda, o pasa-altos puede ser utilizado en la discriminación de texturas periódicas o quasi-periódicas. Otras transformadas tales como Haar y Hadamar pueden ser utilizadas para extraer características, aunque aún es necesario realizar estudios sistemáticos con el fin de determinar sus posibles aplicaciones.

2.2.5 - Técnicas de clasificación

La principal tarea luego de la extracción de características es clasificar el objeto dentro de una entre diversas categorías o clases. [1]

Los procesos de segmentación y clasificación tienen objetivos fuertemente relacionados. La clasificación puede conducir a la segmentación y viceversa. La clasificación de pixeles en una imagen es otra forma de etiquetados de componentes (labeling) que puede resultar en la segmentación de diversos objetos en la imagen. Consideremos a modo de ejemplo el caso de imágenes satelitales donde la clasificación de datos multiespectrales conduce a la segmentación en diversas regiones. De manera similar, la segmentación de imágenes mediante *template matching* como es el reconocimiento de caracteres, conduce a la clasificación o identificación de cada objeto.

Existen dos aproximaciones básicas a la clasificación, supervisada y no supervisada dependiendo de la disponibilidad de un conjunto de prototipos.

2.2.5.1 - Clasificación supervisada

Este tipo de clasificación, puede ser libre de distribución o estadística. Los métodos libres de distribución no requieren conocimiento de ninguna función de distribución de probabilidad a priori y se basan sobre razonamiento y heurísticas. Las técnicas estadísticas se basan sobre modelos de distribución de probabilidad los cuales pueden ser paramétricos o no paramétricos.

2.2.5.1.1 - Clasificación libre de distribución

Una función fundamental en el reconocimiento de patrones es la llamada función discriminante. Para un problema de k clases esta función se define de manera que la k-ésima función discriminante $g_k(x)$ toma el valor máximo si x pertenece a la clase k, es decir que, la regla de decisión está dada por:

$$g_k(x) > g_i(x) \qquad \text{para } k<>i \qquad \text{sii x pertenece a la clase k}$$

Estas funciones dividen el espacio de características n-dimensional dentro de k diferentes regiones. En aquellos casos donde x es clasificado a la clase cuyo centroide es el más cercano en distancia euclídea, el clasificador asociado es llamado clasificador de distancia (euclídea) media mínima. Una regla de decisión alternativa es clasificar x a la clase S_i si de un total de p prototipos más cercanos, el número máximo de ellos pertenece a la clase S_i; este es el clasificador de p vecinos más cercanos. Si p es igual a 1 se transforma en un clasificador de distancia mínima.

2.2.5.1.2 - Clasificación estadística

En la técnica de clasificación estadística se asume que las diferentes clases de objetos y los vectores de características poseen una función de densidad de probabilidad conjunta subyacente. Si tomamos a $P(S_k)$ como la probabilidad a priori de la clase S_k y p(x) la función de densidad de probabilidad del vector de características aleatorio x, es posible determinar un clasificador de riesgo mínimo. El clasificador de riesgo mínimo de Bayes minimiza la pérdida promedio o riesgo de asignar x a una clase errónea. El discriminante del clasificador de error mínimo está dado por

$$g_k(x) = p(x|S_k)\, P(S_k)$$

En la práctica $p(x|S_k)$ se estima a partir de los prototipos mediante técnicas paramétricas o no paramétricas.[1]

2.2.5.2 - Clasificación no supervisada o clustering

Aquí se intenta identificar clusters o agrupamientos naturales en el espacio de características. Un cluster es un conjunto de puntos en el espacio de características para los que la densidad local es grande comparado con la densidad de los puntos de características a su alrededor. Las técnicas de clustering son útiles en la segmentación

de imágenes y en la clasificación de secuencia de datos cuando se desee establecer clases y prototipos. El clustering es también útil para técnicas de cuantización de vectores en la compresión de imágenes.

Un algoritmo general para clustering se basa sobre las ideas de split y merge. Utilizando alguna medida de similitud, los algoritmos de entrada son particionados en subconjuntos. Cada partición es testeada con el fin de determinar si los diversos subconjuntos son suficientemente distintos. Aquellos subconjuntos que no sean suficientemente distintos serán unificados. El procedimiento debe ser repetido sobre cada uno de los subconjuntos hasta que no resulten nuevas subdivisiones o bien no se satisfaga algún otro criterio de convergencia. De esta manera una medida de similitud, un test de distinción y una regla de parada, son requeridos para definir un algoritmo de clustering. Finalmente debe notarse que el éxito de las técnicas de clustering se encuentra fuertemente asociado a la selección de las características. Los clusters no detectados en un espacio de características dado pueden ser fácil de detectar en coordenadas transformadas, escaladas o rotadas. En el caso de imágenes, los elementos del vector de características pueden representar valor de gris, magnitud de gradiente, fase de gradiente o color entre otros atributos.

2.3 - Texturas

La textura es una característica importante para el análisis de diversos tipos de imágenes desde las obtenidas mediante scanners multiespectrales a partir de plataformas tales como aeroplanos o satélites, las cuales son analizadas por la comunidad de remote-sensing, hasta imágenes microscópicas de cultivos de células o muestras de tejidos, las que son analizadas por la comunidad biomédica, e incluso imágenes de escenas exteriores y de superficies de objetos, que son utilizadas por la comunidad de visión computarizada. [6][1]

La textura es una característica importante para el análisis de diversos tipos de imágenes, pero a pesar de su importancia, no existe una definición formal o precisa de la misma. Las técnicas de discriminación de texturas son en su mayoría Ad-hoc. [6]

Los problemas a resolver en el análisis de texturas pueden resumirse de la siguiente manera:

- Dados un conjunto finito de clases de textura y una región texturada, determinar a cuál de estas clases pertenece la región.
- Dada una región texturada, obtener una descripción o modelo de la misma.
- Dada una imagen que presenta diversas áreas texturadas, determinar los límites entre ellas.

El primero de estos problemas suele resolverse con técnicas de reconocimiento de patrones mediante extracción de características texturales. En el segundo caso se utilizarán modelos generativos de texturas. Para la última clase de problemas se pueden utilizar las técnicas anteriores combinadas, de manera de lograr una segmentación de la imagen en base a texturas.

Los primeros trabajos de análisis de texturas de imágenes intentaron descubrir características útiles que guarden alguna relación con la finura y grosor, contraste, direccionalidad, rugosidad y regularidad de la textura de la imagen. La metodología utilizada típicamente consistía en tomar una imagen con una textura determinada y tomar diversas medidas de características texturales que permitan realizar una clasificación de la misma.

En cuanto a las imágenes aéreas, fue posible discriminar entre áreas con vegetación natural y árboles, de áreas con objetos construidos por el hombre, edificios y calles utilizando características de textura. Estas aproximaciones estadísticas incluían el uso de la función de autocorrelación, la función de densidad de potencia espectral, bordes por unidad de área, probabilidades de ocurrencia de niveles de gris, probabilidades de co-ocurrencia de niveles de gris, distribución de largos de corrida (run-length) de niveles de gris, distribución de extremos relativos y técnicas de morfología matemática.

Luego se intentó analizar las texturas de las imágenes buscando obtener un conocimiento más profundo de las texturas en las imágenes mediante la utilización de modelos generativos. Dado un modelo generativo y los valores de sus parámetros, es posible obtener mediante síntesis muestras homogéneas de texturas. Esto provee un medio teórico y visual para el entendimiento de las texturas. El análisis de texturas entonces necesitará tanto de la verificación como de la estimación. En primer lugar será

necesario verificar que una muestra de textura dada coincida con un modelo, entonces se deberá estimar los valores de los parámetros del modelo en base a la muestra observada. Algunos ejemplos de estas técnicas son autorregresión, promedio móvil, modelos de series de tiempo (extendidas a dos dimensiones), campos aleatorios de Markov y los modelos de mosaico, entre otros.

La textura suele ser observada como un patrón estructural de la superficie de los objetos.

La unidad básica de las texturas o *texel* está definida por las primitivas de nivel de gris dispuestas de acuerdo a alguna regla de distribución espacial. Las primitivas de nivel de gris son regiones con ciertas propiedades invariantes en sus niveles de gris. A su vez, la ubicación de las diversas primitivas de nivel de gris podría ser periódica, quasi-periódica o aleatoria.

De acuerdo a lo antes mencionado, una textura está definida por el número y tipo de sus primitivas y su organización espacial.

La organización espacial puede ser aleatoria, puede tener una dependencia de a pares de una primitiva respecto de otra vecina, o puede presentar una dependencia de n primitivas a la vez. Esta dependencia puede ser estructural, probabilística o funcional.

Las texturas pueden ser evaluadas cualitativamente como poseedoras de una o más de las siguientes propiedades: grosor, finura, suavidad, granulación, aleatoriedad, alineación, o bien ser multicoloreadas, regulares, irregulares o convexas. Cada una de estas cualidades relaciona alguna propiedad de las primitivas de los niveles de gris con la interacción espacial entre ellas. Desafortunadamente pocos investigadores han realizado trabajos para mapear significados semánticos en propiedades precisas de primitivas de niveles de gris y su distribución espacial.

Cuando los texels son pequeños en tamaño y la interacción espacial entre las primitivas de niveles de gris se restringe a una actividad muy localizada, la textura resultante es una microtextura. Cuando las primitivas comienzan a tener su propia forma distintiva y organización regular, tendremos una macrotextura. De acuerdo a esto, las texturas no podrán ser analizadas sin un marco de referencia en el cual se establecen las primitivas de niveles de gris. Para cualquier superficie textural, existe una escala en la cual aparecerá como suave o sin textura; a medida que la resolución se incremente, ésta podría aparecer como una textura fina y luego como gruesa, pudiendo incluso repetirse dicho ciclo.

Con el fin de utilizar en forma objetiva niveles de gris y elementos de patrones texturales, es necesario definir explícitamente los conceptos de niveles de gris y textura. Ambos conceptos no son independientes, dado que mantienen una complicada relación. En el contexto de las imágenes, tanto niveles de gris como textura existirán siempre, aunque en algunas situaciones una propiedad puede dominar a la otra. Como consecuencia se definirán ambos conceptos en conjunto.

La interrelación básica en el concepto nivel de gris-textura es que cuando una mancha de área pequeña de una imagen posee poca variación en sus primitivas de niveles de gris, la propiedad dominante será el nivel de gris. Por el contrario, cuando una mancha de área pequeña muestra gran variación en sus primitivas de niveles de gris, la propiedad dominante será la textura. Un aspecto crucial en esta distinción es el tamaño de las manchas de área pequeña, los tamaños relativos y tipos de primitivas de

niveles de gris, así como el número y disposición espacial de las primitivas distinguibles.

A medida que el número de primitivas de niveles de gris distinguibles decrezcan, predominarán las propiedades de nivel de gris. Un caso particular que ejemplificaría lo anterior es cuando la mancha de área pequeña es de sólo un pixel de tamaño, de manera que existe una única característica discreta, la única propiedad presente es un nivel de gris simple. A medida que el número de primitivas distinguibles de niveles de gris se incremente, predominarán las propiedades de textura.

En aquellos casos donde el patrón espacial de las primitivas de niveles de gris sea aleatorio y la variación de nivel de gris entre primitivas sea significativo, resultará en una textura *fina*. Cuando los patrones espaciales se tornen más definidos y las regiones de niveles de gris involucren más pixels, resultará una textura más *gruesa*.

En síntesis, para caracterizar una textura será necesario caracterizar las propiedades de las primitivas de los niveles de gris, así como la relación espacial entre ellos. Esto implica que el nivel de textura presenta una estructura de dos categorías. La primera de ellas especifica las propiedades locales que manifiestan las primitivas de niveles de gris; mientras que la segunda especifica la organización espacial de aquellas.

A continuación haremos una descripción de los métodos más relevantes en el análisis textural de las imágenes.

Desde el punto de vista del análisis de imágenes, las texturas son clasificadas generalmente como estructurales o estadísticas.

2.3.1 - Aproximaciones estadísticas

Desde un punto de vista amplio es posible afirmar que las texturas que presentan naturaleza aleatoria son adecuadas para caracterización estadística. En particular, las texturas naturales suelen ser aleatorias, mientras que las artificiales suelen responder a un patrón determinístico o periódico. De acuerdo a lo mencionado anteriormente, describiremos algunos métodos que suelen utilizarse en el tratamiento de imágenes con estas características.

2.3.1.1 - Características de Histogramas de Primer Orden

La distribución de probabilidad de primer orden de las amplitudes de una imagen I puede ser definida como:

$$P(b) \equiv P_R\{ I(i, j) = b \}$$

donde $0 \leq b \leq L\text{-}1$ denota el nivel de amplitud cuantizado. La estimación del histograma de primer orden de la $P(b)$ estaría dado por

$$P(b) \approx N(b) / M$$

donde M representa el número total de pixels en una ventana centrada en (i, j) y N(b) es el número de pixels de amplitud b en dicha ventana. En general se supone un modelo estacionario, entonces la ventana suele ser la imagen completa.

Las siguientes medidas fueron formuladas con el fin de describir la forma del histograma de primer orden de una imagen.

$$\text{Momentos: } m_i = E[\, I(i,j)\,]^i = \sum_{b=0}^{L-1} b^i P(b)$$

$$\text{Momentos Absolutos: } ma_i = E[\, |\, I(i,j)\,|\,]^i = \sum_{b=0}^{L-1} |\, b\,|^i P(b)$$

$$\text{Momentos Centrales: } \mu_i = E\{\, I(i,j) - E[\, I(i,j)\,]^i\,\} = \sum_{b=0}^{L-1} (\, b - m_1\,)^i P(b)$$

$$\text{Momentos Centrales Absolutos: } \mu a_i = E[\, |\, I(i,j) - E[\, I(i,j)\,]\,|^i\,] = \sum_{b=0}^{L-1} |\, b - m_1\,|^i P(b)$$

$$\text{Entropía: } H = E\{\, -\log_2 [\, P(b)\,]\,\} = \sum_{b=0}^{L-1} P(b) \log_2 [\, P(b)\,]$$

Tabla 1

Algunas medidas utilizadas comúnmente son: *dispersión* = μa_1, *media* = m_1, *varianza* = μ_2, *energía promedio* o valor cuadrático medio = m_2, *skewness* o sesgo = μ_3, *kurtosis* = μ_4 - 3 (el factor 3 aquí normaliza a la medida de kurtosis a cero para los procesos Gaussianos con media cero).

Un histograma estrecho, indicará regiones de bajo contraste. La varianza puede ser utilizada para medir cambios locales en las amplitudes de las imágenes. La desviación standard enfatiza los bordes fuertes en la imagen, mientras que las características de dispersión extraen la estructura de bordes más fina. A su vez, media, extrae características de frecuencias espaciales bajas.

Generalmente estas características son medidas en pequeñas ventanas móviles W. Algunas de estas características pueden ser tomadas sin determinar explícitamente el histograma. A modo de ejemplo consideremos las siguientes medidas, considerando ventanas centradas en (k, l).

$$m_i(k,l) = (1/M) \sum_{(m,n) \in W} \sum [I(m-k, n-l)]^i$$

$$\mu_i(k,l) = (1/M) \sum_{(m,n) \in W} \sum [I(m-k, n-l) - m_1(k,l)]^i$$

Algunas de estas métricas pueden ser calculadas realizando una normalización en términos de la Varianza de la ventana móvil bajo observación. Esta metodología propuesta por Irons y Peterson fue adoptada por un producto comercial llamado Erdas Imagine para mejorar la visualización de texturas [13].

2.3.1.2 - Función de autocorrelación(ACF)

Las texturas se relacionan con el tamaño espacial de las primitivas de niveles de gris en las imágenes. Las primitivas de tamaño significativo indican texturas *gruesas,* mientras que las primitivas tonales más pequeñas son indicativas de texturas más *finas.* La ACF es una característica que describe el tamaño de aquellas primitivas. Este tamaño puede ser representado por el ancho de la ACF dado por $r(k,l) = m_2(k,l)/m_2(0,0)$ de acuerdo a los *momentos* de las características espaciales dados en la Tabla 1. Se espera que la rugosidad de la textura sea proporcional al ancho de la ACF, la cual puede ser representada por las distancias x_0, y_0 tal que

$$r(x_0, 0) = r(0, y_0) = ½.$$

Otras medidas de dispersión de la ACF son obtenidas mediante las *funciones generadoras de momentos*:

$$M(k,l) \equiv \sum_m \sum_n (m-\mu_1)^k (m-\mu_2)^l r(m,n)$$

$$\text{donde} \quad \mu_1 \equiv \sum_m \sum_n mr(m,n) \quad \mu_2 \equiv \sum_m \sum_n nr(m,n)$$

Algunas características de especial interés son el *perfil de dispersión* M(2,0), *cross-relation* M(1,1), y la *dispersión de segundo grado* M(2, 2).

La calibración de la dispersión de la ACF sobre una escala de textura de grano fino depende de la resolución de la imagen. Esto se debe a que, como fue mencionado, una región aparentemente no texturada para una resolución dada, podría aparecer como una textura fina en una imagen de mayor resolución, y como una textura gruesa en una de resolución menor. En general, la ACF no es suficiente por sí misma para distinguir entre diversas texturas, ya que regiones con diferentes texturas pueden tener la misma ACF.

2.3.1.3 - Transformaciones de Imágenes

Algunas características de textura tales como grosor, finura u orientación pueden ser estimadas mediante técnicas de filtrado lineal generalizadas utilizando transformaciones de imágenes. Una transformación bidimensional de la imagen de entrada es pasada a través de distintos filtros pasa banda o máscaras, entonces el resultado representa una característica transformada. Diferentes tipos de máscaras son adecuados para el análisis de texturas. Utilizando ranuras circulares se realizan medidas de energía en diferentes frecuencias espaciales. Las ranuras angulares son útiles en la detección de características de orientación en las texturas. La combinación de ranuras circulares y angulares suelen ser utilizadas para detección de texturas periódicas o quasi-periódicas. Estas técnicas fueron utilizadas para discriminación de diversos tipos de terreno.

Cuando se utiliza esta metodología para análisis de texturas, la imagen digital es particionada en un conjunto de pequeñas subimágenes cuadradas no superpuestas. Si el tamaño de las subimágenes es de n x n, entonces los n^2 niveles de gris en las subimágenes pueden ser pensados como las componentes de un vector de dimensión n^2. Así, el conjunto de subimágenes constituye un conjunto de vectores en un espacio de dimensión n^2. En la técnica de transformación, cada uno de estos vectores es reservado en un nuevo sistema de coordenadas. En el caso particular de trabajar con la transformada de Fourier, el conjunto de funciones base será el sinuzoidal complejo, mientras que si se trata de la transformada de Hadamar, se utilizarán las llamadas funciones de Walsh como conjunto base. Este conjunto de funciones base en el sistema transformado permite una interpretación que relaciona las frecuencias espaciales con determinadas características de las texturas, por lo que las transformaciones pueden ser de utilidad.

2.3.1.4 - Energía Textural

En esta aproximación, en primer término la imagen es convolucionada con un conjunto de kernels. Luego, cada imagen convolucionada es procesada con un operador no lineal con el fin de determinar la energía textural total en cada vecindad de 7x7 pixels. [6]

A partir de un conjunto de experimentos, utilizando diversas muestras, Laws sostiene que su aproximación es capaz de distinguir entre 8 texturas diferentes con una precisión de 94%, mientras que utilizando la técnica de co-ocurrencia espacial de niveles de gris, obtuvo una precisión de tan sólo el 72%.

Una dificultad que presenta esta aproximación es la posibilidad de introducir errores significativos en los límites entre diversas texturas, ya que en estos sitios, todas las aproximaciones basadas en vecindades soportan una mezcla de texturas, y puede ocurrir que las medidas tomadas sobre una vecindad en el límite entre texturas, coincida con las medidas tomadas sobre una tercer textura en la imagen.

Para solucionar este problema, Hsiao y Sawchuck realizan un nivel más de pocesamiento sobre cada imagen de energía textural. Para cada pixel de coordenadas (r,c), calculan las medidas de media y varianza de las cuatro vecindades (15x15) para las que el anterior es el pixel sudoeste, sudeste, noroeste y nordeste respectivamente. A

partir de esto crean una imagen de energía textural suavizada, donde cada pixel (r,c) tendrá el valor de media de la vecindad de 15x15 cuyo valor de varianza haya resultado el mínimo de las cuatro vecindades tenidas en cuenta.

2.3.1.5 - Densidad de Bordes

Tanto la función de autocorrelación como las transformaciones de texturas trabajan en base a la relación entre las texturas y las frecuencias espaciales.

La rugosidad de una textura aleatoria puede también ser representada por la densidad de pixels de borde que posea. Dado un mapa de bordes, la densidad de los bordes se mide como la cantidad promedio de pixels de borde por unidad de área.

Un borde que pasa a través de un pixel, puede ser detectado comparando los valores de diversas propiedades locales obtenidas a partir de vecindades del pixel. Para localizar microbordes, se utilizarán pequeñas vecindades, mientras que para detección de macrobordes, vecindades más grandes. Una de las propiedades que puede ser utilizada con este fin es el gradiente de la imagen, el cual puede ser estimado utilizando el gradiente rápido de Roberts. De esta manera, una medida de textura de una subimagen puede ser obtenida calculando el gradiente de Roberts para cada pixel de la subimagen, y determinar luego el valor promedio del gradiente para dicha subimagen.

2.3.1.6 - Densidad de Extremos Relativos

La idea aquí es calcular el número de máximos y mínimos relativos por unidad de área con el fin de utilizarlo como una medida de textura. Una forma de realizar esta medición es teniendo sólo en cuenta a los dos vecinos en dirección horizontal y señalar a un pixel como máximo local si es mayor o igual que ambos vecinos, o como mínimo local si es menor o igual que ellos. Luego de esto, se centra una ventana cuadrada sobre cada pixel y se cuenta la cantidad de vecinos que son extremos relativos, ya sea máximo o mínimo, indistintamente. Una mejora de esta técnica utiliza una imagen suavizada con el fin de disminuir el impacto del ruido en los resultados.

Es posible agregar a cada extremo dos propiedades: alto y ancho. En el caso de una dimensión, la altura de un extremo puede ser definida como la diferencia con su vecino adyacente de valor más cercano, mientras que el ancho de un máximo será la distancia entre los dos mínimos adyacentes a él. Análogamente, el ancho de un mínimo, estará dado por la distancia entre los dos máximos adyacentes.

En el caso bidimensional, se presentan algunas dificultades adicionales. Una posible forma de encontrar extremos bajo estas condiciones es mediante el uso iterado de algunos operadores de vecindades recursivos, que propaguen los valores extremos en una forma apropiada. Las áreas conectadas de extremos relativos pueden abarcar desde un pixel simple, hasta una superficie conteniendo varios de ellos.

Una forma de marcar a un pixel contenido en el área de un extremo relativo de tamaño N es asociándole el valor de altura h indicando que pertenece a una región cuyo extremo tiene altura h, o bien con el valor h/N. Otra posibilidad es macar al pixel más cercano al centro de la región con el valor h, asociándole al resto de los pixels de la

región el valor 0. De esta manera, dada una ventana, es posible obtener la altura promedio de los pixels involucrados. Una alternativa es suponer altura 1 y simplemente calcular el número de extremos relativos por unidad de área.

Otras propiedades más complejas que pueden tenerse en cuenta son, por ejemplo, determinar el conjunto de pixels alcanzables únicamente desde un máximo relativo dado, mediante caminos monótonos decrecientes. Este conjunto conformará una región conectada donde los pixels de borde serán los mínimos relativos. La altura relativa de estas regiones estará dada por la diferencia entre su máximo y el mayor de sus pixels de borde, su tamaño podrá establecerlo la cantidad de pixels que la compongan, mientras que su forma podrá determinarse a partir de características tales como la elongación, circularidad y/o ejes de simetría. Los histogramas y diversas estadísticas de histograma sobre estas propiedades de primitivas son medidas adecuadas de las propiedades de primitivas texturales.

2.3.1.7 - Características de Histograma de Segundo Orden

De acuerdo a los resultados de las investigaciones de Julesz, la percepción visual humana no puede siempre distinguir campos de textura aleatorios que presenten distribuciones de probabilidad de segundo orden iguales. Estos resultados han sido útiles en el análisis y síntesis de diversos tipos de textura. De esta manera, dos texturas diferentes pueden frecuentemente ser discriminadas comparando sus histogramas de segundo orden. [6][1]

La aproximación utilizada aquí se basa en la dependencia espacial de los niveles de gris, y se caracteriza a las texturas mediante la co-ocurrencia de sus niveles de gris.

Las texturas *gruesas* serán aquellas cuya distribución cambie muy suavemente en función de la distancia, mientras que las texturas finas manifestarán rápidos cambios en su distribución con los cambios de distancia.

La potencia que presenta esta aproximación, es que caracteriza la interrelación de los niveles de gris en un patrón estructural, de manera que permanece invariante a transformaciones de nivel de gris monótonas. El problema que presenta este método es que no es sensible a las variaciones de forma en las primitivas tonales. Resulta difícil entonces trabajar correctamente con texturas compuestas por primitivas de área grande. Otra dificultad que manifiesta es su imposibilidad de capturar la relación espacial entre primitivas de más de un pixel en tamaño.

Si se tienen dos pixels u_1 y u_2 a una distancia relativa r y orientación θ, la función de distribución puede ser escrita explícitamente como:

$$p_u(x_1, x_2) = P[u_1 = x_1, u_2 = x_2] = f(r, \theta; x_1, x_2)$$

Es conveniente señalar que tanto las características de los histogramas de segundo orden, como las mismas matrices en forma directa, pueden ser utilizadas a la hora de clasificar.

Las matrices de co-ocurrencia fueron utilizadas por Haralick y Shanmugam para el análisis de texturas en imágenes satelitales, obteniendo una clasificación con un 89% de precisión. A su vez Vickers y Modestino sostienen que la utilización de características de las matrices de co-ocurrencia para clasificación no es óptimo,

argumentando que se obtienen mejores resultados utilizando en forma directa estas matrices en un clasificador de máxima verosimilitud. Estos autores obtuvieron una precisión cercana al 95% de identificaciones correctas, distinguiendo corteza de árbol, cuero de ternero, lana, arena, piel de cerdo y burbujas de plástico, entre otras texturas. [6]

2.3.1.8 - Modelos de Autorregresión

Estos modelos proveen una forma de estimar el valor de gris de un pixel, dados los niveles de gris de una vecindad que lo contenga, con el fin de caracterizar una textura.

La dependencia lineal de un pixel respecto de otro en una imagen es bien conocida y puede ser ilustrada mediante la función de autocorrelación. Esta dependencia lineal es utilizada en los modelos de autorregresión en el tratamiento de texturas. Es posible investigar los parámetros de una textura dada, lo cual permitirá reproducirla exitosamente en forma sintética, estableciendo previamente las condiciones de borde iniciales. El método trabaja de la siguiente forma: Dados una imagen de ruido generada aleatoriamente y cualquier secuencia de K valores de gris, el próximo nivel de gris puede ser sintetizado como una combinación lineal de los valores sintetizados previamente más una combinación lineal de L valores previos de ruido aleatorio. Los coeficientes de estas combinaciones serán los parámetros del modelo. Estos modelos unidimensionales trabajan adecuadamente en la estimación de texturas que presentan direccionalidades de 0° y 90°, pero su performance puede decaer cuando la direccionalidad de las texturas sea diagonal. Un mejor desempeño sobre cualquier tipo de textura puede ser alcanzado con modelos bidimensionales.

El modelo autorregresivo puede ser utilizado en aplicaciones de segmentación en base a texturas, así como en la síntesis de las mismas.

Con el fin de clasificar un pixel dado como perteneciente a una determinada clase de textura, bastará con realizar la estimación del mismo con los modelos de las posibles clases y tomar al estimador más cercano, si la diferencia con el valor real del pixel no supera un determinado umbral.

2.3.2 - Aproximación estructural

De acuerdo a las definiciones de Jain, las texturas estructurales puras presentan texels determinísticos, los cuales se repiten de acuerdo a alguna regla de ubicación que especificará dónde se ha de colocar cada uno de ellos, esta distribución a su vez, puede ser de naturaleza determinística o aleatoria. Los texels serán aislados identificando un grupo de pixels que presenten ciertas propiedades invariantes, los cuales se repetirán en una imagen dada. [1]

Los texels pueden estar definidos por su nivel de gris, su forma o bien, por la homogeneidad respecto de alguna propiedad local tal como tamaño, orientación o histograma de segundo orden.

La regla de ubicación define la relación espacial entre los texels. Esta relación espacial puede ser expresada en términos de adyacencia, distancia más cercana y periodicidad entre otros, para el caso de reglas de ubicación determinística. Generalmente, cuando se está en presencia de texturas donde la regla de ubicación de las primitivas tonales es determinística, la textura es denominada *fuerte*. De la misma manera, cuando se trata de texels cuya regla de ubicación produce un comportamiento aleatorio, la textura asociada suele etiquetarse como *suave* y las reglas de ubicación pueden ser expresadas en términos de medidas tales como densidad de bordes, run-lengths o texels conectados maximalmente y densidad de extremos relativos, entre otros.

2.3.3 - Otras aproximaciones

Una aproximación que combina las aproximaciones estructural y estadística es una que se basa en los llamados Modelos de Mosaico. Estos modelos representan procesos geométricos aleatorios, los cuales son construidos en dos etapas. La primera de ellas provee un método que permitirá la partición del plano en pequeñas celdas, mientras que la segunda etapa asignará valores apropiados a las mismas. De esta manera es posible realizar mediciones de diversas texturas con el fin de alcanzar su caracterización estadística y/o clasificación.

2.3.4 - Síntesis de Texturas

La síntesis de texturas trata simplemente al conjunto de técnicas que permiten la generación de imágenes de texturas en forma artificial.

Existen muchas aplicaciones en el procesamiento de imágenes en las que estas técnicas son de utilidad. Por ejemplo, si una región de una figura se ha perdido, o bien se encuentra excesivamente corrupta por errores, es posible generar una textura artificial con el fin de reemplazar los datos perdidos. En aplicaciones de codificación de imágenes, las regiones texturales carentes de detalles significativas, pueden ser detectadas y medidas; luego, las regiones artificiales pueden ser substituidas por las originales. En principio la codificación de texturas artificiales será más eficiente que la codificación directa de las imágenes.

Una aproximación fundamental a la síntesis de texturas es aquella que dada una primitiva tonal, la repite en el espacio de acuerdo a alguna regla de ubicación. La primitiva tonal a su vez, podría ser extraída de alguna región conteniendo una textura natural, o bien puede ser construida a partir de algún elemento de primitiva básico como un punto o una pequeña línea. La ubicación de las primitivas puede ser determinística, aleatoria o bien puede seguir alguna estrategia combinada de ambas. Será necesario realizar algunas consideraciones en cuanto al efecto de borde que pueda presentarse en forma indeseada al repetirse la primitiva tonal en las diversas ubicaciones. Generalmente será imprescindible la utilización de alguna técnica de suavizado sobre la *interferencia* entre primitivas.

Otra técnica frecuentemente utilizada se basa en la generación de campos aleatorios bidimensionales correlacionados. Como se mencionó anteriormente los

humanos no siempre son capaces de distinguir entre texturas que posean las mismas estadísticas de segundo orden. En base a esto, si se encuentra disponible un conjunto adecuado de medidas de una textura, una forma de realizar síntesis de imágenes es mediante un proceso de segundo orden bidimensional, el cual genere una imagen de textura con medidas coincidentes con las disponibles a priori. Estas medidas a su vez, pueden ser obtenidas a partir da alguna muestra natural de textura, o bien puede ser generada en forma aleatoria.

CAPÍTULO 3

3 - Herramientas Utilizadas

3.1 - El Lenguaje de Programación C++

Este lenguaje de programación de propósito general fue diseñado como un superconjunto del Lenguaje C. Agregado a los recursos que ofrece este último, C++ proporciona mecanismos flexibles y eficientes para definir nuevos tipos. Un programador puede dividir una aplicación en fragmentos manejables, definiendo tipos nuevos, que se correspondan con los conceptos involucrados en la aplicación. Esta técnica de construcción de programas suele denominarse abstracción de datos. Los objetos de algunos tipos definidos por el usuario, contienen información de tipo. Tales objetos se pueden emplear de manera cómoda y segura en contextos en los cuales no se puede determinar su tipo en el momento de la compilación. En general, se dice que los programas que utilizan objetos de tales tipos, están basados en objetos. Cuando estas técnicas son empleadas correctamente, se obtienen programas más cortos y más fáciles de comprender y mantener.[7]

El concepto clave en C++ es el de *clase*. Una clase es un tipo definido por el usuario. Las clases proveen ocultamiento de información e iniciación garantizada, conversión implícita de tipos definidos por el usuario, determinación dinámica de tipos, administración de memoria controlada por el usuario y mecanismos para sobrecargar operadores (overload).

C++ conserva la capacidad de C para manejar en forma eficiente, los objetos fundamentales de la máquina (bits, bytes, palabras, direcciones, etc), lo cual posibilita la construcción con un alto grado de eficiencia de los tipos definidos por el usuario.

C++ y sus bibliotecas estándar se han diseñado pensando en la portabilidad.

Durante la evolución del lenguaje, se trabajó con el fin de mejorar sus capacidades en cuanto a la abstracción de datos y a la programación orientada a objetos. Estas mejoras fueron apuntadas a mejorar al lenguaje como instrumento para escribir bibliotecas de alta calidad de tipos definidos por el usuario; entendiendo que una biblioteca de alta calidad es aquella que proporciona al usuario un concepto en forma de una o más clases cuya utilización es cómoda, segura y eficiente. En este contexto, seguridad significa que la clase ofrece una interfaz específica, segura respecto a los tipos, entre los usuarios de la biblioteca y sus proveedores; eficiencia significa que el empleo de la clase no impone un costo adicional significativo en términos de tiempo de ejecución o de espacio, en comparación con el código escrito en C. Entre las características agregadas durante la evolución del lenguaje podemos mencionar: una resolución refinada de la sobrecarga de operadores, recursos para administración de memoria, mecanismos para controlar el acceso, enlace seguro con respecto a los tipos, funciones miembro *const* y *static*, clases abstractas, herencia múltiple, patrones (templates) y manejo de excepciones.

3.1.1 - Reseña histórica

Evidentemente la principal deuda de C++ es con C. C es conservado como un subconjunto, al igual que el énfasis puesto en C en cuanto a recursos de bajo nivel útiles en la realización de las tareas más exigentes en la programación de sistemas. La otra fuente principal de inspiración fue Simula67; el concepto de clase (incluyendo clases derivadas y funciones virtuales) se tomó prestado de ese lenguaje. La capacidad de C++ para sobrecargar operadores y la libertad de colocar una declaración en cualquier lugar donde pueda aparecer un enunciado se asemejan a las de Algol68.

Las principales áreas modificadas durante la evolución del lenguaje fueron la resolución de sobrecargas, el enlace y los recursos para administración de memoria. Se realizaron también varios cambios secundarios con el fin de aumentar la compatibilidad con C. A su vez, se añadieron varias generalizaciones y unas cuantas extensiones de importancia, entre las que podemos mencionar herencia múltiple, funciones miembro *static*, funciones miembro *const*, los miembros protected, los patrones y el manejo de excepciones. El objetivo general de estas extensiones y modificaciones fue convertir a C++ en un lenguaje mejor para escribir y emplear bibliotecas.

El recurso de patrones fue diseñado, en parte, para formalizar el empleo de macros, y fue sugerido parcialmente por los genéricos de Ada, (tanto sus cualidades como sus defectos) y en parte también , por los módulos parametrizados de Clu. De manera similar, el mecanismo de C++ para manejo de excepciones se inspiró parcialmente en Ada, en Clu y algo también en ML.

Otros adelantos como herencia múltiple, las funciones miembro static y las funciones virtuales puras, según el mismo Stroustrup, surgieron como generalizaciones motivadas por la experiencia en la utilización de C++ más que por ideas importadas de otros lenguajes.

3.1.2 - Algunas restricciones

Un lenguaje de programación sirve para dos propósitos relacionados. Proporciona un vehículo para que el programador especifique las acciones a ejecutar, y provee un conjunto de conceptos que le sirven al programador para pensar qué es posible hacer. La primer consideración requiere idealmente de un lenguaje cercano a la máquina o de bajo nivel, a fin de poder manejar todos los aspectos importantes de una máquina en forma sencilla y eficiente, y razonablemente obvia para el programador. El segundo aspecto requiere idealmente un lenguaje cercano al problema a resolver, con el fin de poder expresar directa y concisamente los conceptos de una solución. En el diseño de los recursos que se agregaron a C para crear C++ se tuvo principalmente esto en consideración.

El Lenguaje C++ fue diseñado bajo estrictas restricciones en cuanto a criterios de compatibilidad, consistencia interna y eficiencia [7], pero entre los objetivos del diseño inicial del lenguaje, no se contemplaron determinadas características, lo que podría llevar a

- Incompatibilidad con C tanto en código fuente como a nivel de Linker.

- Overhead en tiempo de ejecución o espacio requerido para un programa escrito en lenguaje C.

- Incrementar significativamente el tiempo de compilación de comparado con C.

- Implementaciones agregando requerimientos sobre el ambiente de programación. (linker, loader, etc), ya que no serían posibles implementaciones en forma simple y eficiente en un ambiente de programación C tradicional.

Estas características podrían haber sido provistas, pero no fue realizado de esta manera dado que estos criterios sólo podrían ser cumplimentados incluyendo garbage collection, clases parametrizadas, excepciones, herencia múltiple, soporte de concurrencia e integración en un ambiente de programación.

No todas estas posibles extensiones podrían ser apropiadas para C++, dado que podría llegarse a un lenguaje grande, ineficiente, inmanejable y confuso, a menos que se contemplen fuertes restricciones en el momento de seleccionar y diseñar dichas características.

Estas restricciones iniciales sobre el diseño del lenguaje, probablemente hayan sido beneficiosas y continúen siendo una guía durante su evolución.

3.1.3 - Límites de C++

C++ está diseñado para ser un 'mejor C', para soportar la abstracción de datos y la programación orientada a objetos. Esto lo hace sin dejar de cumplir el requisito de ser útil para las tareas más exigentes de programación de sistemas.

Un problema importante de diseño en el caso de un lenguaje definido para aprovechar las técnicas de ocultamiento de datos, abstracción de datos y programación orientada a objetos, es que para tener derecho a ser considerado un lenguaje de propósito general deba poseer las siguientes características:

- Ejecutarse en máquinas tradicionales
- Coexistir con sistemas operativos y lenguajes tradicionales
- Competir con los lenguajes de programación tradicionales en términos de eficiencia durante la ejecución.
- Servir para todas las áreas de aplicación importantes.

Esto implica la necesidad de disponer de recursos para trabajo numérico efectivo (aritmética de punto flotante sin costos adicionales que harían parecer atractivo a Fortran) y para obtener acceso a memoria de manera que sea posible escribir manejadores de dispositivos. También se debe poder escribir llamadas que se ajusten a las muchas veces extrañas normas requeridas para las interfaces con sistemas operativos tradicionales. Además debe ser posible llamar funciones escritas en otros lenguajes desde un lenguaje que soporte la programación orientada a objetos, y que las funciones escritas en el lenguaje que soporte la programación orientada a objetos, sean llamadas desde un programa escrito en otro lenguaje.

Otra implicación es que un lenguaje de programación no puede depender por completo de mecanismos que no se puedan realizar de manera eficiente en una arquitectura tradicional y que aún así se espere utilizarlo como un lenguaje de propósito general.

La alternativa de incluir características de bajo nivel en un lenguaje, es manejar las principales áreas de aplicación empleando lenguajes distintos de bajo nivel (como C o algún asembler) para muchas tareas. El diseño de C++ garantiza que cualquier cosa que se pueda realizar en C, se podrá hacer en C++ sin agregar costos durante la ejecución. En general, el diseño de C++ asegura que no se agreguen costos adicionales sin una solicitud explícita del programador.

C++ está diseñado de modo tal que depende de la tecnología de los compiladores para mantener la consistencia de los programas y producir versiones ejecutables, compactas y eficientes. La intención es que la verificación estricta de tipos y el encapsulamiento sean los medios principales para controlar la complejidad de los programas. Esto es de particular importancia en el caso de programas grandes desarrollados por muchas personas y cuyos usuarios no son los programadores originales, o ni siquiera programadores. Debido a que casi ningún programa se escribe sin depender de bibliotecas escritas por otros, lo anterior se aplica prácticamente a todos los programas.

C++ está diseñado para soportar la noción de programa como modelo de algún aspecto de la realidad y la de clase como la representación concreta de un concepto de la aplicación. A esto se debe la presencia generalizada de las clases en los programas escritos en C++, y por tanto resulta indispensable que el concepto de clase sea flexible, que los objetos sean compactos y que su utilización sea eficiente. Si el empleo de las clases fuera inconveniente o costoso, éstas no se utilizarían y los programas se convertirían en programas del estilo 'un mejor C'. Los usuarios perderían entonces los principales beneficios que se procuró ofrecer con C++.

3.1.4 - Soporte de la Programación Orientada a Objetos

El soporte básico que un programador necesita para escribir programas orientados a objetos consiste en un mecanismo de clases con herencia y un mecanismo que permita que las llamadas a funciones miembro dependan del tipo real de los objetos, en los casos en que no sea posible determinar el tipo real en tiempo de compilación. El diseño del mecanismo para llamado de funciones miembro es crítico. Son importantes además los recursos para soportar las técnicas de abstracción de datos, dado que los argumentos a favor de la abstracción de datos y de sus refinamientos para apoyar un elegante empleo de los tipos, son igualmente válidos cuando se cuenta con un soporte para la programación orientada a objetos. El éxito de las dos técnicas depende del diseño de los tipos y de su facilidad de uso, como así también de su flexibilidad y eficiencia. La programación orientada a objetos permite que los tipos definidos por el usuario sean más flexibles y generales que los diseñados exclusivamente mediante técnicas de abstracción de datos.

3.1.4.1 - Clases

Una clase es el bloque de construcción más importante para el sofware orientado a objetos. Una clase define el tipo de los datos, el cual consiste tanto de un conjunto de estados posibles, como del conjunto de operaciones que realizan las transiciones entre dichos estados. Una clase provee entonces el conjunto de operaciones, usualmente públicas, y un conjunto de datos representando a los valores abstractos que una instancia de tal clase puede tener.

3.1.4.2 - Objetos

Un objeto, en este contexto, es simplemente una región de memoria de almacenamiento con una semántica asociada. En la programación orientada a objetos en general y en el Lenguaje C++ en particular, Objeto significa una instancia de una clase. De esta manera una clase define el comportamiento de algún conjunto de objetos.

3.1.4.3 - Encapsulamiento.

Podemos decir que una interfaz es buena cuando provee al usuario (programador) una visión simplificada de alguna pieza importante de software (una clase). Para alcanzar tal visión simplificada es necesario ocultar intencionalmente aquellos detalles que no sean relevantes para su uso, lo cual reduce significativamente la cantidad de errores en la programación.

El encapsulamiento podría ser definido entonces como la prevención de acceso no autorizado a determinadas piezas de información o funcionalidad.

La clave aquí consiste en separar la parte volátil, de la parte estable de las piezas de software. El encapsulamiento entonces evita el acceso a las partes volátiles, de manera que otras piezas de software sólo puedan acceder a las porciones estables. Otra ventaja importante que provee el encapsulamiento, es que la parte volátil de las piezas de software pueden ser cambiadas sin afectar a los usuarios.

La parte volátil referida está dada por los detalles de implementación, mientras que la parte estable antes mencionada está dada por la interfaz, la cual está dada simplemente por las funciones públicas member y friend.

Diseñar una interfaz clara y separarla de su implementación permite a los usuarios utilizarla, mientras que el encapsulamiento fuerza a los usuarios a utilizar tal interfaz.

C++ posee un mecanismo general de protección/encapsulamiento, implementado mediante miembros protegidos, públicos y privados. Los miembros públicos pueden ser accedidos desde cualquier lugar. Por ejemplo los métodos de Push y Pop de una Pila. Los miembros privados sólo pueden ser accedidos desde el interior de la clase que los contienen. En el ejemplo anterior, la representación de la Pila será típicamente privada.

Los miembros protegidos son accesibles desde una clase y también desde las subclases o clases derivadas. La representación de la Pila podría ser declarada protegida, permitiendo el acceso a las subclases. El lenguaje C++ permite también especificar friends de las clases, a los cuales se les permitirá el acceso a todos los miembros.

Otro item en la protección dice si ésta se realizará por clase o por objeto. El método más comúnmente empleado es el de protección por clase, adoptado también por C++. Aquí los métodos de las clases pueden acceder a cualquier objeto, y no sólo al receptor del método, como en el caso de protección por objeto.

3.1.4.4 - Herencia

La herencia es lo que diferencia la programación orientada a objetos de la programación basada en tipos de datos abstractos. Le herencia entonces es utilizada como un mecanismo de especificación.

El lenguaje C++ permite expresar la herencia mediante la palabra reservada public. Un ejemplo de su utilización está dado por

```
class Automovil: public Vehículo {
        public:
        // ...
        };
```

Le relación anterior puede ser pensada de las siguientes formas:

Automóvil *es un tipo de* Vehículo

Automóvil *es un derivado de* Vehículo

Automóvil *es una especialización de* Vehículo

Automóvil *es una subclase de* Vehículo

Vehículo *es una clase base de* Automóvil

Vehículo *es una superclase de* Automóvil

C++ implementa algunas reglas de restricción de accesos, con el fin de obtener encapsulamiento a través de la jerarquía de clases. A continuación enumeramos las más relevantes.

Una función o un dato miembro declarado en la sección privada de una clase puede sólo ser accedida por funciones miembro o friends de la clase.

Un miembro de la clase, ya sea dato o función, declarado en una sección protegida de una clase, puede sólo ser accedido por funciones miembro y friends de la clase, y por funciones miembro o friends de las clases derivadas.

Un miembro de la clase, ya sea dato o función , declarado en la sección pública de una clase puede ser accedido sin restricciones.

Las clases derivadas tienen el acceso restringido a los miembros privados de la clase base con el fin de preservarlas de los posibles futuros cambios de los miembros privados en esta clase.

Una clase posee dos interfaces distintas, dirigidas hacia dos tipos de clientes diferentes:

Una interfaz pública, pensada para servir a las clases no relacionadas.

Una interfaz protegida, pensada para servir a las clases derivadas.

Las clases base entonces deberían tener sus partes más relevantes como privadas, y utilizar funciones inline protegidas mediante las cuales, las clases derivadas accederán a los datos privados de la clase base. De esta manera, las partes más relevantes de la clase base podrán ser cambiadas, sin afectar a las clases derivadas, a menos que se realicen cambios en las funciones de acceso protegido.

3.1.4.5 - Herencia Múltiple

Si una clase A es base de otra clase B, B hereda los atributos de A; es decir que los objetos de la clase B son A, además de cualquier otra cosa que pudieran ser. En vista de esta explicación, podría ser útil que una clase B pueda ser heredera de dos clases base A1 y A2. Esto se denomina herencia múltiple.

En caso de contar sólo con herencia simple, pueden producirse repeticiones de código y/o pérdida de flexibilidad. Mediante la posibilidad de especificar herencia múltiple es posible conservar las facilidades antes mencionadas sin una pérdida significativa en cuanto a tiempo y espacio en comparación con la herencia simple, sin sacrificar la verificación estática de tipos.

Las ambigüedades se resuelven mediante la determinación explícita, especificando cuál de las clases base debe responder al mensaje que pueda poseer más de una de ellas. Por ejemplo:

```
class A1 {
public:
    void rastreo();

}

class A2 {
public:
    void rastreo();

}

class B: public A1, public A2 {
    // No se define rastreo aquí
};
```

```
class B: public A1; public A2 {
public:
    void rastreo(){
        A1::rastreo();
        A2::rastreo();
    }
};
```

3.1.4.6 - Polimorfismo en C++

En C++, Overriding es el término utilizado para redefinir un método en una clase derivada. Para que sea posible la redefinición del comportamiento, el método debe ser declarado *virtual.*

Desde un punto de vista funcional, Stroustrup [7] asegura que el uso de clases y funciones virtuales suele ser llamado programación orientada a objetos. Además, la capacidad de llamar a una variedad de funciones utilizando exactamente la misma interfaz - como es provisto por las funciones virtuales de C++ - es a veces llamado polimorfismo.

3.1.4.7 - Mecanismos de llamado

El recurso decisivo del lenguaje para soportar la programación orientada a objetos es el mecanismo con el cual se llama a una función miembro para un objeto determinado, cuyo tipo exacto se desconoce en el momento de la compilación. En aquellos casos donde se invoque una función declarada virtual, el mecanismo de llamada debe examinar al objeto con el fin de determinar qué función real debe ejecutar. Una forma común de implementar este mecanismo consiste en convertir el nombre de la función invocada en un índice para acceder a una tabla de punteros a funciones. Las funciones en la tabla de funciones virtuales, llamada *vtbl,* permiten una utilización correcta del objeto, aún cuando el invocador de la función desconozca el tamaño del objeto, la disposición de los datos en el objeto y en la tabla de funciones virtuales. Este mecanismo de llamada de funciones virtuales puede hacerse prácticamente tan eficiente como el mecanismo de llamada normal de funciones.

3.1.4.8 - Verificación de tipos

C++ depende mucho de la verificación estática de tipos. El propósito de la verificación es asegurar que el programa sea tan consistente en la utilización de tipos como sea posible determinar antes de ejecutarlo, con el fin de asegurar la ausencia de grandes clases de errores en los programas en ejecución.

La necesidad de verificar el tipo de las llamadas a funciones virtuales puede ser un factor de limitación para los diseñadores de bibliotecas. Por ejemplo, es útil poder definir una Pila de cualquier tipo de elementos, pero C++ no permite esto, aunque las combinaciones de patrones y herencia pueden aproximar la flexibilidad y facilidad de

diseño y de uso de las bibliotecas en lenguajes que se apoyan en la verificación dinámica de tipos (en ejecución).

Consideremos a modo de ejemplo un patrón de pila declarado de la siguiente forma:

```
template <class T> class Pila {
    T* p;
    int tamaño;
public:
    pila ( int );
    ~pila();

    void Push( T );
    T&  Pop();
};
```

Esta pila puede ser utilizada ahora como una pila de elementos de algún tipo A sin afectar a la verificación estática de tipos, de la siguiente forma:

```
pila < A* > pi(200);

void func(){
    ...
    pi.Push( new A1 );
    ...
    pi.Pop() ->mensaje;
    ...
}
```

El mecanismo para determinar si una operación se puede llevar a cabo o no con un objeto en tiempo de ejecución suele ser costoso si se compara con una llamada a función virtual en C++.

3.1.5 - Model / View / Controller

Las aplicaciones Windows son fáciles de utilizar y poseen una rica interfaz gráfica con el usuario, lo cual es provisto por una interfaz de programación de aplicaciones (API), compuesta por un conjunto de más de 600 funciones.

Entre las posibilidades ofrecidas por dichas funciones, podemos mencionar: independencia del dispositivo de visualización, una gran variedad de componentes de interfaz de usuario predefinidas (botones, menúes, cajas de diálogo, listas y ventanas de edición, entre otras), y la inclusión de una amplia interfaz a cualquier dispositivo gráfico, que permite dibujar gráficos y texto.

En particular, Borland C++ 4 provee una librería de clases llamada OWL que facilita el desarrollo de aplicaciones para Windows, proporcionando un completo marco para las aplicaciones.

Para facilitar la tarea de construir una aplicación, se recomienda la utilización de una arquitectura del tipo *Model-View-Controller*, la cual está presente en el lenguaje de progamación Smalltalk-80 y es adecuada para la construción de aplicaciones para Windows.

La arquitectura *Model-View-Controler* divide a las aplicaciones en tres capas separadas:

- *Model* se refiere a la *capa de aplicación*, donde residen todas los objetos dependientes de la aplicación.
- *View* es la *capa de presentación*, la cual presenta los datos de la aplicación al usuario. A su vez, esta capa provee también la interfaz gráfica con el usuario.
- *Controller* se refiere a la *capa de interacción,* la cual provee la interfaz entre los dispositivos de entrada y las capas Model y View.

La idea entonces es realizar el trabajo de separación de responsabilidades entre los objetos del sistema. Los detalles específicos de la aplicación estarán aislados de la interfaz del usuario, mientras que la interfaz del usuario a su vez, se encontrará dividida en dos partes: la presentación manejada por el View y la interacción por el Controller.

Cuando se trabaje con la librería de clases OWL, se utilizará una arquitectura MVC modificada donde existirá una clase correspondiente al *Model* y otra que se corresponderá con el par *View-Controller*. [9]

3.2 - Khoros

3.2.1 - Introducción

Khoros es un ambiente de desarrollo e integración de software que pone énfasis particular tanto en el procesamiento de la información, como en la exploración de datos.

El objetivo de este software es proveer un ambiente completo de desarrollo de aplicaciones que redefine el proceso de ingeniería de software con el fin de incluir a todos los miembros de un proyecto dado, abarcando desde el usuario final de aplicaciones hasta el programador de infraestructura. Khoros es entonces un sistema amplio que puede ser observado desde distintos puntos de vista, dependiendo de los objetivos perseguidos y de las necesidades científicas.

Para aquellos que necesitan soluciones a problemas científicos, de las cuales serán el usuario final, este software puede ser utilizado como base, ya que provee un basto conjunto de programas para procesamiento de información, así como también visualización y exploración de datos. Los operadores de manipulación de datos multidimensionales incluyen pointwise arithmetic, cálculos estadísticos, conversión de datos, histogramas, organización de datos y operadores de tamaño; también provee rutinas de procesamiento de imágenes y manipulación de matrices. Asimismo, los programas para visualización interactiva de datos incluyen un paquete para manipulación y visualización de imágenes, un programa para animación, un paquete para *ploteo* 2D/3D, una herramienta para modificación del mapeo de los colores así como también una aplicación para clasificación de imágenes/señales interactiva. Además ofrece diversas rutinas para procesamiento de datos con el fin de obtener una visualización 3D de los mismos, mediante una aplicación de software rendering.

Los operadores de Khoros están generalizados de manera que cada uno de ellos puede resolver problemas de un amplio rango de áreas específicas tales como imágenes médicas, remote sencing, control de procesos, procesamiento de señales y análisis numérico entre otras.

Todos los programas de procesamiento y visualización de la información se encuentran disponibles desde un ambiente de programación denominado Cantata, el cual es un lenguaje de programación visual, data flow, orientado a eventos con el fin de proveer un ambiente de programación dentro del sistema Khoros. *Data flow* es una aproximación "visible naturalmente" en la que se describe un programa visual mediante un grafo dirigido, en el cual cada nodo representa una función u operador y cada arco dirigido representa un camino sobre el cual fluyen los datos. El lenguaje visual provee un soporte para programadores tanto novatos como experimentados, dado que se basa en diagramas de bloque, los cuales resultan familiares para la mayoría de ellos. Cantata soporta procesamiento distribuido de *grano grueso,* y además puede manejar datos tanto en bloque como en cadenas (streams). Las características de jerarquía visual, iteración, control de flujo y los parámetros basados en expresiones hacen de Cantata una potente herramienta tanto de simulación como de prototipación.

Desde el punto de vista del desarrollo de aplicaciones, el sistema de cajas de herramientas para el programador de Khoros, provee un conjunto de servicios de programación, como así también herramientas para el desarrollo de software que

soportan la implementación completa de aplicaciones científicas y de nuevas ingenierías. Las aplicaciones escritas para Khoros pueden hacer uso de las mismas capacidades ofrecidas por las rutinas de visualización y procesamiento de datos del ambiente Khoros, entre ellas la posibilidad de acceso transparente a grandes conjuntos de datos distribuidos a través de una red, operar sobre diversos formatos de datos y archivos sin una conversión, soportando simultáneamente distintos widget sets, manteniendo una presentación consistente con una interfaz con el usuario estandarizada. Asimismo, el ambiente de desarrollo de software provee al usuario de una herramienta para diseño de la interfaz con el usuario gráfica, de manipulación directa utilizando la metología de programación por demostración, generación automática de código, documentación e interfaz con el usuario estandarizadas y manejo interactivo de configuración. Este sistemas de desarrollo puede ser también utilizado para integración de software, dado que los programas existentes pueden ser reunidos en un ambiente cohesivo, estandarizado y consistente.

Khoros provee un potente ambiente de trabajo para la comunidad científica mediante el desarrollo rápido de aplicaciones basadas en X-Windows, prototipación de soluciones a problemas complejos, utilizando los recursos de una red distribuida. La aproximación *por capas* y el concepto de *servicios de programación* proveen flexibilidad a los desarrolladores para la creación de aplicaciones complejas, mientras ocultan los detalles de los sistemas operativos y de los sistemas X-Windows.

Podemos entonces pensar a Khoros como un ambiente completo de desarrollo de software y exploración de datos, que reduce el tiempo utilizado en resolver problemas complejos y permitiendo a su vez compartir libremente ideas e información, promoviendo además la portabilidad del software.

3.2.2 - Servicios de Programación de Khoros

Los Servicios de programa de Khoros son un gran grupo de librerías organizadas por capas con el fin de proveer al desarrollador de software de una variedad de interfaces de programación que ofrecen un compromiso razonable entre complejidad reducida y control detallado. Cada servicio de programación tiene un nombre que indica su propósito principal en cuanto a la funcionalidad que proveen al desarrollador; cada servicio de programación está compuesto a su vez de una o más librerías distintas. Estos servicios pueden ser clasificados como Servicios de Datos, Servicios de Visualización e interfaz Gráfica del Usuario (GUI) y Servicios Fundamentales.

3.2.2.1 - Servicios Fundamentales

Bajo este nombre se encuentran englobadas nueve librerías pertenecientes a la caja de herramientas denominada *Bootstrap*. Los programas de aplicación desarrollados utilizando estos llamados a funciones obtendrán automáticamente la portabilidad provista por los Servicios Fundamentales. Asimismo, los progamas de Aplicación que utilicen estos servicios, obtendrán en forma transparente la capacidad de soportar una variedad de mecanismos de transporte de datos, inclusión de archivos así como uso de memoria compartida. Además, estos servicios proveen un conjunto de funciones

matemáticas, un parser de expresiones simbólicas y una gran variedad de funciones y utilidades comúnmente utilizadas.

Los servicios fundamentales constituyen una de las tres categorías principales en las que se encuentran divididas las librerías de Khoros. Los Servicios de Datos incluyen las librerías de la caja de herramientas DataServ; estas librerías se encuentran constituidas por un conjunto de rutinas para acceso y manipulación de datos. Los Servicios GUI y de Visualización, constituidos por las librerías incluidas en la caja de herramientas Design, proveen las rutinas para visualización de datos en forma de imágenes y gráficos. En conjunto los Servicios Fundamentales, Servicios de Datos y Servicios GUI y de Visualización, contienen todas las librerías que se distribuyen con el sistema principal de Khoros. Los Servicios de Programa se refieren al conjunto de los tres grupos de servicios antes mencionados.

Podemos decir que, en conjunto, los servicios fundamentales cumplen con aquellos requerimientos de la infraestructura del software Khoros que no incluya tareas tales como procesamiento o visualización de datos. Los Servicios Fundamentales incluyen los siguientes servicios de programa:

- **Servicios Básicos**

Los Servicios Básicos proveen al desarrollador de software un amplio rango de funciones y utilidades comúnmente empleados, incluyendo un gran número de funciones para manejo de alocación de memoria, manipulación de strings, parsing de strings así como también reporte de mensajes.

- **Servicios Matemáticos**

Estos servicios proveen implementaciones independientes de la arquitectura, de operaciones matemáticas comunes y ofrecen además una variedad de extensiones útiles a las funciones matemáticas estándar. Las rutinas incluidas en estos servicios fueron diseñadas de manera de obtener alta portabilidad y eficiencia.

- **Servicios de Expresión**

Estos servicios contienen un parser de expresiones simbólicas que pueden ser de utilidad en la evaluación de funciones y ecuaciones matemáticas.

- **Servicios de Sistema Operativo**

Estos servicios independizan al Khoros del Sistema Operativo y a su vez extienden las capacidades del Sistema Operativo dando un soporte para cálculo distribuido.

- **Servicios de Software**

Estos servicios fueron diseñados con el fin de proveer al programador de cajas de herramientas una visión coherente de los diversos componentes de un programa con el fin de maximizar la productividad durante la tarea de desarrollo de software, manejando al conjunto de archivos asociados a un programa , a una librería o a un script, como un *Objeto de Software.*

- **Servicios de Interfaz de Usuario**

Estos servicios proveen el soporte de bajo nivel necesario para mantener una interfaz con el usuario de comandos de línea (CLUI) de los programas Khoros. Los servicios de interfaz de Usuario manejan también la traducción de archivos de Especificación de Interfaz de Usuario (UIS) a programas CLUIs, generación automática de código para soporte tanto de CLUI como de Interfaz Gráfica de Usuario (GUI) de un programa, así como también generación automática de documentación.

3.2.2.2 - Servicios de Datos

Los Servicios de Datos constituyen un potente sistema para acceso y manipución de datos. El objetivo de estos servicios es proveer al programador de aplicaciones, la capacidad de acceder y operar sobre datos en forma independiente de su formato de archivo o sus características físicas tales como tamaño y tipo de datos. Los servicios de Datos fueron diseñados para satisfacer las necesidades de un gran número de dominios de aplicación abarcando procesamiento de imágenes y señales, visualización geométrica y análisis numérico.

Los servicios de datos de Interfaz de programación de aplicaciones (API) consisten de un conjunto de funciones de librerías simples que proveen acceso a un objeto de datos abstracto. Esta API permite almacenar y recuperar datos a partir de los objetos de datos y acceder a las características de los objetos sin tener que preocuparse por la complejidad de la estructura de datos o el complejo manejo de archivos. Estas APIs encapsulan una cantidad significativa de funcionalidades que manejan de manera eficiente las tediosas tareas de acceso y manipulación de datos. Esto libera al programador de las tareas de acceso a los datos y permite que se concentre sobre los detalles de implementación específicos de su problema.

Diversos y numerosos dominios de aplicación pueden utilizar los servicios de datos. Los datos son interpretados de acuerdo al modelo de datos dictado por el dominio de la aplicación. Los servicios de datos contiene una serie de modelos de dato disponibles, cada uno de los cuales fue diseñado con el fin de satisfacer las necesidades de un dominio simple o de una familia de ellos. El más potente de ellos es el *modelo de datos polimórfico*, el cual provee una interpretación consistente a través de diferentes dominios. Se encuentran también disponibles un modelo de datos geométrico y un modelo de datos de color.

Desde el punto de vista del bajo nivel, los servicios de datos constituyen un soporte para lectura y escritura sobre diversos formatos de archivo de datos, así como

también un sistema de manejo de memoria para el acceso a grandes conjuntos de datos. El sistema completo es construido en base a la abstracción de transporte provista por Khoros; los objetos de datos pueden ser accedidos en forma independiente del transporte subyacente, sin tener en cuenta si éste se realiza mediante archivos, pipes o memoria compartida. La funcionalidad provista por los servicios de datos permite escribir aplicaciones versátiles y robustas con un esfuerzo mínimo.

La importancia de los servicios de datos debe ser recalcada. Hay gran cantidad de energía desperdiciada en el desarrollo de algoritmos de procesamiento de datos que hacen cosas similares, pero que no pueden trabajar juntos debido a diferencias en las estructuras de datos y representaciones. Los servicios de datos proveen una solución para este problema.

3.2.2.3 - Servicios GUI y de Visualización

Estos servicios proveen todas las capacidades relacionadas con la visualización gráfica utilizando X Windows. Están contenidas aquí, todas las funciones necesarias para crear, cambiar interactivamente, y mantener una interfaz gráfica con el usuario.

Los servicios de GUI y de Visualización ofrecen también una cantidad importante de capacidades, incluyendo visualización y manipulación de imágenes, control de mapeo de colores, ploteo 2D y 3D, rendering de superficies y anotaciones.

A modo de resumen de los servicios de programación ofrecidos por las distintas librerías pertenecientes a Khoros, presentamos la siguiente tabla dividida en base a la clasificación de los diversos servicios.

3.2.3 - Herramientas de Programación

Las herramientas de programación incluyen facilidades para el programador visual, utilidades importantes ocultas para el usuario y herramientas CASE para el programador de toolboxes. Las herramientas de programación del toolbox central están resumidas en la siguiente tabla:

Nombre del programa	Descripción del programa
Phantomd	Phantomd es un utilitario que agiliza el cálculo distribuido. Es el responsable de la comunicación entre procesos de Khoros y del manejo de transporte remoto.
Craftsman	Los toolboxes de aplicaciones específicas son manejados mediante Craftsman. Éste permite abrir un objeto toolbox con el fin de acceder a sus objetos de programa.
Composer	Los objetos de programa son creados y manejados utilizando Composer. La herramienta de diseño GUI (Guise), los generadores de código, y los generadores de documentación son invocados desde aquí.

Cantata	Cantata es un lenguaje visual data flow avanzado que soporta procesos distribuidos de grano grueso, flujo y bloques de datos. Su jerarquía visual, iteración, control de flujo, y parámetros basados en expresión hacen que sea un sistema de prototipación y simulación poderoso. Un programador puede crear un panel frontal para un programa visual, con el fin de encapsular la complejidad del diagrama de flujo de datos y distribuirlo como un sistema de producción.
Guise	Guise es una herramienta de diseño GUI interactiva. Opera interpretando y modificando la especificación de interfaces de usuario (UIS) o la representación abstracta de un GUI. En cualquier nivel del desarrollo de GUI, el código en forma de un framework de aplicación puede ser generado automáticamente desde un archivo UIS. Aún cuando la aplicación ha sido compilada, el programador es capaz de editar los objetos GUI interactivamente en tiempo de ejecución porque todas las aplicaciones GUI en Khoros están ubicados en el tope de la librería widget abstracta.
Concert	Cualquier aplicación escrita utilizando los servicios de GUI, soporta la interacción de aplicaciones múltiples de usuario final en lugares distribuidos geográficamente. La interfaz de usuario distribuida de una sesión cooperativa es manejada por el programa Concert el cual provee groupware para grupos de trabajo.

3.2.3.1. El sistema de programación visual

El sistema de programación visual Khoros, consiste de la integración de un lenguaje visual data flow con una herramienta de diseño GUI (interfaz de usuario gráfica). El nombre de la interfaz visual para esta capacidad es Cantata. Es importante notar que Cantata es un editor de diagramas data flow. El programador visual escribe un programa conectando operadores para formar un grafo data flow; los operadores son seleccionados de los toolboxes de aplicaciones específicas. El programador visual puede encapsular un grafo data flow en una aplicación para un usuario final, en la forma de panel frontal, usando la herramienta de diseño integrada de GUI. Si el operador deseado no se encuentra disponible, entonces el programador puede, crear sus nuevos operadores utilizando las herramientas de desarrollo de software Craftsman y Composer.

3.2.3.2 - El Lenguaje Visual Cantata

Cantata es un lenguaje de programación de propósito general basado en un paradigma data flow con soporte adicional para condicionales, iteración y procedimientos. Los términos "data flow" son utilizados vagamente aquí, dado que comunican la idea básica que hay detrás del lenguaje visual Cantata, a pesar de no satisfacer todas sus características. Por ejemplo, Cantata incluye operadores de control

de flujo *while* y *for* que no están disponibles en los sistemas data flow estrictos. Proveen herramientas de programación poderosas, pero rompen el paradigma data flow estricto ya que tiene caminos explícitos e implícitos para el flujo de los datos.

El subsistema de ejecución Cantata interpreta el grafo data flow dinámicamente para administrar los operadores y luego despacharlos como procesos. Los operadores son considerados de grano grueso porque se corresponden con procesos enteros, no con segmentos de código o procedimientos. El usuario puede seleccionar el scheduling de manejo o scheduling de demanda de operadores dependiendo de la respuesta deseada y de su estilo de programación preferido.

Los operadores son ejecutados local o remotamente para hacer uso eficiente de una red heterogénea de máquinas. Cantata utiliza a Phantomd para negociar el transporte remoto de datos y depositar procesos en máquinas remotas. El programador visual asigna operadores a máquinas específicas interactivamente, ya sea para optimizar la velocidad de ejecución o utilizar hardware específico. Las máquinas remotas no necesitan las instalaciones completas de Khoros, sólo se requiere que se encuentre en ejecución Phantomd.

Los dominios específicos de aplicación de procesamiento de imagen o visualización geométrica típicamente procesan datos como bloques; sin embargo, los dominios de telecomunicaciones y control de procesos trabajan los datos como flujo. Los operadores de Cantata pueden consumir y producir datos ya sea en bloques o en flujos, en donde un bloque es pensado típicamente como datos de grano grueso y el flujo es pensado como datos de grano fino.

El lenguaje visual Cantata extiende el paradigma básico data flow para hacer más poderoso al ambiente de simulación y prototipación de aplicaciones. El flujo de control y de datos es provisto por operadores de control como *if-else, while, count* o *for* y *trigger*. Los procedimientos visuales se encuentran disponibles para programador con el fin de soportar el desarrollo de grafos data flow jerárquicos con cientos de operadores. Los parámetros de los operadores son inicializados interactivamente por el usuario o calculados en tiempo de ejecución mediante expresiones matemáticas sujetas a valores de datos o variables de control.

Es importante notar que los operadores no necesitan ser compatibles con el modelo de datos de Khoros o linkeados con librerías Khoros para ser utilizado en Cantata. Las conexiones de la red data flow no imponen ninguna estructura de datos o transporte específicos. En resumen, Cantata es utilizado por el programador visual para desarrollar interactivamente aplicaciones complejas distribuidas e interactivas, combinando operadores de datos y operadores de visualización desde los toolboxes provistos, en un nuevo programa.

3.2.3.3 - La herramienta de diseño GUI de Cantata

El programador visual puede usar la herramienta de diseño GUI integrada de Cantata para empaquetar un programa visual en una aplicación de visualización de datos y de procesamiento de información. El programador visual tiene acceso completo a todos los parámetros del operador, a las conexiones del operador y a las características del lenguaje visual. Estas características de flexibilidad y accesibilidad presentan demasiada complejidad y requieren mucho aprendizaje para el usuario final, en aquellos

casos donde el programa visual deba ser efectivo para trabajo de producción. La herramienta de diseño GUI puede ser utilizada para diseñar un GUI alternativo y contenido en sí mismo para el programa visual que permita sólo interacción limitada con los parámetros de los operadores.

Los parámetros de los operadores son accedidos en Cantata "abriendo" el ícono de operador o Glyph, y luego inicializando valores en forma interactiva, en los objetos GUI. Un programa visual con 100 operadores puede tener más de 400 objetos GUI, cada uno de los cuales afecta tanto a los resultados como a la operación del programa. El programador visual construye un GUI alternativo al programa, seleccionando algunos de los más importantes objetos GUI y luego ubicándolos en el panel frontal. El panel frontal puede ser pensado como una interfaz de usuario personalizada y simplificada para el programa visual. Una vez que el panel frontal está finalizado, se le da un nombre para que pueda ser distribuido como una aplicación separada.

Es importante notar que el panel frontal GUI es interpretado, no se necesita compilación para correr o distribuir el panel frontal como una aplicación separada. También el usuario de aplicación final, puede seleccionar o editar el GUI de acuerdo a sus necesidades o gustos sin que afecte al programa visual. El programa visual original puede ser siempre accedido para futura exploración y control más detallado sobre la aplicación.

3.2.3.4 - Herramientas de Desarrollo de Software

Un programador de toolbox puede hacer uso de las herramientas de desarrollo de software provistas, con el fin de crear operadores para el lenguaje visual o desarrollar aplicaciones de usuario final que sean completamente independientes del lenguaje visual. Las herramientas de desarrollo de software de Khoros se integran con otras herramientas CASE, comúnmente utilizadas con el fin de proveer un ambiente de desarrollo de software completo. Las herramientas Craftsman y Composer soportan el proceso iterativo de desarrollo, permitiendo distribuir y compartir software en el contexto de Khoros. A su vez, estas herramientas mejoran la productividad actuando como asistentes del programador, proveyendo automatización donde es posible, y reforzando la consistencia cuando sea necesario. Además, ocultan la complejidad tanto de la configuración fuente subyacente, de los generadores de código, como así también de los formateadores de documentación.

- **El manejador de toolbox Craftsman**

Un toolbox es una colección de programas que son manejados independientemente del resto del software del grupo de trabajo. Un grupo de trabajo es probable que esté creando y manteniendo simultáneamente varios toolboxes, o puede estar preparando un toolbox para distribución. El programador de toolbox maneja estas actividades mediante Craftsman.

Un usuario Craftsman realiza operaciones sobre un toolbox tales como abrir, cerrar, empaquetar, desempaquetar, borrar, mover, instalar y copiar. La metodología de toolbox para manejar software permite al grupo de trabajo repartir responsabilidades y eliminar dependencias.

- **El manejador de programa Composer**

Un toolbox está hecho de uno o más objetos de programa, donde un objeto de programa puede ser pensado como composición de código fuente, especificaciones de interfaces de usuario (UIS), documentación y archivos de configuración de un operador ejecutable simple o aplicación. Cada componente del objeto de programa requiere clases de operaciones substancialmente diferentes. A modo de ejemplo podemos considerar que las operaciones sobre código incluyen generación, compilación, depuración e instalación, mientras que las operaciones sobre UIS incluyen creación y edición.

Composer provee al programador de toolboxes acceso conveniente a todos los componentes de objetos de programa, y puede invocar a todas las operaciones necesarias para crear, editar y manejar objetos de programa. Asimismo, maneja las dependencias entre los componentes a medida que el programador de toolbox progresa a través del ciclo de vida del software de un objeto de programa.

Una de las capacidades más importantes de Composer es la generación automática de código de interfaces de usuario de línea de comando y gráficas (CLUIs y GUIs). Guise es invocado por el programador de toolboxes desde el Composer, y luego es utilizado con el fin de diseñar interactivamente una GUI. La GUI es almacenada en forma de UIS, la cual a su vez es utilizada como entrada de los generadores de código.

- **La herramienta de diseño GUI, Guise**

Esta herramienta es utilizada en aquellos casos donde se desee crear e interactivamente diseñar una GUI para operadores de procesamiento de información o para operadores de visualización de datos. Guise provee al programador de toolboxes un menú de objetos GUI que pueden ser ubicados interactivamente en una interfaz de usuario. Una vez presente en la UI, todos los atributos de un objeto GUI pueden ser editados directa e interactivamente. Una importante capacidad de la combinación de Guise y del generador de código, es que todos los aspectos de la interfaz de usuario están separados de la funcionalidad de la aplicación. Por ello, el desarrollo interactivo es soportado a través del ciclo de vida completo de la aplicación.

La GUI de un operador de visualización de datos es relativamente complejo comparándolo con la GUI de un operador de procesamiento de información. Por esta razón, la especificación de interfaces de usuario (UIS) creada por Guise para el operador de visualización, es utilizada como entrada de un generador de código GUI para crear un framework de aplicación. Por el otro lado, la UIS creada por Guise para un operador de procesamiento de información es interpretada en tiempo de ejecución. En cualquier caso, la UIS es también usada como entrada de un generador de código de interfaces de usuario de línea de comando.

- **El sistema de grupo (groupware) Concert**

Concert permite a cualquier aplicación Khoros construida a través de servicios de interfaz de usuario, operar como un groupware. La capacidad de computación distribuida de los servicios fundamentales y la capacidad de grabar eventos de interfaces de usuario provista por los servicios de interfaces de usuarios, son combinadas para producir la funcionalidad del programa Concert. Cuando Concert es utilizado con el fin

de invocar a un sistema de visualización de datos u operador, o una de las herramientas de programación, replica el proceso a una red de máquinas y maneja las comunicaciones entre procesos. El resultado es la interacción simultánea de múltiples usuarios finales en lugares distribuidos geográficamente.

Existen numerosos grupos desarrollando toolboxes en áreas de aplicación como redes neuronales, control de procesos, telecomunicaciones, control de instrumentos, imágenes médicas, restauración de imágenes, reconocimientos de patrones, morfología, análisis de elemento finito, simulación y modelación.

3.3 - Evaluación Comparativa de las herramientas de desarrollo

Tal vez resulten ingratos los resultados de la comparación entre estas dos herramientas de programación, dado que una de ellas es de propósito general mientras que la otra fue diseñada específicamente para el procesamiento de imágenes/señales. Sin embargo, creemos conveniente efectuar esta comparación ya que una porción de nuestro trabajo involucró el desarrollo de una aplicación. Parte de esta evaluación y las implementaciones involucradas fueron el centro del trabajo referido en [14].

3.3.1 - Experiencia de desarrollo en C++

De acuerdo a lo recomendado por la bibliografía, con el fin de facilitar la tarea de desarrollo y respetando el estilo de programación propuesto por el lenguaje, organizamos nuestra aplicación en base a una arquitectura Model-View-Controller (MVC) tomando ventaja asimismo de las facilidades ofrecidas por la programación orientada a objetos soportada por el lenguaje.

A continuación describimos las distintas tareas llevadas a cabo, organizadas en función de la arquitectura MVC antes mencionada.

Model

En base a la aproximación elegida, con el fin de definir los objetos que finalmente constituirían esta capa del sistema, fue necesario diseñar e implementar las diversas clases que luego intervendrían en la aplicación, entre las cuales se destaca aquella que denominamos Imagen. De las funcionalidades que provee esta clase en cuanto al tratamiento de las imágenes involucradas en nuestra aplicación, podemos señalar las que resultan más relevantes, entre ellas:

- Diseño e implementación de la estructura y métodos para almacenamiento y manipulación de datos de las imágenes, así como también de los resultados intermedios y finales obtenidos durante el procesamiento.

- Tratamiento de los diferentes formatos estándar para almacenamiento de imágenes como son BMP, TGA, TIF y PCX. La tarea realizada aquí consiste básicamente de la lectura de los datos de imagen dados en alguno de los formatos anteriormente mencionados, así como su conversión y posterior almacenamiento en la estructura de datos diseñada especialmente a tal fin.

- Implementación de cada uno de los algoritmos para extracción de características mediante el cálculo de las diferentes métricas que utilizamos con el fin de distinguir a las diversas texturas. Entre las cuales podemos mencionar: Media, Varianza, Dispersión, Skewness y Kurtosis, que representan características de histograma de primer orden, tomadas en base a vecindades sobre cada imagen.

- Diseño e Implementación del algoritmo de clasificación que permita determinar la membrecía de cada pixel respecto de las posibles clases de textura especificadas. En nuestra aplicación en particular utilizamos un método ad-hoc de clasificación, dadas las restricciones iniciales en cuanto a la obtención de los ejemplos para entrenamiento del

clasificador. Este método permite ingresar tanto ejemplos *positivos* (representantes de la textura a identificar), como *negativos (representantes de la clase Reject).* El comportamiento global del clasificador es similar a los clasificadores de Distancia Euclídea Mínima, con el agregado de una restricción adicional: un margen para la distancia mínima aceptable al ejemplo positivo más cercano. En la aproximación utilizada se tienen en cuenta sólo dos clases. La primera de ellas corresponde a la textura que se intenta identificar mientras que la segunda corresponde a la clase *Reject* o descarte.

- Implementación de una clase para los puntos seleccionados como ejemplos positivos y negativos. Esto involucra la adquisición y el almacenamiento de los mismos.

View-Controller

De acuerdo a lo mencionado en secciones anteriores, cabe señalar que en este fragmento de la aplicación, se encuentran combinados los diversos objetos que permitirán la interacción entre los dispositivos de entrada, tales como teclado y mouse, y las capas Model y View, así como la presentación al usuario de los datos de interés, ya sea en forma gráfica, numérica o textual. Para ello fue necesario diseñar e implementar las clases que provean las siguientes funcionalidades:

- Mapeo o normalización de los resultados de los algoritmos para extracción de características con el fin de presentarlos visualmente en forma de imagen, lo cual permitió la realización de dos tareas relevantes en cuanto a la elección y obtención de las distintas medidas propuestas: depuración de los algoritmos implementados, así como también una primer evaluación en cuanto al poder de discriminación alcanzado.

- Interfaz con el usuario, para la obtención de los diversos parámetros necesarios en la ejecución de los diferentes algoritmos. Cada uno de los algoritmos para obtención de medidas, como así también el método de clasificación, deben ser alimentados inicialmente con parámetros tales como el tamaño de ventana a ser tenido en cuenta para la extracción de las características, la distancia mínima aceptable al ejemplo positivo más cercano, así como también las métricas a tener en cuenta en un proceso de clasificación dado.

- Manejo de ventanas y menúes: Si bien el ambiente de programación provee herramientas que facilitan el diseño de las ventanas y los menúes que serán utilizados por la aplicación, la funcionalidad asociada a cada ítem de menú debe ser obtenida relacionándolos en forma explícita mediante código, con los métodos de los objetos correspondientes que la provean. Análogamente, las ventanas que permitan la interacción con el usuario, deberán ser asociadas con los objetos que demanden los resultados de dicha interacción.

3.3.2 - Experiencia de desarrollo en el ambiente Khoros

Durante el desarrollo de la aplicación basada en este ambiente, dividimos la tarea en tres módulos principales:

El primero de ellos se encargó de realizar la *entrada* de las imágenes con el fin de presentarlos al segundo, el cual efectuó la *extracción de las características*, cuya salida

sería finalmente presentada al módulo de *clasificación*, el cual a su vez calcula y presenta los resultados finales del procesamiento.

A continuación detallamos para cada uno de estos tres módulos, qué operadores fueron utilizados, así como también los operadores implementados en aquellos casos donde corresponda.

- Entrada: En este módulo se concentran los diversos operadores utilizados en la entrada de imágenes. Dicho ingreso de datos fue realizado de dos maneras distintas, que a continuación se describen.

  - *User Defined:* Este operador permite al usuario definir sobre qué archivo se ha de realizar entrada/salida.
  - *Import Raw*: Este operador se utiliza para leer datos binarios raw e importarlos a un segmento de datos de la estructura estandar de Khoros, basada en el modelo de datos polimórfico. En nuestra aplicación este operador es utilizado tomando la salida del anterior.
  - *Extract:* Este operador permite realizar la extracción de una región rectangular de un objeto. En nuestro caso, es utilizado en combinación con el operador Inset.
  - *Inset:* Este operador realiza la inserción de un objeto dentro de otro. La utilización que dimos a este operador en cuanto a la entrada de las imágenes, fue la confección de imágenes de testeo compuestas por diversas imágenes conteniendo cada una de ellas una única textura bien definida.

- Extracción de Características: Tal vez aquí se concentra la parte principal del sistema. En este módulo desarrollamos los operadores específicos utilizados en nuestra aplicación, ellos son: Media, Dispersión, Varianza, Skewness, Kurtosis y Suavizado HS, los cuales fueron utilizados en combinación con algunos operadores de Khoros, como así también combinados entre ellos mismos. Describimos a continuación los operadores utilizados y/o implementados, así como la funcionalidad provista por cada uno de ellos.

  - *Convert Type:* Este operador convierte el tipo de datos de almacenamiento utilizado por un objeto. Realizamos la conversión del tipo de los datos de entrada con el fin de evitar la pérdida de precisión durante el cálculo de las operaciones matemáticas involucradas en los pasos posteriores.
  - *Media:* Este operador permite realizar el cálculo de la media para cada pixel de una imagen, en base a su vecindad cuyo tamaño es especificado por el usuario. Su resultado, como el de los operadores subsiguientes, puede ser pensado como una nueva imagen, a la que podríamos llamar imagen de media.
  - *Dispersión:* Este operador permite realizar el cálculo de la dispersión para cada pixel de una imagen, en base a una vecindad de tamaño variable.
  - *Varianza:* Este operador permite realizar el cálculo de la varianza para cada pixel de una imagen, en base a ventanas de tamaño variable.
  - *Skewness:* Este operador permite realizar el cálculo del sesgo para cada pixel de una imagen, en base a vecindades de tamaño variable.

- *Kurtosis:* Este operador permite realizar el cálculo de la curtosis para cada pixel de una imagen, normalizado con el valor de varianza al cuadrado, en base a una vecindad de tamaño variable.
- *Suavizado HS:* Este operador realiza el suavizado referido en la sección 2.3.1.4 de este trabajo, permitiendo la especificación del tamaño de ventana a utilizar.
- *Características de Dispersión:* Calcula las medidas de Media y Dispersión de la imagen de dispersión obtenida con el operador correspondiente. La salida de este operador entonces consta de dos imágenes, una representando la imagen de media de la dispersión y la otra representando la dispersión de la imagen de dispersión, todas ellas mejoradas con el suavizado HS. Estas dos últimas imágenes serán entregadas al clasificador final, entre otras.
- *Características de Varianza:* Este operador calcula las características de media y dispersión para cada uno de los pixels de la imagen de Varianza, mejoradas con el suavizado HS, en forma análoga al anterior.
- *Características de Skewness:* Se calculan aquí las características de media y dispersión para cada elemento de la imagen de Skewness, mejorados con el Suavizado HS.
- *Características de Kurtosis:* Se calculan aquí las características de media y dispersión para cada elemento de la imagen de Kurtosis, mejorados también con el Suavizado HS.
- *Append:* Este operador permite realizar la combinación de las imágenes de características obtenidas a partir de los operadores previamente descriptos, en un único objeto, dispuestas a través del segmento *element.*
- *Operadores accesorios:* Si bien en su mayoría estos operadores no conforman el conjunto utilizado en la aplicación final, prestaron una utilidad fundamental en la etapa de desarrollo, durante la evaluación de los diversos operadores utilizados y/o implementados facilitando la exploración de diversos datos, tanto para este como para los otros módulos involucrados. Entre los operadores accesorios utilizados durante esta etapa podemos mencionar: Data Object Info, en combinación con Convert Type; Statistics, en conjunto con File Viewer; Histogram junto a 2D Plot, como así también Add, Display Image y Animate.

• Clasificación: En este módulo es donde se realiza la segmentación final de la imagen de entrada, clasificando a cada uno de los pixels de acuerdo a un criterio de distancia mínima respecto de los distintos ejemplos de textura provistos. Los operadores utilizados para la realización de este módulo fueron los siguientes:

- *Minimum distance:* este operador implementa un clasificador de distancia mínima Euclídea, el cual toma como parámetros por un lado la imagen a ser clasificada obtenida de la salida del Append y un conjunto de prototipos para cada clase posible.
- *Piecewise linear, Extract e Inset:* También durante el desarrollo de este módulo se utilizaron ciertos operadores que permitieron la extracción de los diversos prototipos representativos de cada tipo posible de textura. Los operadores Extract

e Inset aquí permitieron extraer y combinar en un único objeto, los diversos prototipos, mientras que el operador Piecewise Linear hizo posible el etiquetado de los distintos prototipos.

- *Operadores accesorios:* Otros operadores que fueron utilizados durante el desarrollo de este módulo fueron el Print Data, el cual imprime los segmentos de datos seleccionados de un objeto, en formato ASCII, en un archivo o por pantalla, como también el operador File Viewer

3.3.3 - Conclusiones de la Comparación entre las Experiencias de Desarrollo

Como fue mencionado anteriormente, las aplicaciones Windows son fáciles de utilizar y poseen una rica interfaz con el usuario pero, desafortunadamente para los desarrolladores de software, esta facilidad de uso se alcanza a expensas de una compleja interfaz de programación de aplicaciones (API). Podríamos decir entonces que la gran cantidad de funciones provistas por el sistema operativo y la complejidad de su utilización, presentan tantas facilidades como inconvenientes durante la tarea de desarrollo de software en el caso de utilizar C++. Por el contrario, estas facilidades son alcanzadas en forma natural y sencilla cuando se utiliza el ambiente de desarrollo Khoros.

Una de las grandes ventajas del Khoros, es que provee una solución optimizada para los problemas de representación y manipulación de los datos de imagenes/señales, mediante la utilización del modelo de datos polimórfico. Por otra parte en C++ cada programador resuelve estos problemas de acuerdo a su conveniencia, dificultando asi la compatibilización de desarrollos realizados por diferentes personas o grupos.

En cuanto a la reusabilidad provista por ambos ambientes, probablemente la metodología de programación orientada a objetos posea desde el punto de vista teórico mediante la utilización de clases y mecanismos de herencia, un mayor potencial en la reusabilidad. A pesar de esto y desde un punto de vista pragmático, el gran conjunto de problemas comunes a las aplicaciones de procesamiento de imágenes ya resueltos en Khoros, conforma una importante base que facilita en gran medida, la tarea de desarrollo mediante la reutilización de operadores provistos y/o implementados por uno mismo, dado que cada operador desarrollado puede quedar disponible en el ambiente para ser reutilizado en un futuro. Además, la inclusión de operadores desarrollados por otros usuarios del ambiente puede realizarse en forma inmediata.

Para nuestro caso particular, los problemas que pudimos resolver simplemente haciendo uso de las facilidades provistas por Khoros podemos mencionar: representación y manipulación optimizadas de los datos de imagen mediante la utilización del modelo de datos polimórfico, visualización de imágenes y resultados de procesamiento, cálculo y visualización de histogramas, clasificación, así como también la obtención de los parámetros para extracción de características.

Cabe señalar que, si bien la estructura utilizada y los métodos implementados durante el desarrollo de la aplicación C++ son adecuados y alcanzan para satisfacer los requerimientos del problema, éstos no son suficientemente generales como para ser

utilizados en cualquier aplicación de procesamiento de imágenes sin sufrir alguna modificación previa.

Probablemente una de las sensaciones más fuertes que tuvimos al trabajar sobre el ambiente de desarrollo Khoros, es que nos resultó notoriamente mayor la cantidad de operadores que *utilizamos* respecto de los que necesitamos *implementar*, es decir que pudimos concentrarnos más en los problemas específicos planteados por la aplicación, y no tanto en aquellos de índole general en el procesamiento de imágenes. Por el contrario, durante el desarrollo en C++, la tarea que se destaca es la codificación y no la utilización de soluciones. Un ejemplo está dado por la confección de las imágenes de prueba: en Khoros, estas imágenes las compusimos utilizando operadores provistos por el ambiente como son Extract e Inset, mientra que para obtenerlas para la applicación desarrollada en C++, hubo que escribir el código necesario o bien utilizar aplicaciones comerciales. Resulta obvio que cada nueva funcionalidad que se pretenda agregar debe ser al menos diseñada, codificada y depurada, lo cual dependiendo de la complejidad de la tarea, puede insumir un tiempo considerable.

El desarrollo en Khoros es mayoritariamente visual, mientras que en C++ sólo algunas partes de la interfaz pueden realizarse utilizando esta metodología. Como fue mencionado, las acciones a tomar asociadas a un botón por ejemplo, deben ser especificadas mediante código en el caso de C++, mientras que en Khoros la forma de realizar este trabajo resulta mucho más natural, dadas las facilidades de programación visual.

Por otra parte, la escritura de código es similar en los dos, ya que en Khoros se utiliza también el lenguaje C. Sin embargo la cantidad de código que debe ser escrito en este ambiente es menor comparado con C++, dado que una buena parte de él es creada a partir del generador de código que posee.

A modo de conclusión podríamos decir entonces que en la comparación entre las dos experiencias, resultó más sencillo el trabajo en el ambiente Khoros debido a la facilidad en su aprendizaje, la naturalidad en el trabajo y fundamentalmente a que no fue necesario resolver problemas de índole general como lo son la interfaz con el usuario, la interfaz con el sistema operativo así como todos aquellos solucionados por el mismo ambiente Khoros en cuanto al procesamiento y exploración de datos, lo que finalmente nos permitió concentrarnos casi exclusivamente en la resolución de los problemas específicos planteados por nuestros requerimientos.

Asimismo, entre los objetivos de nuestro trabajo se encontraba la idea de realizar una herramienta de software que pueda ser utilizada por los usuarios expertos de la DAIS. En esta Dirección se trabaja sobre plataformas Microsoft Windows, Windows '95 y Windows NT, lo cual descartaba al Khoros como ambiente para el desarrollo de la aplicación final. Entre los lenguajes disponibles, creímos que el C++ era una buena opción ya que provee un buen soporte para la Programación Orientada a Objetos, manteniendo una gran performance en cuanto al tiempo de procesamiento requerido.

CAPÍTULO 4

4 - Métricas: Estudio y Utilización

4.1 - Evaluación

Dadas las características de aleatoriedad de las texturas naturales observadas en las imágenes a tener en cuenta para la clasificación final, resultan adecuadas aquellas métricas que intenten capturar la información de distribución de valores de gris en términos estadísticos. [6][1]

Existen diversos métodos que permiten estudiar la similitud de texturas aleatorias, entre ellos podemos mencionar métricas de Laws, Densidad de bordes, Densidad de Extremos relativos, Características de histogramas de primero y segundo orden, entre otras.

Como es sabido, los histogramas entregan información de las distribución de los valores de gris dentro de la región bajo observación. Las características de histograma de primer orden resultan particularmente interesantes para el estudio, dados su costo relativamente bajo en cuanto a tiempo y su simplicidad algorítmica, sólo comparables a las métricas de Laws. Estas últimas poseen una restricción adicional: están definidas con un tamaño de ventana fijo, lo cual le quita la flexibilidad necesaria para el análisis de las imágenes involucradas en nuestro trabajo.

En base a estas consideraciones decidimos evaluar el alcance de estas métricas en cuanto a la discriminación basada en características de textura, lo que constituye el objetivo central de nuestro trabajo.

El requisito obvio que deben cumplir las métricas a tener en cuenta es que dadas texturas distintas, se obtengan valores distintos y dadas texturas iguales, se obtengan medidas aproximadamente iguales.

Puede observarse a modo de ejemplo que los histogramas pertenecientes a porciones de la imagen exhibiendo texturas distintas, presentan formas bien distinguibles, lo que hace suponer que las mediciones que se tomen sobre dichos histogramas arrojarán también valores distinguibles.

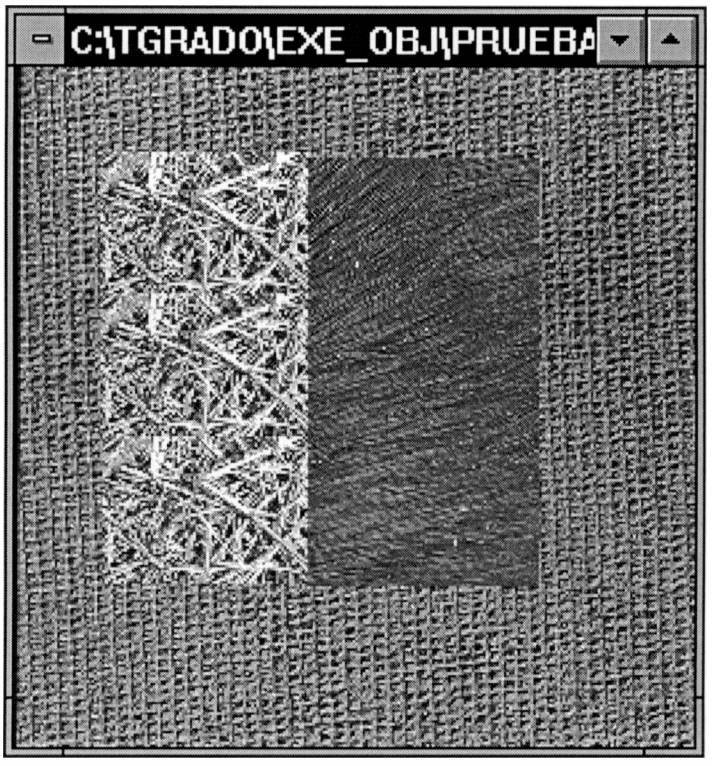
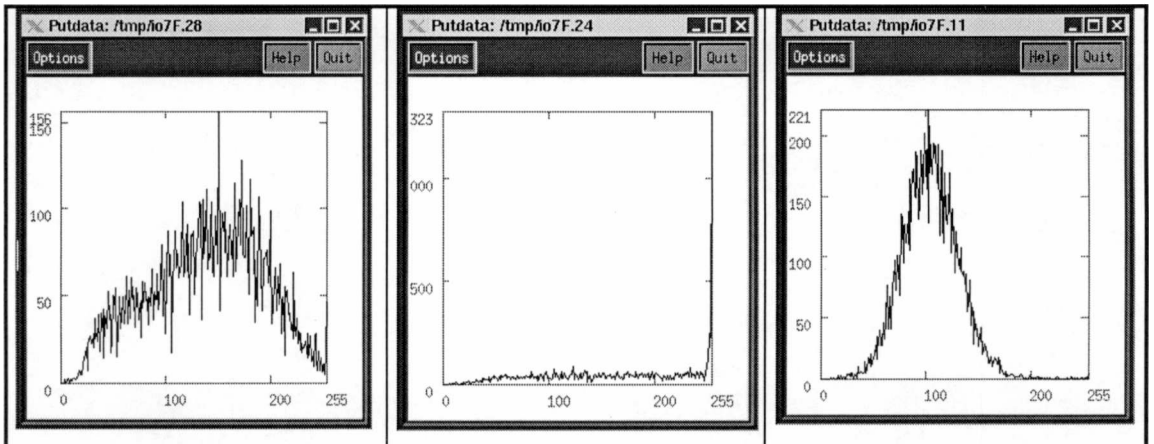


Figura 1



Arpillera	Pasto	Pelo
Dispersión = 44,762	Dispersión = 63,228	Dispersión = 19,804
Varianza = 2861,957	Varianza = 5212,520	Varianza = 653,098
Skewness = 73,972	Skewness = 93,100	Skewness = 43,230
Kurtosis = 2,167	Kurtosis = 1,898	Kurtosis = 4,012

Las medidas tomadas sobre los histogramas calculados son Dispersión, Varianza, Skewness y Kurtosis. Estas métricas proveen información acerca de la variación de los valores de gris de la región de la imagen analizada.

Las medidas son tomadas de acuerdo a las siguientes fórmulas, donde las sumas se realizan sobre cada pixel de la vecindad x_i., $i \in [\ 1..n\ ]$

- Dispersión : $D = \Sigma|x_i\text{-}M| / (n\text{-}1)$
- Varianza : $V = \Sigma(x_i\text{-}M)^2 / (n\text{-}1)$
- Skewness : $S = \Sigma|x_i\text{-}M|^3 / ((n\text{-}1)\ V)$
- Kurtosis : $K = \Sigma(x_i\text{-}M)^4 / ((n\text{-}1)\ V^2)$

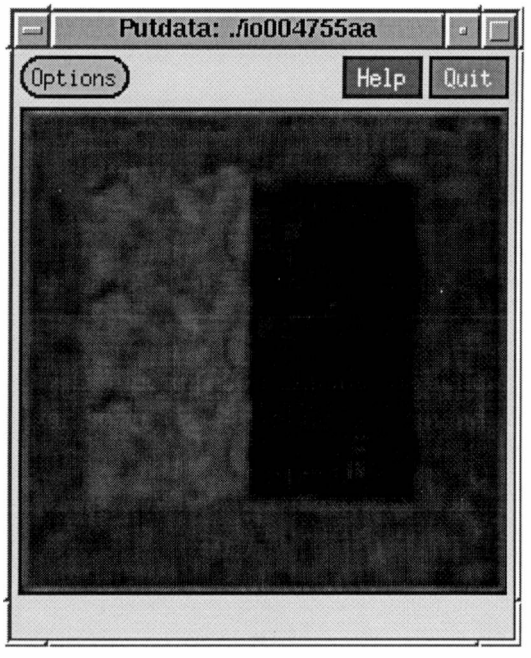
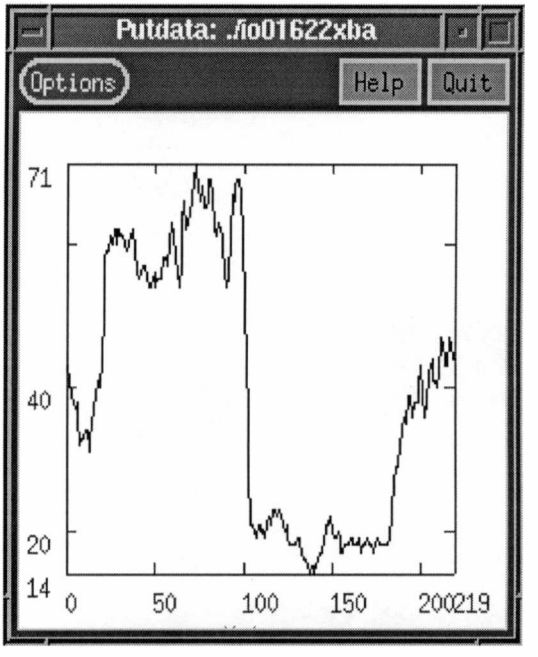


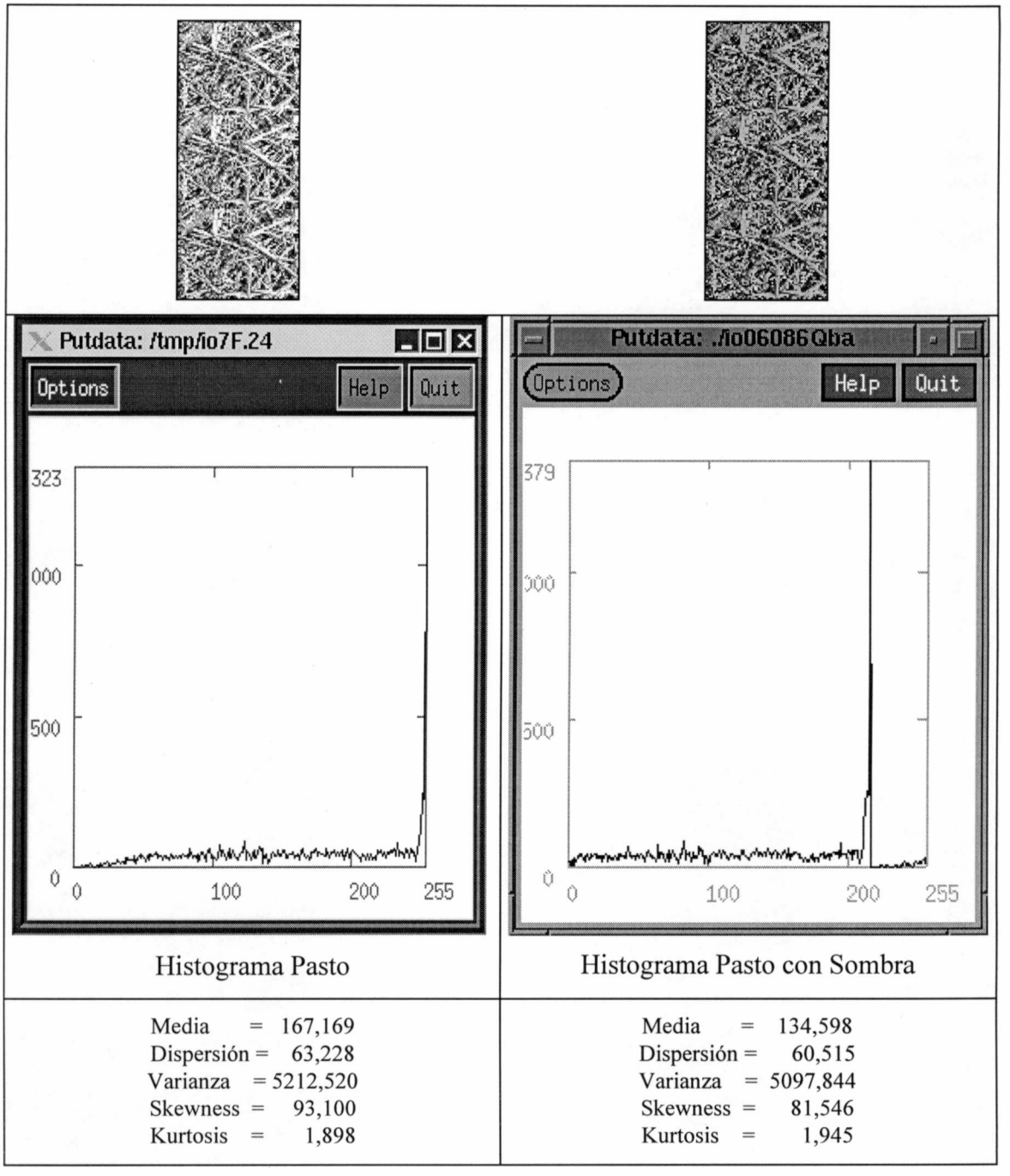
Imagen de Característica



Corte Transversal

En la figura anterior puede apreciarse la imagen resultante al tomar la Dispersión de la imagen representada en la figura 1 en base a vecindades sobre cada pixel y un corte transversal ejemplificando los valores alcanzados.

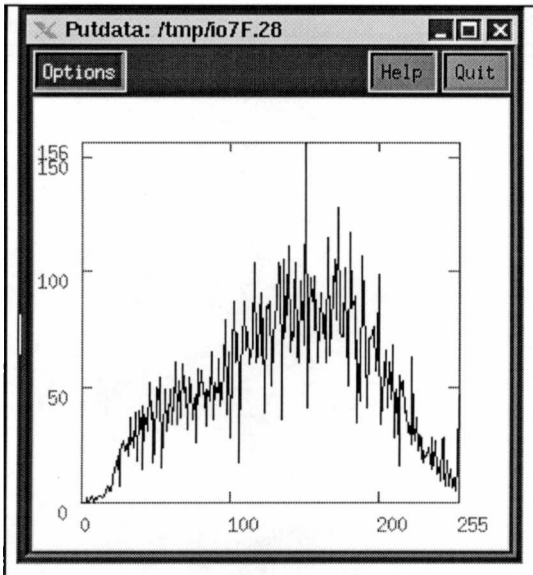
Una atributo relevante en el reconocimiento en base a características de textura es que las medidas a tener en cuenta no deben depender de la amplitud media de los grises en una región dada. En efecto, la textura de una imagen de una zona de bosque no varía significativamente si por ejemplo, una nube produce sombra sobre ella. Las medidas que se tomen deben cumplir con este requisito, es decir que no deberían variar bajo circunstancias tales. Como puede observarse en los ejemplos siguientes, a pesar de poseer valor medio distinto, ambas regiones conservan valores similares en las otras medidas utilizadas.



Histograma Pasto	Histograma Pasto con Sombra
Media = 167,169 Dispersión = 63,228 Varianza = 5212,520 Skewness = 93,100 Kurtosis = 1,898	Media = 134,598 Dispersión = 60,515 Varianza = 5097,844 Skewness = 81,546 Kurtosis = 1,945

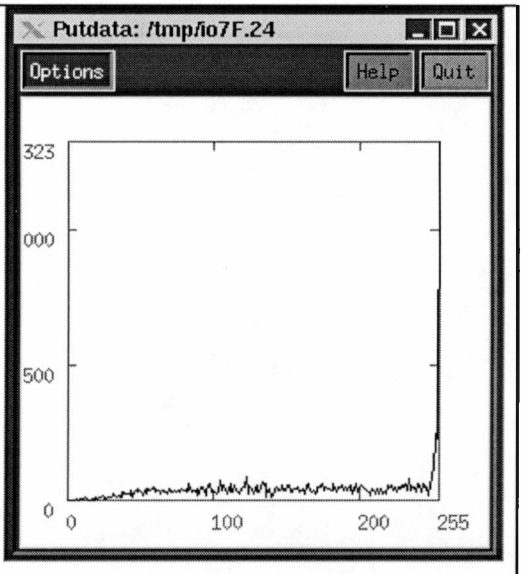
De acuerdo al método elegido para clasificación, cada pixel deberá ser analizado y clasificado por separado. Para esto es necesario evaluar una vecindad del pixel en cuestión con el fin de obtener un histograma de donde extraer las características deseadas. Se define entonces una pequeña ventana centrada en el pixel de interés que determinará la cantidad de pixels que intervendrán finalmente en las medidas tenidas en cuenta.

La performance final de la clasificación resulta sensible a la elección correcta del tamaño de dicha ventana. Si se tiene en cuenta una vecindad pequeña, puede perderse información relevante en la discriminación. En el caso extremo, si sólo se utiliza un pixel, no será posible la discriminación entre texturas.



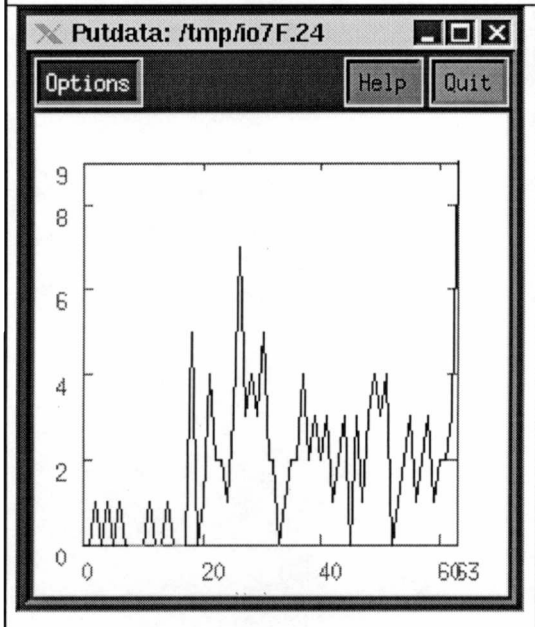
Arpillera Completa

Dispersión = 44,762
Varianza = 2861,957
Skewness = 73,972
Kurtosis = 2,167



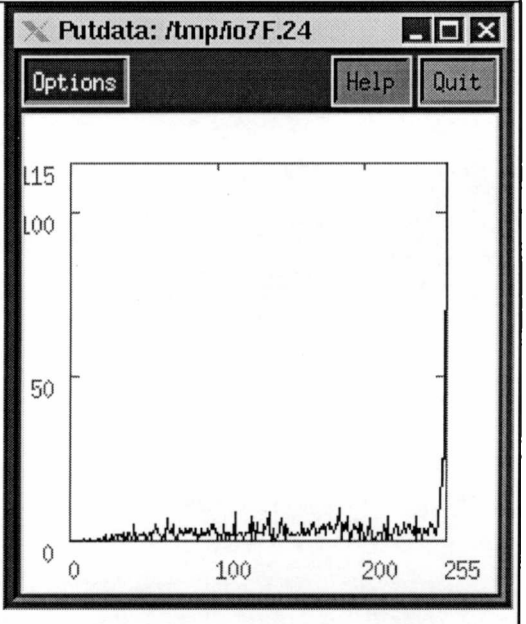
Pasto Completo

Dispersión = 63,228
Varianza = 5212,520
Skewness = 93,100
Kurtosis = 1,898



Ventana chica de pasto

Dispersión = 46,860
Varianza = 3176,870
Skewness = 80,040
Kurtosis = 2,260

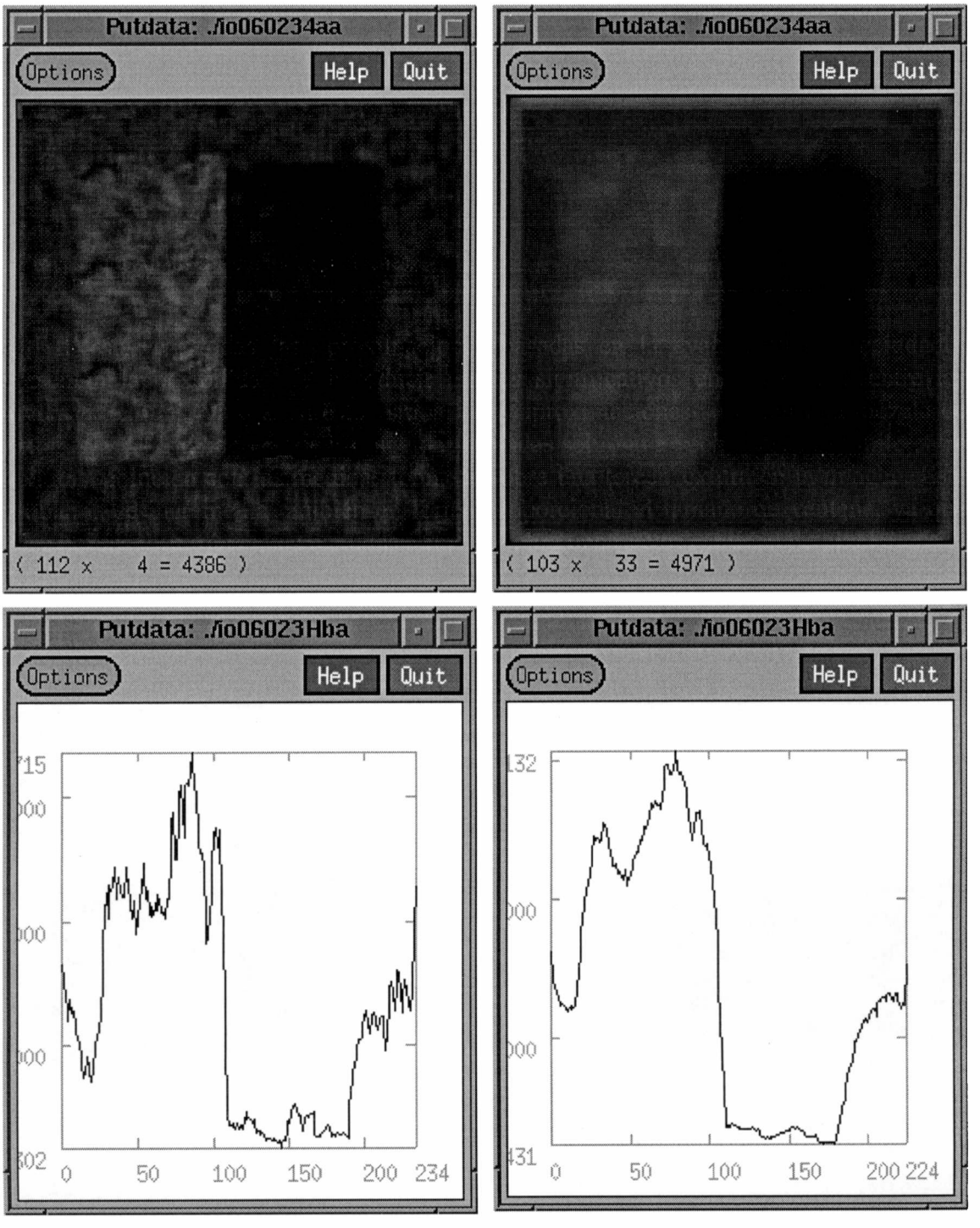


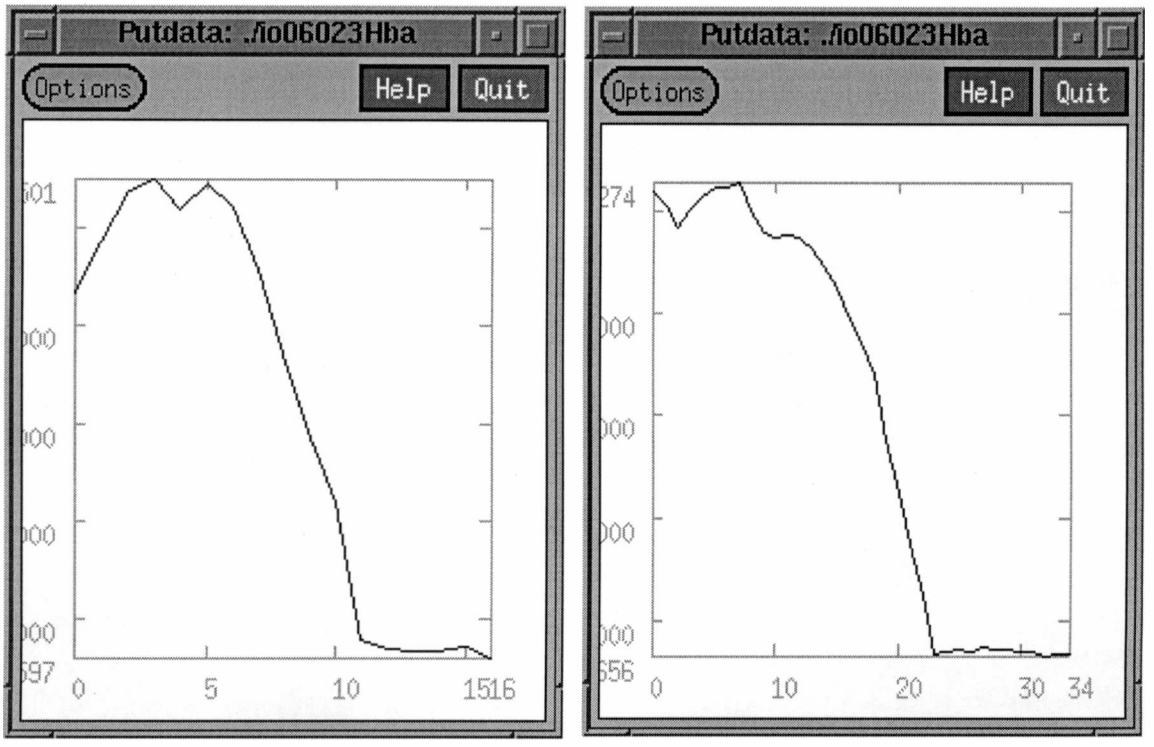
Ventana adecuada de pasto

Dispersión = 58,820
Varianza = 4610,060
Skewness = 88,510
Kurtosis = 1,880

Como puede observarse en la figura anterior, la elección de un tamaño de ventana pequeño puede aumentar la taza de error de la clasificación final.

A su vez, todas las aproximaciones basadas en la evaluación de vecindades presentan la posibilidad de introducir errores significativos en los límites entre las diversas texturas, ya que en estos sitios, tales aproximaciones soportan una mezcla de texturas, y podría ocurrir que las medidas tomadas sobre una vecindad en el límite entre texturas, coincida con las medidas tomadas sobre una tercer textura en la imagen. Este problema se incrementa también en forma proporcional al tamaño de ventana, ya que tales bordes influirán en las medidas tomadas sobre una mayor cantidad de pixels.





En la figura anterior se observan dos imágenes de Varianza calculadas utilizando ventana de 9x9 y 17x17 respectivamente. Luego pueden verse sendos cortes transversales representando los valores alcanzados por las medidas calculadas, donde se aprecia la coincidencia de los valores del borde entre las regiones de pasto y pelo, con los de arpillera. Finalmente se muestra una vista más detallada del borde, donde se aprecia un crecimiento significativo en función del tamaño de ventana seleccionado. En el primer caso, el borde entre las regiones abarca cerca de 7 pixels, mientras que en el segundo caso, los pixels afectados son aproximadamente 15.

El costo en cuanto a tiempo crece proporcional a la cantidad de puntos incluídos por el tamaño de ventana utilizado, por lo que resulta importante mantener tal tamaño tan pequeño como sea posible, siempre sin perder características de la textura.

Tal vez, los problemas más relevantes entre los antes mencionados sean aquellos que conducen a errores en la clasificación, ya que los problemas en cuanto a tiempo de procesamiento pueden ser mejorados introduciendo técnicas de procesamiento concurrente.

Consideraciones acerca del tamaño de ventana

Este problema está asociado con el *grano* que presenten las diversas texturas intervinientes en la imagen analizada. Intuitivamente puede decirse que la región correspondiente al pasto en la figura 1 posee un *grano* más grueso que las otras.

El grano de una textura estará asociado a la sensación visual experimentada al observarla. Las texturas que presenten mayor rugosidad visual, tendrán en general grano grueso, mientras que las texturas que presenten mayor suavidad visual, poseerán generalmente grano fino.

En el contexto de imágenes naturales, podemos pensar al texel como aquella región mínima que incluya todos los distintos *elementos* de relevancia intervinientes en la textura bajo observación. Por esto, el tamaño del texel crece con la cantidad y tamaño de los distintos elementos que lo constituyen.

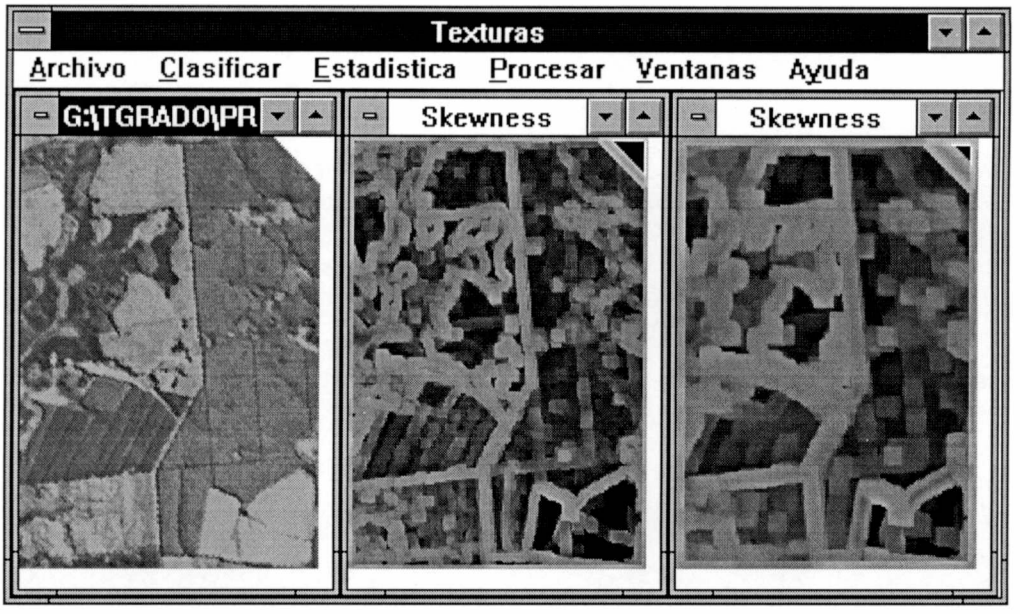
Cada *elemento* está compuesto por los diversos grupos de pixels conexos dentro del texel que presentan una diferencia pequeña entre sus valores de gris. Estos son llamados por Haralick *primitivas de niveles de gris*. [6]

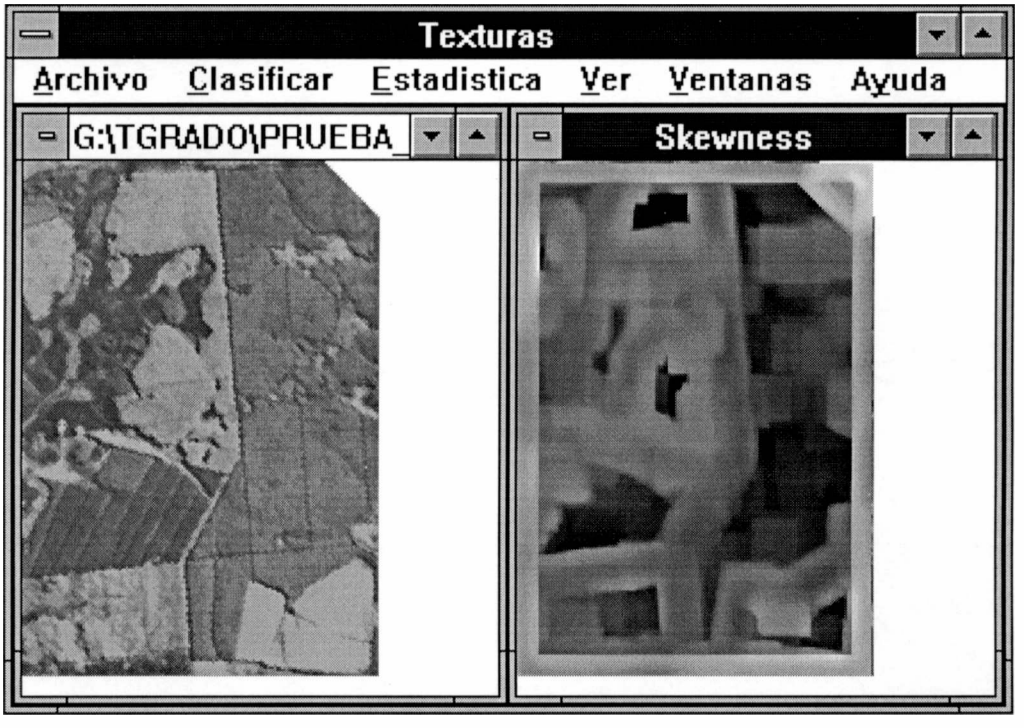
El grano y la rugosidad serán entonces directamente proporcionales al tamaño de tales elementos.

Establecemos a continuación, algunas pautas para la decisión del tamaño de ventana, teniendo en cuenta nuestras experiencias.

De acuerdo a lo visto anteriormente resulta importante para la performance final de la clasificación, el tamaño de ventana T elegido, por lo cual tratamos de establecer algunas pautas para determinar dicho tamaño en forma adecuada.

- Encontramos una cota inferior de T considerando el tamaño de texel de la textura a identificar. T debe ser al menos tan grande como el tamaño de dicho texel, de manera de no perder información relevante.
- Idealmente, T deberá ser igual al tamaño del texel de mayor dimensión existente en la imagen analizada, lo cual constituiría otra cota inferior. Esta sugerencia difícilmente pueda ser respetada ya que no siempre resulta conveniente utilizar un T demasiado grande.
- Asimismo, deben considerarse las dimensiones de las distintas regiones de textura intervinientes en la imagen, dado que la existencia de regiones pequeñas hará inconveniente la utilización de un T grande, ya que las medidas utilizadas resultarían fuertemente afectadas por las regiones de textura adyacentes. Esto constituye entonces una nueva cota superior para T.
- Otra cota superior estará dada por el crecimiento de los bordes en forma proporcional a T.
- Finalmente también debe tenerse en cuenta el costo en tiempo del procesamiento, el cual crece en forma directamente proporcional a T.





En las figuras anteriores se muestran los resultados de tomar las medidas de Skewness a una proción de una imagen con diferentes tamaños de ventana. En el primer resultado se utilizó una ventana de 7x7, y vemos que aparecen diferencias en regiones uniformes, pero no se deterioran zonas adyacentes a los bordes, en la segunda se utilizó una ventana de 9x9 y se extendieron los bordes ocupando regiones vecinas. En el tercer caso procesamos con el tamaño de ventana aproximado al tamaño del mayor de los texels de toda la imagen, utilizando 19x19 y vemos que se acentuó el efecto del ensanchamiento de los bordes, perdiendo subregiones pequeñas, pero en la región donde había una amplia zona de la misma textura se ve mejor identificada, sin los errores introducidos en los casos anteriores.

4.2 - Suavizado HS

Tomemos ahora el problema presentado anteriormente referido a los potenciales errores en los límites entre texturas.

Figura 2

La figura muestra un conjunto de regiones de superficie pequeña. Al intentar clasificar, se observan los problemas antes mencionados y analizados en [15]. En el primer resultado, puede apreciarse la dificultad en cuanto a la inclusión de los bordes de la región, lo que hace perder una superficie considerable de ella. En el segundo resultado, se observa la inclusión errónea de bordes de otras regiones, como presentando la misma textura que la seleccionada.

Hsiao y Sawchuck proponen un método de suavizado no lineal que permite reducir el problema de los bordes presente durante la utilización de las métricas de Laws, en las imágenes de energía textural, comentado en la sección 2.3.1.4 de este trabajo.

Este método se basa en la siguiente apreciación. Dado que al procesar con vecindades la posible degradación en los bordes lleva a cambios abruptos en las métricas obtenidas, y en su entorno se mantienen valores uniformes, se intenta hacer avanzar las regiones uniformes del entorno sobre el borde. para ello se toma el valor medio de la región de menor variación del entorno. Como los valores distorsionados no afectan más allá de un ancho de ventana, se toma la variación y el valor medio de las vecindades con un tamaño de dos veces el ancho de la ventana utilizada en la obtención de las métricas, de acuerdo lo realizado por Hsiao y Sawchuck.

Este método tiene un doble efecto: por un lado, logra reducir el ancho de los bordes que introducen un potencial error en la clasificación, y a su vez realizan un suavizado sobre las medidas observadas, lo cual resulta sumamente beneficioso, ya que los valores pertenecientes a regiones con texturas similares, presentarán valores más similares aún. Esto puede observarse en el ejemplo siguiente.

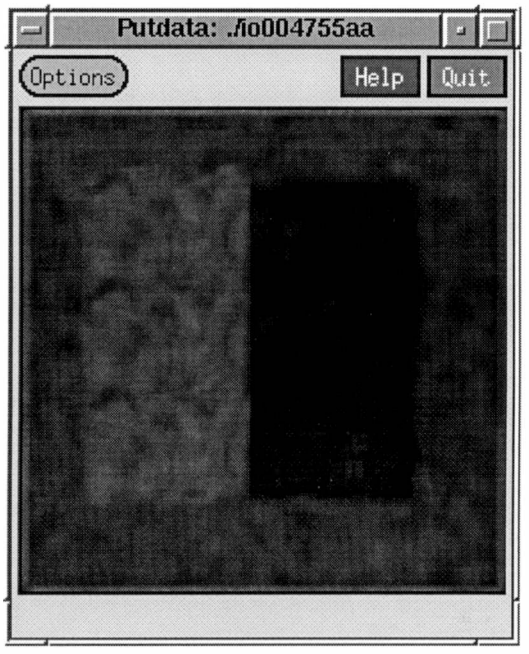
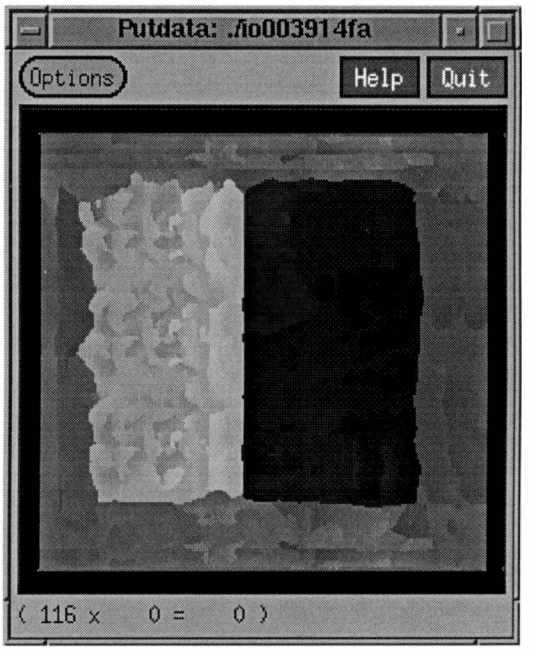
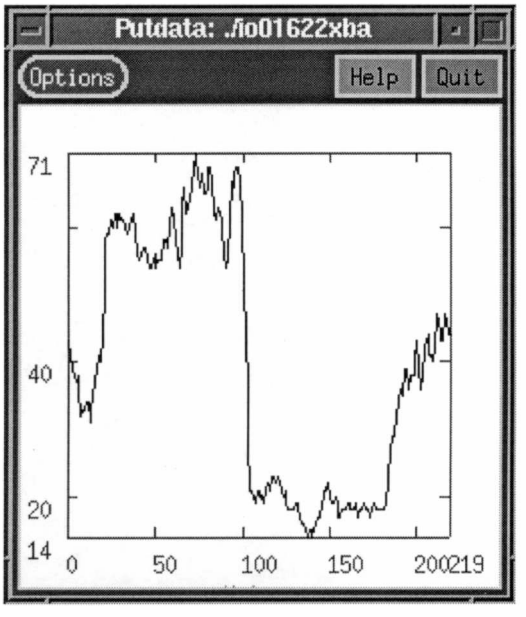


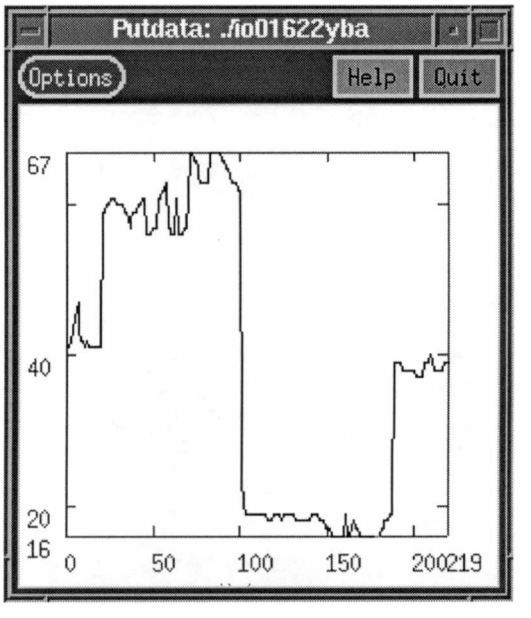
Imagen de Característica



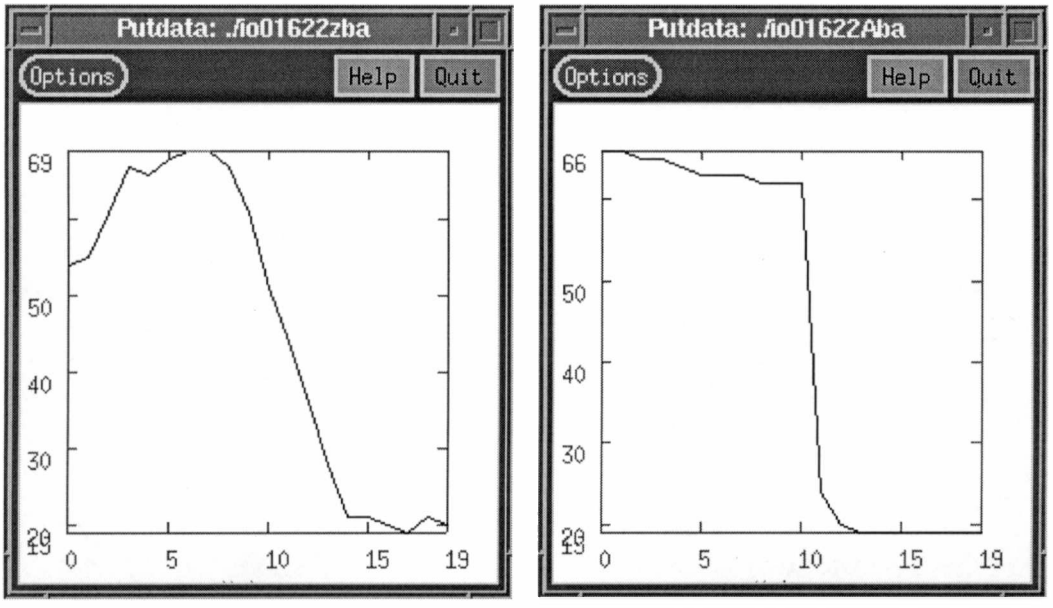
Mejorada



Corte Transversal



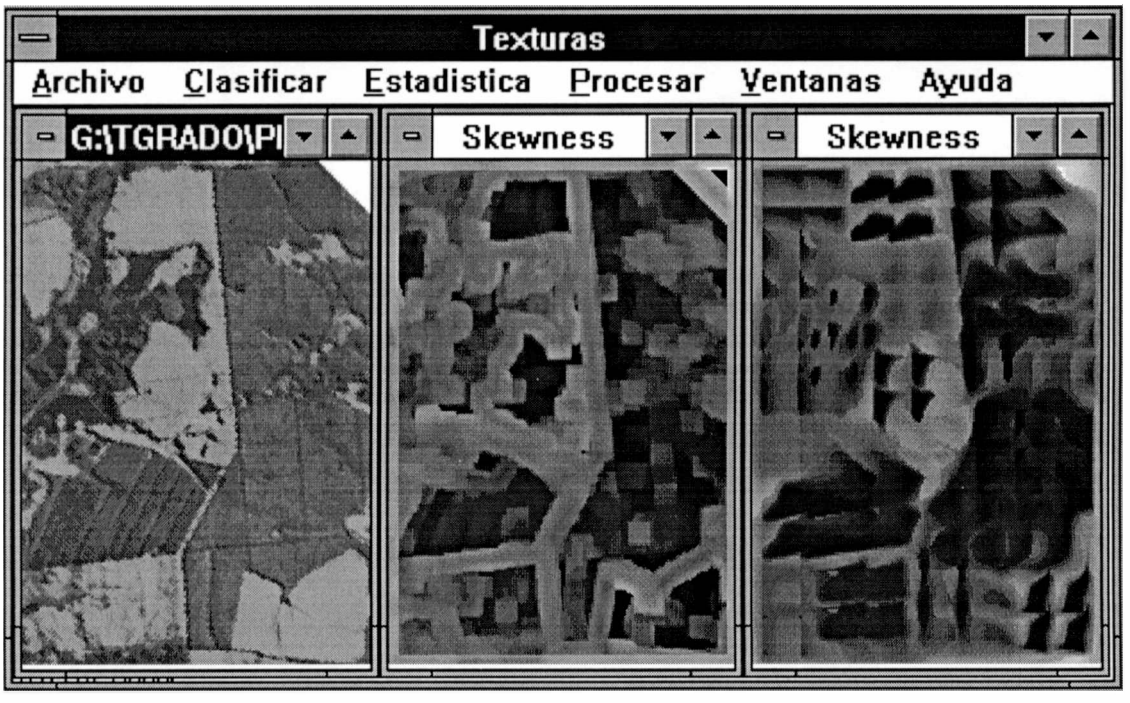
Corte Transversal mejorado



Zoom del Borde

Zoom del Borde Mejorado

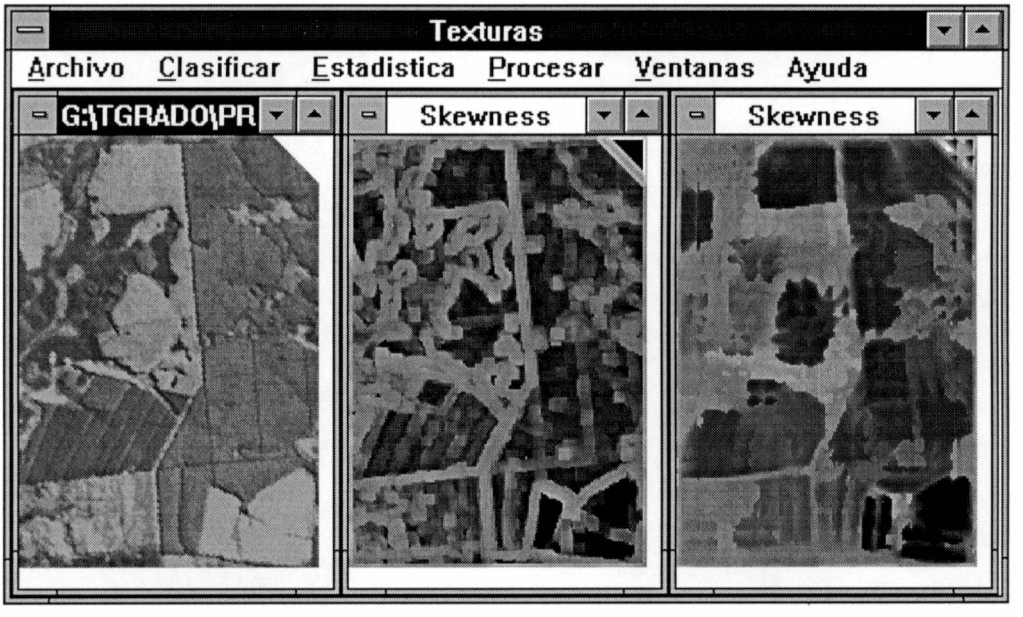
El hecho de trabajar con un tamaño de ventana correspondiente al doble del utilizado por las métricas, puede resultar perjudicial para aquellas regiones que presenten una superficie demasiado pequeña, siendo posible que estas desaparezcan al realizar el suavizado antes mencionado.



En la Figura anterior pueden verse (de izquierda a derecha) la imagen original y las imágenes correspondientes a los valores de Skewness y Skewness luego de aplicar el suavizado HS. Como puede apreciarse en la imagen central, la utilización de un tamaño de ventana excesivo produce la aparición de bordes importantes en la imagen resultante. En este caso particular, algunas regiones que presentan valores homogéneos

en la imagen perteneciente a la métrica utilizada, son degradados al aplicar el suavizado HS resultando en el efecto contrario al buscado. Este problema sólo ocurre en aquellas regiones que presentan una superficie relativamente pequeña respecto de las dimensiones de la ventana utilizada. A su vez, puede apreciarse que algunas regiones que presenten un alto porcentaje de borde, son interpretadas por el suavizador como una región homogénea, resultando en el crecimiento de tales bordes. Esto se debe a la existencia de texturas de grano considerablemente más grueso que el promedio de los granos de las texturas intervinientes.

En la figura siguiente puede observarse que tal efecto negativo no se produce al tomar un tamaño de ventana adecuado.



4.3 - Clasificación

La tarea más importante luego de la extracción de características es la clasificación de los pixeles de la imagen según alguna función discriminante.

De acuerdo a lo visto anteriormente, los valores obtenidos a partir de pixels pertenecientes a una región de una textura dada presentarán valores aproximadamente iguales, pero en general aparecerán variaciones alrededor de algún valor determinado.

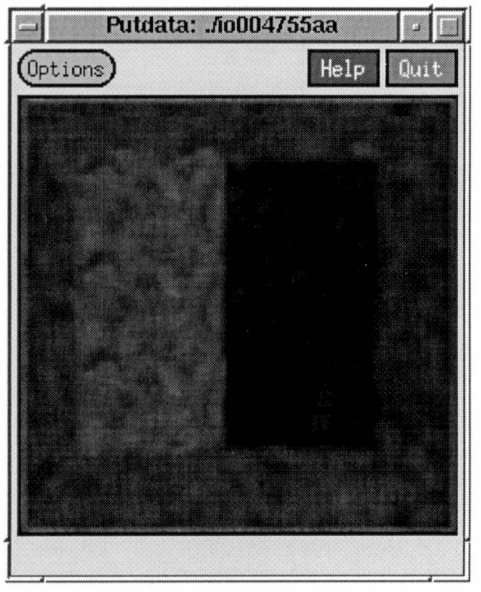
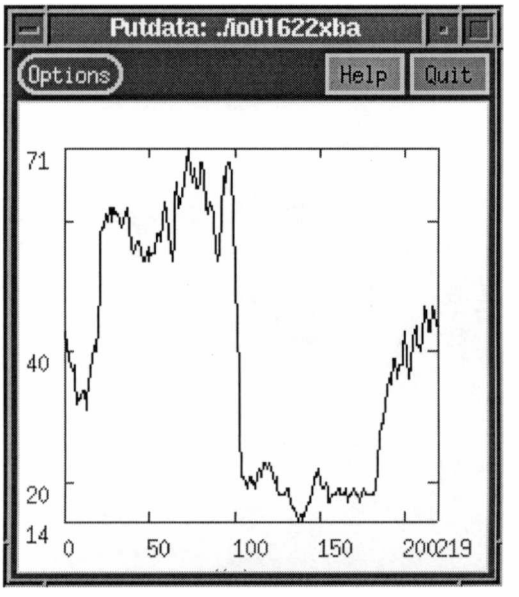
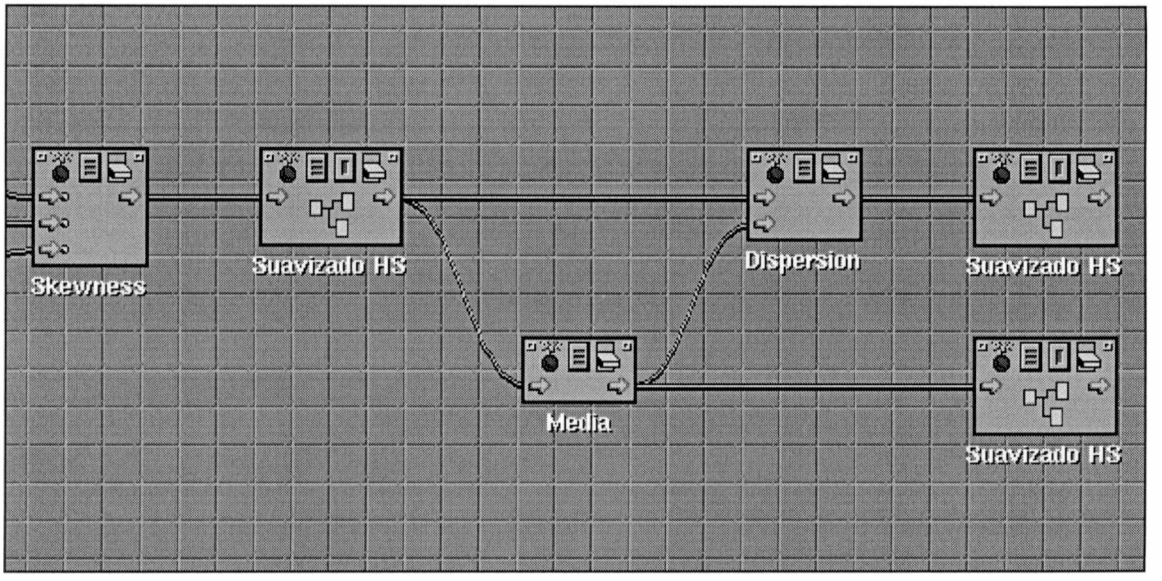


Imagen de Característica



Corte Transversal

Puede observarse en la figura que las medidas correspondientes a pixels pertenecientes a una región de textura homogénea, presentan valores aproximadamente iguales, pero aparecen también las variaciones antes mencionadas. Por esto resulta conveniente la utilización de medidas tomadas sobre las métricas originales, que permitan capturar estas variaciones teniendo en cuenta las vecindades alrededor del pixel a ser clasificado. Las medidas que permiten captar esta información son Media y Dispersión, que deben ser calculadas sobre los valores entregados por las métricas aplicadas en forma directa a la imagen analizada.



Las medidas que intervendrán finalmente en la clasificación son obtenidas también en base a vecindades y en consecuencia resulta conveniente aplicarle el suavizado HS con el fin de mejorar la respuesta en los bordes y homogeneizar las regiones correspondientes a una misma textura. De esta manera, suman ocho las medidas que finalmente conformarán el vector de características y serán la entrada del módulo de clasificación.

Es conveniente recordar que la idea propuesta originalmente por los especialistas de la DAIS era permitir al usuario experto señalar una determinada región que presente una textura identificada visualmente, para luego localizar todas las regiones sobre la imagen que muestren la misma textura.

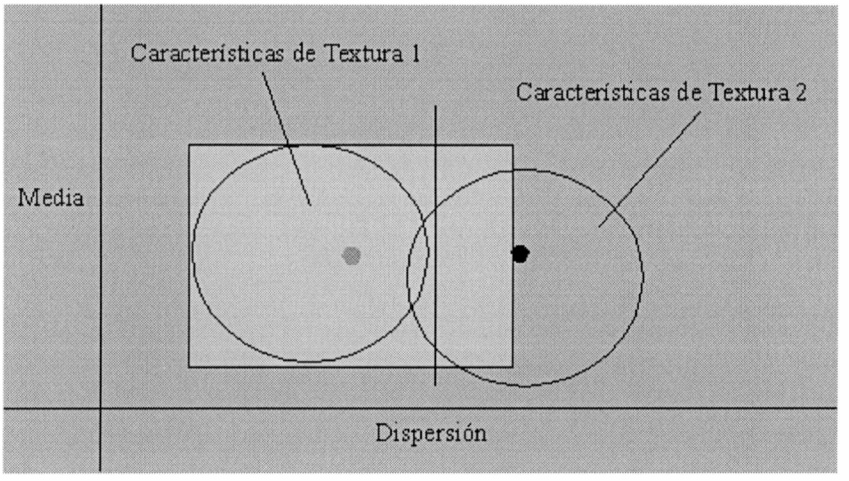
De esta manera, se impone una fuerte restricción sobre la cantidad y calidad de información que es posible tener en cuenta en el momento de analizar las imágenes. Una alternativa sería determinar un conjunto de texturas a tener en cuenta, y realizar a partir de esto un estudio profundo de las medidas de los prototipos disponibles sobre cada una de ellas.

De acuerdo a la metodología de trabajo propuesta, sólo se dispondrá de algunos pocos prototipos de la textura a identificar y eventualmente algunos prototipos de las texturas que puedan resultar similares a la señalada, pero que pertenezcan a otra clase de textura. Estos últimos prototipos pueden ser llamados Ejemplos Negativos.

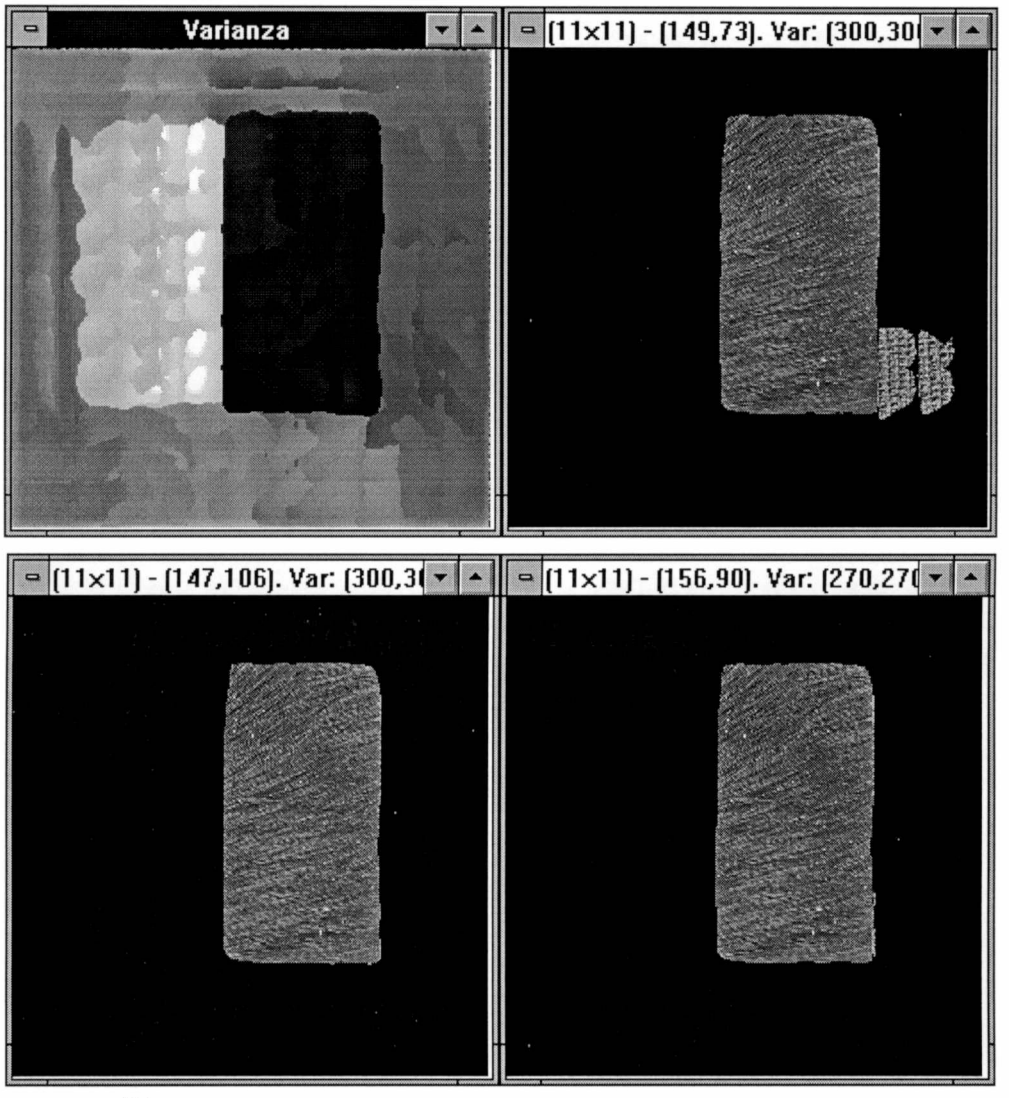
Así, la cantidad de clases tenidas en cuenta por el clasificador se reduce a dos: la primera correspondiente a la textura a identificar y la segunda correspondiente a la clase de los Rechazados. El criterio de clasificación utilizado corresponde entonces a un clasificador de distancia mínima.

Debido a que la clase de los ejemplos negativos interviene en la clasificación en forma opcional (los ejemplos negativos pueden **no** ser incluidos), es necesario definir un criterio adicional para determinar la pertenencia a la clase de la textura seleccionada. Con este fin se define un conjunto de umbrales o márgenes de tolerancia, uno para cada uno de las componentes del vector de características.

Esta tolerancia no necesariamente es la misma para todas las medidas, por lo que se puede dar diferentes rangos de tolerancia según la métrica, y diferentes aún para las dos medidas (Media y Dispersión) tomadas sobre cada una de las métricas.



En esta imagen vemos la influencia de los márgenes en la clasificación, donde el punto gris representa los valores de las características seleccionadas en la textura 1 y el rectángulo que lo rodea, el rango que abarcaría con un margen para cada una de las mismas. Como se puede apreciar, tomando esos márgenes se incluirían todas las posibles medidas correspondientes a la textura 1, pero se incluirían también algunos puntos de las textura 2. Utilizando un ejemplo negativo (punto negro) en la textura 2, los puntos de características más cercanos al ejemplo negativo no serían incluidos como parte de la textura 1, por lo que la clasificación mejoraría sustancialmente con respecto al caso en el que no se tomara en cuenta dicho ejemplo negativo. A continuación se muestran algunos ejemplos.



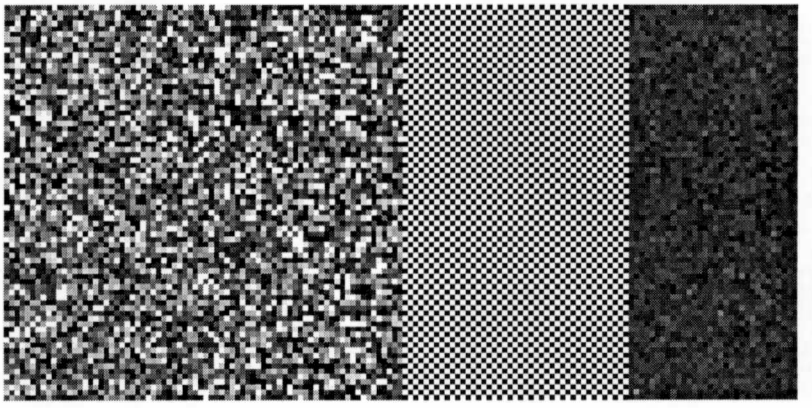
Primera ventana de clasificación Márgenes: (300, 300) Bien en la región selec: **94,13** % Mal fuera de la región: **3,11** % Error: **3,70** %	
Clasificación con Ptos Negativos Márgenes: (300, 300) Bien en la región selec: **94,13** % Mal fuera de la región: **0,15** % Error: **1,32** %	Clasificación con Márgenes Menores Márgenes: (270, 270) Bien en la región selec: **93,46** % Mal fuera de la región: **0,22** % Error: **1,58** %

Como puede apreciarse en la figura, al clasificar tomando un margen grande, se produce una determinada taza de errores. En las imágenes siguientes se ejemplifican las posibles formas de evitar tal problema. En el primer caso se incluyó un ejemplo negativo, mientras que en el segundo se redujeron los márgenes de tolerancia, logrando en ambos casos una mejora en la clasificación final.

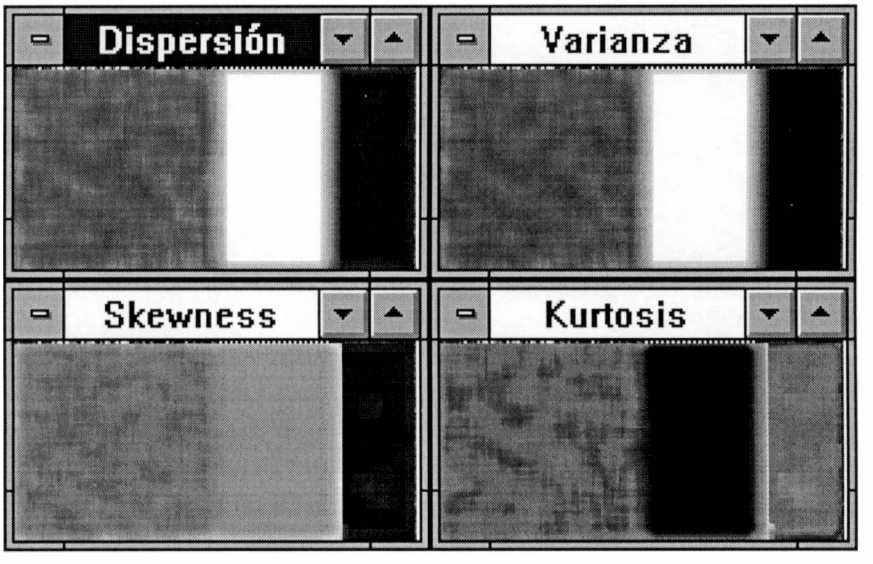
No siempre son necesarias todas las métricas para reconocer una textura, por ello no es utilizado en todos los casos el conjunto completo de las características para

clasificar, sino las que sean más adecuadas de acuerdo al tipo de textura a reconocer y al contexto donde ella se encuentre. Por ello es posible seleccionar en cada caso qué métricas intervendrán finalmente en la tarea de clasificación.

De acuerdo al análisis llevado a cabo al principio de este capítulo y a las experiencias realizadas intentamos determinar algunas pautas para la elección adecuada de las métricas a elegir, dada una textura a identificar. Analizaremos ahora algunas imágenes especialmente diseñadas a tal fin.

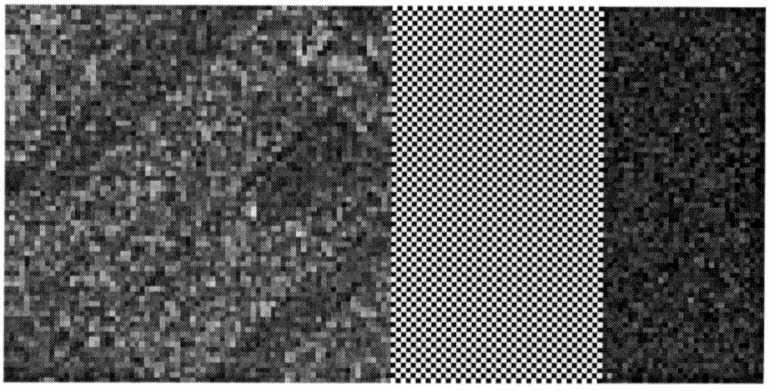


En la primera de ellas observamos (de izq. a der.) una región de actividad local muy importante en el contexto de imágenes reales, otra de actividad local mucho mayor que la anterior (la máxima posible, dificilmente alcanzada por una imagen real) y una tercera de actividad local muy baja, lo cual puede apreciarse en los valores dados por la Dispersión.

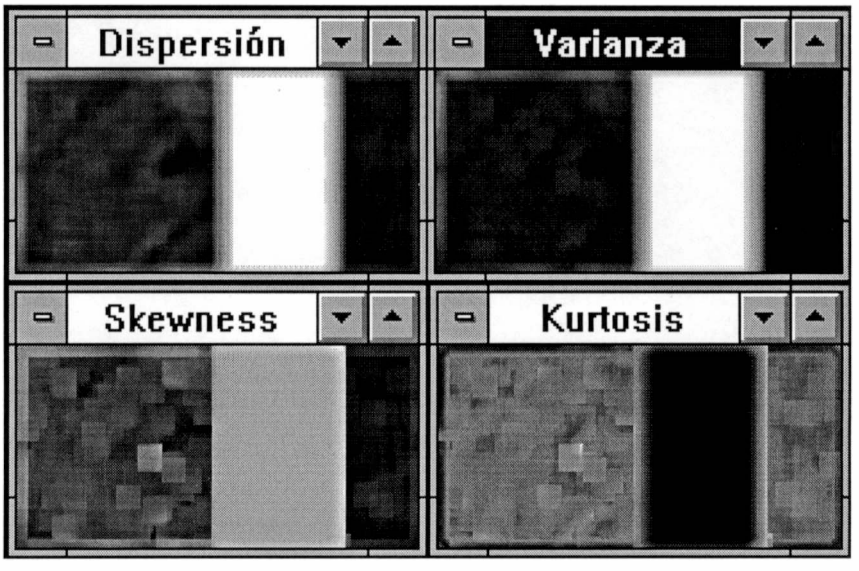


Los valores de Dispersión, Varianza y Skewness nos darán una medida directamente proporcional a la actividad local de los valores de gris en la región analizada. El incremento en las medidas de Varianza será más notorio que aquellas de

Dispersión debido a su crecimiento exponencial, lo cual se acentúa cuando se trate de regiones de actividad local alta.



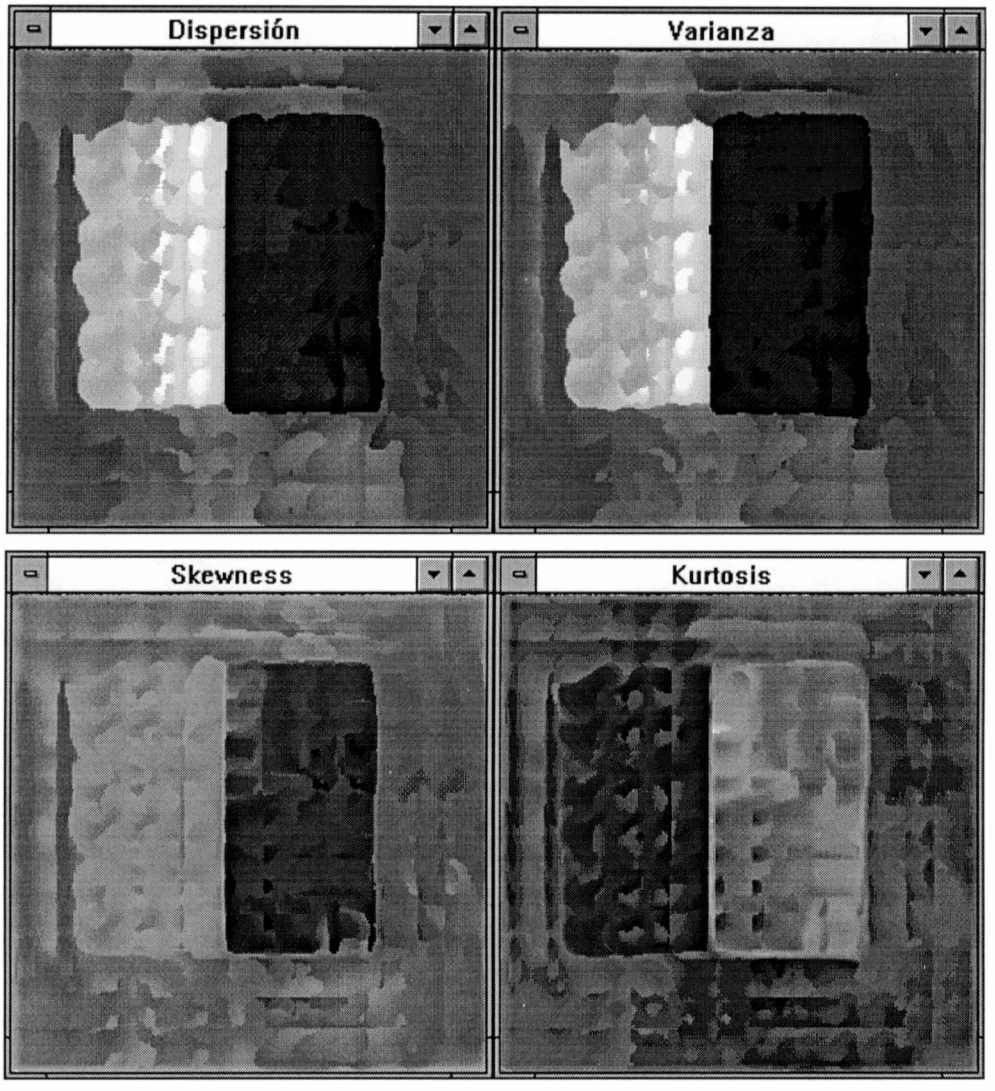
En esta imagen observamos (de izq. a der.) una región de actividad local relativamente baja en el contexto de imágenes reales, y las otras dos porciones descriptas en el ejemplo anterior.



Aquí podemos apreciar que las medidas de Skewness permiten una mayor discriminación entre regiones de actividad local baja. Esta métrica no es adecuada para distinguir regiones que presenten una actividad local alta, ya que los valores obtenidos en estos casos serán similares.

Utilizando Kurtosis se obtiene una buena distinción entre regiones de actividad local alta y no es aplicable en la discriminación de regiones de actividad local baja.

La figura siguiente muestra las imágenes correspondientes a cada una de las métricas evaluadas, luego de aplicarles el suavizado HS.

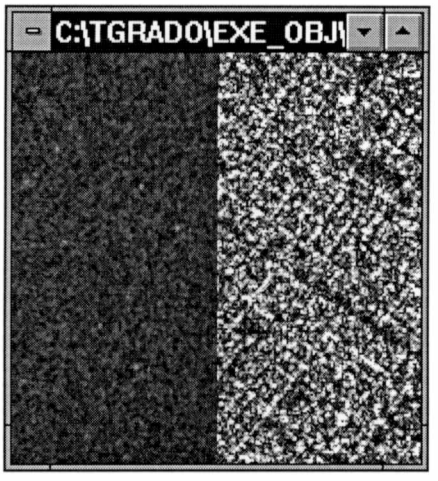


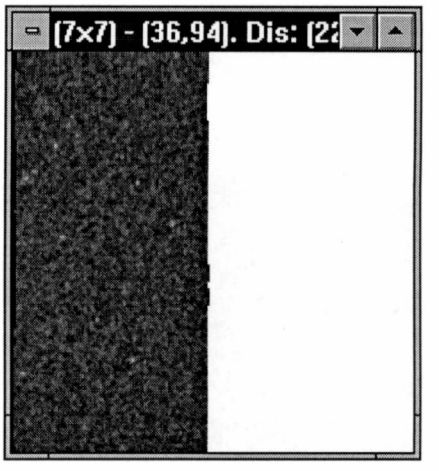
La medida de similitud utilizada como función discriminante para determinar la semejanza entre los vectores de características y los prototipos involucrados, es la Distancia Euclídea.

En el trabajo en Khoros utilizamos un clasificador de distancia mínima existente, el cual separa una imagen en diferentes regiones en base a un conjunto de prototipos representantes de las distintas texturas a ser identificadas. De esta manera se obtiene una imagen representando las diferentes regiones de textura existentes.

El criterio final de clasificación utilizado determina que un vector de características pertenece a la clase de la textura buscada si el prototipo más cercano a él pertenece a esta clase y se ubica dentro de los márgenes de tolerancia, teniendo en cuenta sólo aquellas métricas seleccionadas. En caso contrario, el vector es rechazado.

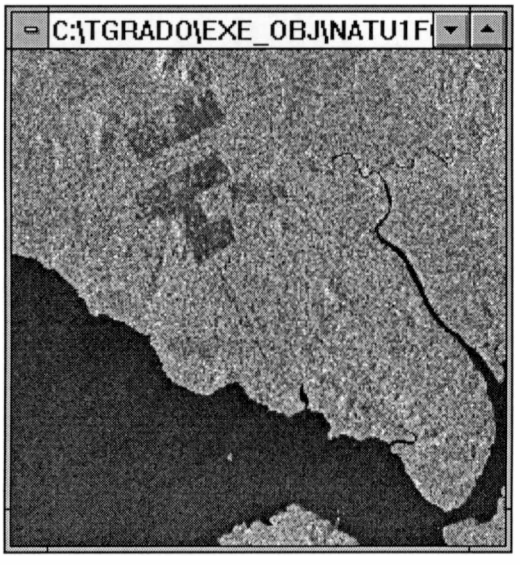
Observemos finalmente algunos resultados finales de clasificación.

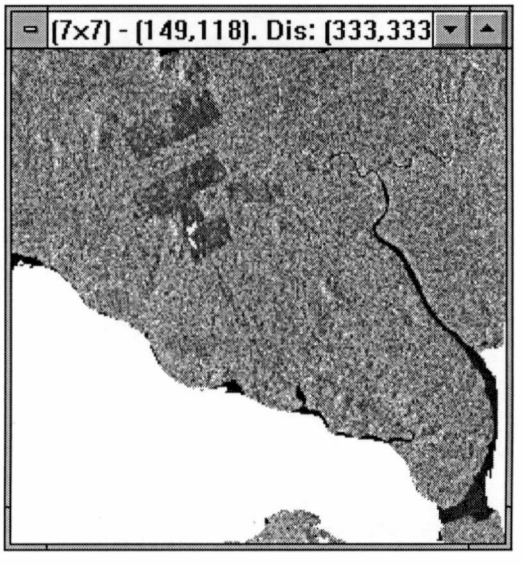




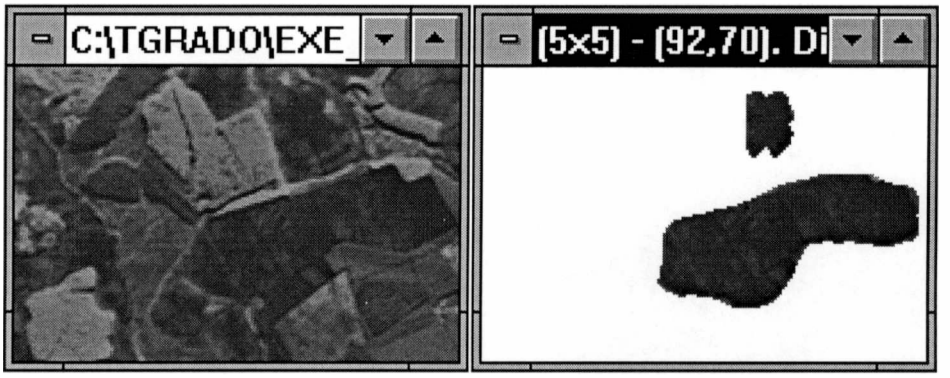
Bien en la Textura seleccionada:	**97,04 %**
Bien fuera de la Textura:	**99,59 %**
Total de Aciertos:	**98,01 %**
Error Total:	**1,99 %**

En la figura previa se observa una imagen de radar compuesta de dos regiones de texturas distintas. A su derecha, el resultado luego de una clasificación, donde los pixels rechazados son exhibidos con tonalidad blanca. Debajo de ellas se muestran los porcentajes de acierto/error obtenidos.





En la figura anterior se observa una imagen obtenida por medio de radar, donde pueden distinguirse dos regiones, una de ellas correspondiente a una superficie terrestre, y la otra a una superficie marina. En la imagen de la derecha puede apreciarse el resultado de clasificación, luego de haber seleccionado la superficie terrestre para su reconocimiento. La región correspondiente al río es clasificada como superficie terrestre debido a un efecto no deseado del suavizado HS. A pesar de ello, se observa una performance razonable en los resultados finales de la clasificación.



En la figura anterior se aprecia una imagen tomada desde un satélite, donde aparecen diversos tipos de terreno, así como superficies correspondientes a distintas vegetaciones. Aquí puede observarse un buen resultado de clasificación donde no son incluidas aquellas regiones que presentan la misma textura que la seleccionada, pero sobre una superficie demasiado pequeña. Asimismo se aprecia una mejora significativa respecto de la clasificación mostrada en la figura 2, debido a la utilización del suavizado HS.

CAPÍTULO 5

5 - Apreciación Final y Líneas de Trabajo Futuras

5.1 - Conclusiones

Este trabajo constituye el comienzo de un nuevo tema de estudio e investigación en el contexto del Departamento de Informática, en lo que a nuestro conocimiento respecta, lo cual implica la posibilidad futura de profundizar en las diversas líneas de análisis textural de imágenes mencionadas en este informe, y el estudio de sus posibles aplicaciones.

Se estudiaron y utilizaron herramientas de desarrollo tales como C++ y Khoros con el fin de realizar la evaluación de las métricas alrededor de las cuales se centró nuestro trabajo y a su vez, desarrollar una aplicación que asista en la interpretación de imágenes. En este sentido, fue necesario agregar algunos operadores al ambiente de desarrollo Khoros, ya que en esta herramienta específica para procesamiento de imágenes no se encuentran los operadores que permitan extraer las características deseadas en base a vecindades de los pixels [14]. Asimismo, fue implementada una aplicación completa como herramienta para fotointerpretación asistida, utilizando el lenguaje de desarrollo C++.

Otro resultado parcial de este trabajo determinó que es posible adaptar la técnica de suavizado propuesta por Hsiao y Sawchuck (utilizada entonces para mejorar la performance en la discriminación de texturas utilizando métricas de Laws), con el fin de incrementar la capacidad de discriminación mediante las medidas de histogramas de primer orden aquí estudiadas.

Podemos decir finalmente que las métricas de Dispersión, Varianza, Skewness y Kurtosis parecen adecuadas para su utilización en la etapa de extracción de características durante el análisis textural de imágenes, de acuerdo a las experiencias realizadas y expuestas en este trabajo. En particular, estas medidas pueden ser utilizadas para la fotointerpretación asistida sobre imágenes de origen natural, como lo son las imágenes obtenidas mediante radares o sensores multiespectrales, o bien a partir de fotografías aéreas, teniendo en cuenta las pautas establecidas en este trabajo en cuanto a la elección del tamaño de ventana y la selección de las métricas adecuados.

5.2 - Aplicaciones y líneas de trabajo futuras

El análisis textural de imágenes en general, y las métricas de histogramas de primer orden en particular, pueden ser de utilidad en problemas tales como:

- Análisis de imágenes médicas: detección de tumores, medición de tamaño y forma de órganos internos.
- Automatización industrial: identificación de partes sobre líneas de ensamble, inspección de defectos y fallas.

- Robótica: reconocimiento e interpretación de objetos en una escena, control de movimiento y ejecución mediante retroalimentación visual.
- Procesamiento de imágenes de sensores remotos: análisis de imágenes multiespectrales, predicción de fenómenos meteorológicos, clasificación y monitoreo de zonas urbanas, agrícolas y marinas.

Asimismo, algunos de estos problemas pueden implicar respuestas en Tiempo Real, lo cual haría necesario investigar las formas de paralelizar los algoritmos involucrados.

Anexo A: Bibliografía

1 JAIN, A., *"Fundamentals of Digital Image Processing"*, PRENTICE HALL, 1989.

2 SCHALKOFF, R., *"Pattern Recognition: Statistical, Structural and Neural Approaches"*, JOHN WILEY & SONS, INC., 1992.

3 SCHALKOFF, R., *"Digital Image Processing and Computer Vision"*, JOHN WILEY & SONS, INC., 1989.

4 PRATT, W., *"Digital Image Processing"*, JHON WILEY & SONS, INC., 1978.

5 NIEMANN, H., *"Pattern Analisys and Understanding"*, SPRINGER-VERLAG, 1990.

6 HARALICK-SHAPIRO, *"Computer and Robot Vision"*, ADDISON-WESLEY, 1992.

7 STROUSTRUP, B., "The C++ Programming Language" ADDISON-WESLEY, Second Edition, 1991.

8 WEISKAMP K. - Flamig B., "The Complete C++ Primer", ACADEMIC PRESS INC, 1990.

9 PAPPAS C., MURRAY W."Manual de Borland C++ 4.0", MC GRAW - HILLS, 1994.

10 BARKAKATI N.,"Borland C++ - Developer's Guide", PRENTICE HALL - SAMS PUBLICING, 1994.

11 K.R.I. - *Khoros Pro Handbooks*- KHOROS PRO, 1996.

12 JORDAN-LOTUFO, *"Digital Image Processing with Khoros 2.0"*, World Wide Web Courseware, 1994.

13 ERDAS, User manual, Field Guide. ERDAS.

14 CHAMPREDONDE, MULINARIS, RAIMONDI, PLATZECK, DE GIUSTI, "A Comparison of Development Facilities Between Khoros and C++", Khoros Symposium Proceedings 1997.

15 CHAMPREDONDE, MULINARIS, RAIMONDI, PLATZECK, DE GIUSTI, "Algoritmos de reconocimiento de patrones en base a texturas", Anales de I.C.I.E. 1997