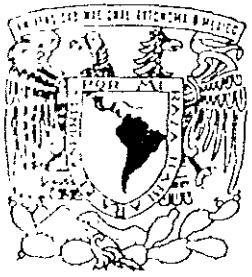


UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

CAMPUS "ACATLAN"

SISTEMA DE VISUALIZACION DE DATOS TOPOGRAFICOS DEL TERRITORIO NACIONAL

T E S I S

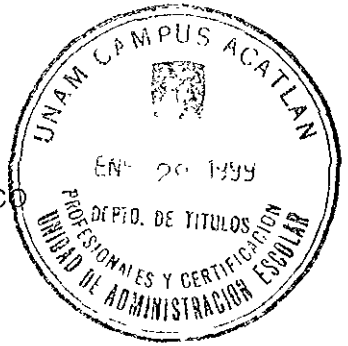
QUE PARA OBTENER EL TITULO DE:

LICENCIADO EN MATEMATICAS APLICADAS Y COMPUTACION

P R E S E N T A :

ELIO VEGA MUNGUIA

ACATLAN, ESTADO DE MEXICO 1999



Dedico este trabajo a:

Mi **papá Juvenal** por su ejemplo, y apoyo incondicional que siempre me ha brindado.

Mi **mamá Sofía** por su dedicación y amor incondicional

A mis **hermanos José, Lila y Nadia** por su cariño y alegrias que hemos vivido

A mis **tios y tias Hugo, José, Lila, Virginia, Leonor, Martha, Elsa, Cristina** por su apoyo incondicional.

A mis demás Familiares por su confianza.

A mi **amigo y asesor Dr. Luis Javier Alvarez** por su paciencia, apoyo, confianza, conocimientos e inestimables consejos

A mis **amigos y amigas Carmen Ramos, Victor Godoy, José Luis Villarreal, Lizbeth Heras, Karina Jimenez, Ivonne Jimenez, Oscar Garcia, Ignacio Vega, Luis Enrique Villavicencio, Gabriela Barbosa, Verónica Alegria** por su amistad, compañia y apoyo.

A la memoria de mis abuelitos José y Elena.

Agradezco a la U.N.A.M. por que es la fuente de mis conocimientos. A mis maestros y en especial a todas las personas del Laboratorio de Visualización, del Departamento de Supercómputo y del Laboratorio de Simulación de Materiales de la DGSCA por el apoyo para realizar este trabajo.

Índice General

Índice de Figuras

Índice de Tablas

Introducción

El Instituto Nacional Estadística, Geografía e Informática (*INEGI*) publicó en 1994 la topografía de todo el territorio nacional en forma digital en un disco compacto. Esta información viene acompañada de una serie de programas para *DOS* que permiten recuperar la información, además despleguearlos. Sin embargo, con estos programas la información que se puede recuperar se limita a la región delimitada a un solo cuadrante geográfico (que es lo que abarca cada uno de los archivos), tambien el despliegue es muy pobre pero lo más importante, es que es un sistema limitado ya que no se pueden realizar análisis más específicos y útiles (calcular áreas, distancias, gradientes, etc), además de que la información se encuentra cifrada.

Con el objeto de contar con un sistema más versátil de análisis de los datos topográficos contenidos en la publicación del *INEGI* se desarrolló un sistema de visualización y manipulación de datos para estaciones de trabajo gráficas utilizando el paquete comercial *AVS* (por sus siglas en inglés *Advanced Visual System*).

AVS permite la programación visual y modular de procesos tanto gráficos como numéricos, con lo cual, se puede calcular una serie de parámetros geomorfológicos interesantes en ciencias de la tierra, tales como volúmenes, distancias sobre el terreno, alturas y pendientes, entre otras cosas. También se pueden manipular imágenes rotándolas, escalándolas y trasladándolas. Finalmente toda esta información se puede correlacionar con datos geográficos geológicos.

De esta forma, al contar con este conjunto de archivos y el sistema avan-

zado de visualización *AVS*, es factible el pensar en una serie de módulos (programas) para que en forma conjunta, se pueda decodificar la información, crear una superficie que represente a la topografía del terreno en forma tridimensional, calcular las curvas de nivel del terreno, realizar recortes de la zona o abarcar áreas tan vastas como se desee,calcular gradientes, esto involucra el crear algorítmos para disponer la información de tantos archivos cómo cuadrantes geográficos comprenda el área a visualizar Posteriormente se puede provechar esta información para calcular algunos parámetros geomorfológicos de volcanes que se encuentran en la zona, cómo por ejemplo el campo volcánico de Michoacán y Guanajuato, buscando que tanto el despliegue gráfico, el cálculo de las curvas de nivel, la superficie del terreno y los parámetros geomorfológicos, se realicen de una manera rápida al utilizar el procesamiento en forma distribuida así como el supercómputo.

En el primer capítulo se presenta una visión general de la visualización como una metodología computacional para hacer investigación; en el segundo se presentan los conceptos fundamentales de la programación modular, en la que está basado el funcionamiento del paquete utilizado para desarrollar el sistema: el capítulo tres constituye propiamente la memoria del desarrollo del sistema, los resultados de los cálculos de los parámetros geomorfológicos del campo volcánico mencionado se presentan en el tercer y último capítulo; para finalmente, recoger las conclusiones.

La pertinencia de la realización de este trabajo, reside en la oportunidad que su desarrollo puede brindar en el modo que se utiliza la formulación explícita del fundamento teórico de la visualización, explorar las posibilidades de explotación de la programación modular en la creación de sistemas que entrañan cierta complejidad y finalmente, resolver el problema particular de calcular de manera automática, sin necesidad de hacer exploración de campo, una serie de parámetros geomorfológicos de interés, como por ejemplo, el proyecto de investigación de procesos físicos y fisicoquímicos de erupción volcánica que se lleva a cabo en el Laboratorio de Simulación de Materiales de DGSCA, UNAM.

Capítulo 1

Visualización y supercómputo

Durante la última década, se han desarrollado nuevas técnicas para el diseño de programas y computadoras, cuyas aplicaciones en la *investigación* científica, en donde la necesidad de almacenar y procesar información es crucial para resolver problemas. Las técnicas más modernas para realizar cálculos a gran escala y para analizar los resultados son la visualización y el supercómputo. En este capítulo se da una explicación de los conceptos más importantes que se manejan en estas dos disciplinas de las matemáticas aplicadas.

1.1 Visualización

En cierto tipo de problemas que se resuelven numéricamente utilizando recursos de supercómputo, el análisis de la información que se obtiene requiere de técnicas especiales de representación gráfica, especialmente cuando la cantidad de información es muy grande. Se entiende por visualización al conjunto de estas técnicas. Recientemente se han desarrollado metodologías para integrar cálculos numéricos y análisis de datos en la visualización. Esto proporciona a los científicos un poder enorme para interactuar con sus datos. La capacidad de dirigir los cálculos mientras se visualizan los resultados permite tomar el control de los cálculos y estudiar los resultados mediante la

observación de características como: estructura, patrones, tendencias, anomalías y relaciones, logrando identificar regiones de interés y/o parámetros apropiados para enfocar un *análisis cuantitavo* disponiéndolos en un dominio particular. En este dominio se pueden representar uno o más componentes sean éstos escalares o vectoriales que representen atributos pertinentes al problema bajo estudio.

Los usos de la visualización como metodología de investigación se pueden resumir en los siguientes tres rubros:

- Investigación. De este tipo de aplicación se pueden obtener resultados inesperados, que de otra forma ya sea experimental o teórica son imposibles de detectar.
- Confirmación. En el sentido del intercambio de ideas o creencias acerca de un sistema, entre la teoría y la experimentación.
- Divulgación. Esta es la más simple de las aplicaciones pues sirve solamente para ilustrar ideas y conceptos acerca de los sistemas bajo estudio.

En este trabajo el dominio a visualizar es simplemente R^3. Se cuenta con una malla regular en coordenadas geográficas con un campo asociado que representa las alturas de cada nodo de la malla en metros sobre el nivel medio del mar. Los nodos de la malla están numerados en un cierto orden, como se verá después. En el caso de las aplicaciones de este trabajo se utilizan, como se verá mas adelante, los tres tipos de visualización que se mencionan en el párrafo anterior.

Una nueva técnica de programación y desarrollo está constituida por el cómputo visual. Con ella es posible integrar el proceso de datos y su análisis visual en tiempo real, con lo cual es posible incorporar los elementos del ciclo de la ciencia computacional (figura 1.1).

Como se ve en la figura, este ciclo está compuesto por dos partes, en el ciclo externo predominan las investigaciones en grupo. En este ciclo se

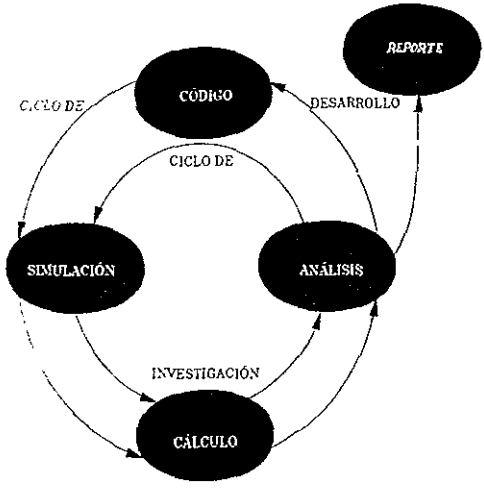


Figura 1.1: Ciclo de la Ciencia Computacional.

desarrollan y utilizan las aplicaciones. En el ciclo interno, esta el grupo de investigación, que es responsable de plasmar nuevas ideas y conceptos en los programas que se desarrollan.

El cómputo visual es una tecnología en crecimiento, se buscó un sistema en donde se pudieran incluir aplicaciones propias y a la vez aprovechar una gama muy amplia de metodologías de programación y procesamiento de datos. Esto conduce a aprovechar más tiempo en investigación y menos en el desarrollo de programas.

Como se mencionó en la introducción la herramienta que se utilizó para realizar la visualización, es un programa de visualización y programación comercial llamado AVS. A este tipo de sistemas se les considera ambientes de visualización modular. Para utilizarlo se necesitaron considerar los siguientes requerimientos

- Investigar cómo acoplar programas.
- Cómputo heterogéneo y distribuido.

Además de incluir:

- Despliegue gráfico.
- Técnicas avanzadas de visualización.

Al conjuntar todo esto, lo que se obtiene es una combinación típica de supercomputadoras, estaciones de trabajo, terminales-X y servidores, con la cual es posible hacer la visualización en forma conjunta. Su funcionamiento depende del orden en que son obtenidos los resultados de cada programa involucrado, ya que esto es la base para utilizar aplicaciones visuales en forma distribuida.

En el presente trabajo se hizo uso de la supercomputadora *CRAY-YMP* de la *UNAM* y de las estaciones de trabajo de los laboratorios de Visualización y de Simulación de Materiales de la DGSCA, procesando los cálculos en forma distribuida, esto es: se realizaron los cálculos de las transformaciones matemáticas en la *CRAY* y el despliegue gráfico en las estaciones de trabajo de los laboratorios mencionados.

Los programas que se desarrollaron fueron creados específicamente para la recuperación y visualización de la topografía del territorio nacional. Sin embargo, esto no impide que puedan ser utilizados en otras aplicaciones. Solamente hay que escoger desde qué máquina se ejecuta el programa principal para que éste sea el encargado de dirigir la ejecución de los diferentes módulos que lo conforman, sean locales o remotos.

En la figura 1.2 se muestra el programa visual que se desarrolló para este trabajo, los módulos etiquetados en inglés son implícitos de AVS.

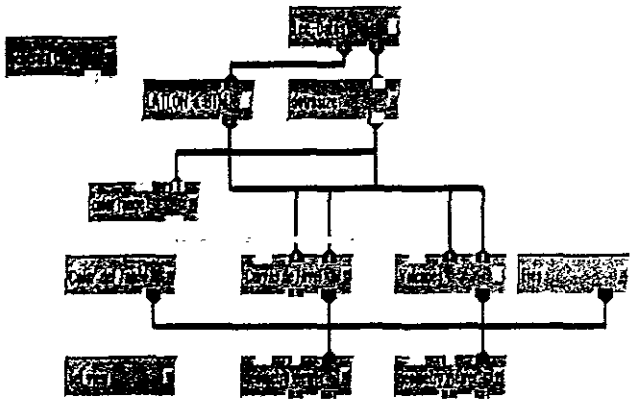


Figura 1.2: Programa Visual Principal

1.1.1 Módulos

La unidad fundamental del programa que se presenta en este trabajo es el módulo. Un módulo es una función programada en C o C++ que procesa datos de entrada y genera datos de salida. Estas funciones o módulos llevan a cabo cálculos relativamente pesados.

Para construir el sistema se conectaron gráficamente una serie de módulos formando así un programa visual. Las conexiones entre los módulos representan el flujo de datos que existe entre ellos: los datos originales están en diferentes archivos y solamente pueden ser leídos por un programa que esté corriendo en la red de módulos. Este puede ejecutarse en la misma máquina o en una remota. Posteriormente, los datos son transformados en una o más geometrías para que después otros módulos puedan desplegarlos en forma de una imagen.

1.1.2 Ejecución del flujo en la red

El propósito de construir esta red es disponer de un conducto con el cual se pueda observar el procesamiento de datos en cada paso. Los datos de salida de un módulo pueden ser la entrada de otro, dependiendo de las funciones que representen. Esta es la *forma en que* los datos pueden estar fluyendo a través de los módulos de la red, y finalmente pueden desplegarse o almacenarse en archivos. Este proceso requiere que cada módulo en la red sea invocado a su debido tiempo, supervisando qué cambios se han realizado y el medio por el que son transferidos entre los módulos. En la red, cada módulo es un proceso independiente.

Para los propósitos de este trabajo, la independencia de los módulos es muy conveniente, debido a que los módulos se pueden ejecutar en paralelo o de manera distribuida. En el caso del sistema que se presenta, el módulo *Curvas_de_Nivel* consume muchos recursos y es conveniente utilizarlo en la supercomputadora *CRAY-YMP* para acelerar los cálculos, y en forma simultánea, pero en la *ONYX*, se ejecuta el módulo *field_to_mesh*, que construye una superficie.

En resumen, las operaciones que incluimos para adaptar el flujo de datos y optimizar la arquitectura multi-procesos son:

- Utilizar varios módulos en una única ejecución.
- Ejecutar módulos en paralelo
- Ejecutar módulos en máquinas remotas.
- La dirección del flujo de los datos es "hacia abajo".
- Los datos en memoria se comparten por varios módulos.
- La comunicación entre los módulo es directa.

1.1.3 Modelo de flujo de datos

Como se mencionó anteriormente, a cada módulo en la red se le puede considerar como un sólo proceso que espera alguna modificación en sus parámetros o entradas para iniciar su tarea. Para coordinar este flujo se utiliza una cola que contiene los módulos que sufrieron alguna alteración. El orden de esta cola depende de la posición de los módulos en la red

Es gracias a esta coordinación que se pueden eliminar redundancias y asegurar la correcta sincronización de la compleja interdependencia de la red. Esto es importante para aquellos módulos que dependen de los resultados de otros procesos y por consiguiente también tienen una dependencia temporal

A continuación se explican algunas de las técnicas de procesamiento que se usaron para realizar este trabajo:

Diversos módulos ejecutándose en un sólo proceso: La manera de realizar los cálculos eficientemente es ordenandolos de tal manera que se ejecutan primero aquellos con menos trabajo. Además se procura que en un proceso se ejecute más de un módulo. Esto se puede hacer cuando los módulos son sencillos y su actividad se encuentra ligada a otros Por ejemplo, como se verá adelante, los módulos *Set view* y *Geometry Viewer*, que se utilizan para obtener distintas perspectivas de la superficies del terreno, se pueden unir en un sólo proceso.

Ejecución de módulos en paralelo: Para aprovechar las ventajas del equipo de cómputo que se usa, como son estaciones de trabajo con multiprocesadores y medios heterogéneos de cómputo, hacemos que algunos módulos usen el procesamiento en paralelo. Esto es posible cuando no dependen de los datos de otros módulos. Por ejemplo los módulos *LatLon_a_UTM*, *Ejes* y el que calcula el mapa de colores, pueden ser ejecutados al mismo tiempo.

Comunicación remota de módulos: Para realizar el procesamiento remoto se necesitan mecanismos de comunicación para red y establecer la dependencia de la información entre aquellos módulos que residen en diferentes maquinas. Cuando se utiliza un módulo remoto, se crea un nuevo proceso

con el cual se controla la forma en que el módulo realizará los cálculos y con el cual se puede establecer una comunicación directa con *UNIX* que utiliza el comando *rsh* (*shell* remoto) que sirve para enviar la información por medio del protocolo *TCP/IP*.

Como ya se ha mencionado, los módulos de este trabajo pueden ejecutarse local o remotamente y pueden utilizar el mismo equipo de cómputo o bien diferente pero siempre utilizando *AVS* y *UNIX*. De esta manera cualquier módulo puede ser compilado, ligado y almacenado en un nodo remoto e incorporarlo en la red en cualquier momento.

Para que la comunicación remota sea clara en todo momento, se usa un formato de representación externa de datos. Esta consiste en un conjunto de datos binarios, que contienen las dimensiones de cada polígono y sus elementos, lo cual garantiza la posibilidad de representar geometrías en cualquier máquina.

Al ambiente en que se encuentran inmersos estos módulos y la forma de controlar sus procesos, se le conoce como ambiente de visualización modular (*AVM*) Este ambiente hace de la programación un medio para observar al sistema lo cual contrasta con otros ambientes de programación gráfica basados en símbolos (*Mathcad, Mathematica*) y con la programación tradicional de texto.

Este tipo de ambientes hacen uso de iconos para representar a los módulos con el fin de utilizar una forma gráfica de las llamadas que se realizan a los programas, estos iconos se agrupan en un algo parecido a un diagrama de flujo o red. Para mostrar esto, la figura 1.3 contiene el ambiente gráfico en donde se desarrolló la visualización que aquí se presenta.

1.2 Supercómputo

Para obtener un alto rendimiento del cómputo se pueden utilizar las instalaciones de computadoras que minimizan el tiempo que toma un programa desde que comienza a ejecutarse hasta que finaliza. Este periodo de tiempo puede reducirse al escoger diferentes algoritmos y ponerlos en marcha en diferentes arquitecturas de computadoras o identificar cuáles son los que consumen menos tiempo. De esta manera se puede obtener una gran reducción en el tiempo y utilizar eficientemente los recursos. Esto es lo que precisamente se busca al diseminar los diferentes módulos que utilizamos en la visualización

El alto rendimiento no se obtiene simplemente utilizando un procesador rápido. Es necesario utilizar los recursos de manera de explotar sus potencialidades. Por ejemplo, es cierto que un procesador de una supercomputadora es más veloz que el de una estación de trabajo, pero la ventaja real del procesador de la supercomputadora puede ser, o bien su capacidad de comunicarse con otros procesadores y llevar a cabo procesos en paralelo, o bien su capacidad de vectorizar. Los sistemas que tienen capacidades de supercómputo se tienen que afinar, en el sentido de establecer parámetros que permitan la explotación de alto rendimiento.

Para enfatizar este punto, la figura 1.4 ilustra la relación entre distintos tipos de computadoras. El cómputo distribuido, por si mismo, no garantiza un alto rendimiento puesto que es necesario contar con comunicaciones eficientes entre las máquinas. La manera de procesar datos en este trabajo utiliza el cómputo distribuido y, en el caso de algunos procesos, simultáneo. En este sentido es que se habla de paralelismo Entre los problemas que se encuentran en este tipo de procesos el más conflictivo es la heterogeneidad de los equipos utilizados.

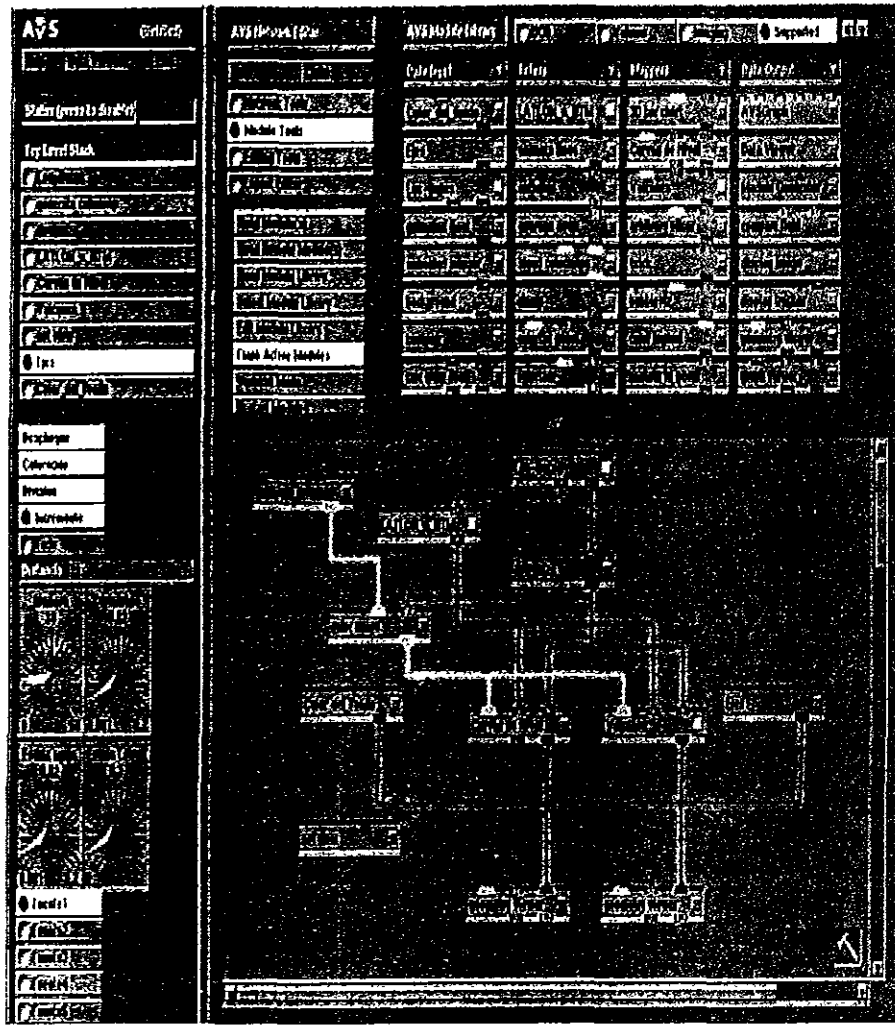


Figura 1.3: Ejemplo de un ambiente de visualización modular. Los módulos son los iconos rectangulares conectados entre sí para formar el programa visual o red.

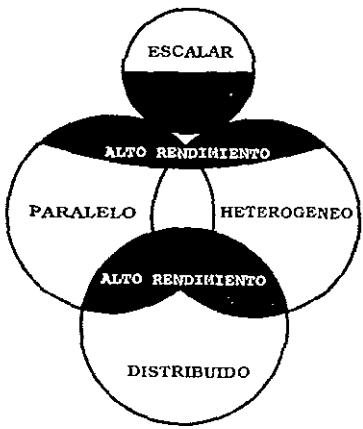


Figura 1.4: Relación entre los sistemas escalares, paralelos, distribuidos y heterogéneos con el alto rendimiento en cómputo.

Capítulo 2

Programación modular

Como se explicó en el capítulo anterior, la visualización ha tenido un desarrollo muy rápido debido a que cada día más, se generan enormes conjuntos de datos. Para entender su significado es necesario representarlos visualmente. Con este objeto se han utilizado los "Ambientes de Visualización Modular". Estos sistemas permiten conjuntar varios elementos como el diseño de sistemas, la graficación y el análisis de datos, de tal forma que no interfieran unos con los otros. Al final se pueden crear aplicaciones que realizan cálculos en tiempo real y pueden ser usadas interactivamente. Estos sistemas cuentan con algoritmos de graficación y visualización.

2.1 Ambiente de visualización modular

Una de las características de los sistemas de AVM es la programación visual, con la cual se pueden diseñar y desarrollar aplicaciones específicas. Por ejemplo, si se ha construido una nueva interfase, los nuevos elementos gráficos se pueden activar inmediatamente y probarse en el contexto de la aplicación final, ya que para poder conectar los valores obtenidos en los objetos gráficos con los parámetros de entrada, se hace uso de una serie de llamadas al sistema, en donde las llamadas son acopladas por medio de funciones.

En este trabajo se hace la distinción entre programación visual y visua-

lización de datos. La programación visual consiste en diseñar y ejecutar los programas de forma gráfica. La visualización de datos, como se mencionó antes, es la representación de los datos por medio de imágenes.

La diferencia que existe entre los lenguajes visuales y los textuales reside en que en los primeros se usan figuras y diagramas y en los segundos, cadenas de palabras que representan comandos. La primer ventaja al usar gráficas en lugar de texto, es la posibilidad de tener en todo momento una imagen de la estructura del programa y del flujo de información. Esta característica se puede explotar al momento de programar ya que los modelos visuales representan tanto a los cálculos como a sus resultados.

El diseño de los lenguajes visuales está dirigido básicamente a llevar a cabo los siguientes procesos:

- Control de la información.
- Operaciones sobre los datos.
- Representación de los datos.

Aunado a estos tres componentes, el sistema debe contar con una interfase gráfica. En los lenguajes visuales que usan iconos, el flujo de datos se representa utilizando únicamente grafos. Por otro lado, las operaciones pueden representarse por medio de pequeñas imágenes o iconos. En el caso del sistema que se presenta en este trabajo, se utilizan pequeñas cajas rectangulares que representan a los módulos.

El desarrollo de programas utilizando la programación visual requiere necesariamente de bibliotecas de graficación. Con ellas, la programación se simplifica ya que, por ejemplo, una ventana se puede generar con la llamada a una función.

El *AVS* tiene una serie de módulos preconstruidos que realizan funciones convencionales, sin embargo, para aplicaciones específicas, es necesario diseñar y programar módulos *ad hoc*. La programación de módulos se realiza

en lenguaje "C" y simplemente se incorporan a las bibliotecas a las que el AVS tiene acceso.

Los *AVM* pueden ser: modulares, que pueden ser usados para unir distintos bloques de código, modificar y ampliar el sistema con conjuntos de módulos y en ellos se puede establecer una equivalencia a la programación orientada a objetos. Adaptables, en los que se utiliza un entorno de programación visual y permiten manipular el sistema de manera interactiva. También pueden ser usados para la visualización. La visualización se puede utilizar como un prototipo para desarrollar aplicaciones que utilizan grandes cantidades de datos.

La visualización se lleva a cabo en las estaciones de trabajo de manera similar a un estudio fotográfico. En una escena se tienen los objetos, las luces y las cámaras. En la visualización las cámaras corresponden a los ojos del observador, los objetos a las imágenes que se estudian y las luces, como en fotografía, se usan para iluminar los objetos.

Los datos a visualizar pueden ser prácticamente cualquier cosa que se pueda manejar con las estructuras lógicas del lenguaje de programación "C". Así, se pueden usar estructuras de datos en forma de listas de apuntadores, listas ligadas, etcétera. En el caso de la visualización de la topografía del territorio nacional, se utilizó la técnica de apuntadores que permite el manejo de grandes cantidades de datos.

Una característica adicional de los *AVM* es que se utiliza de manera natural, la idea de "programación por demostración", ya que permiten la observación simulánea de los programas, de su funcionamiento y de los resultados. Además el proceso tradicional de edición, compilación y concatenación o ligadura de rutinas objeto, no se hace explícitamente, sino que el *AVM* se encarga de ello en cuanto se introduce una modificación al programa

En la figura 2.1 se muestra una pantalla típica del AVM que se utilizó en este trabajo

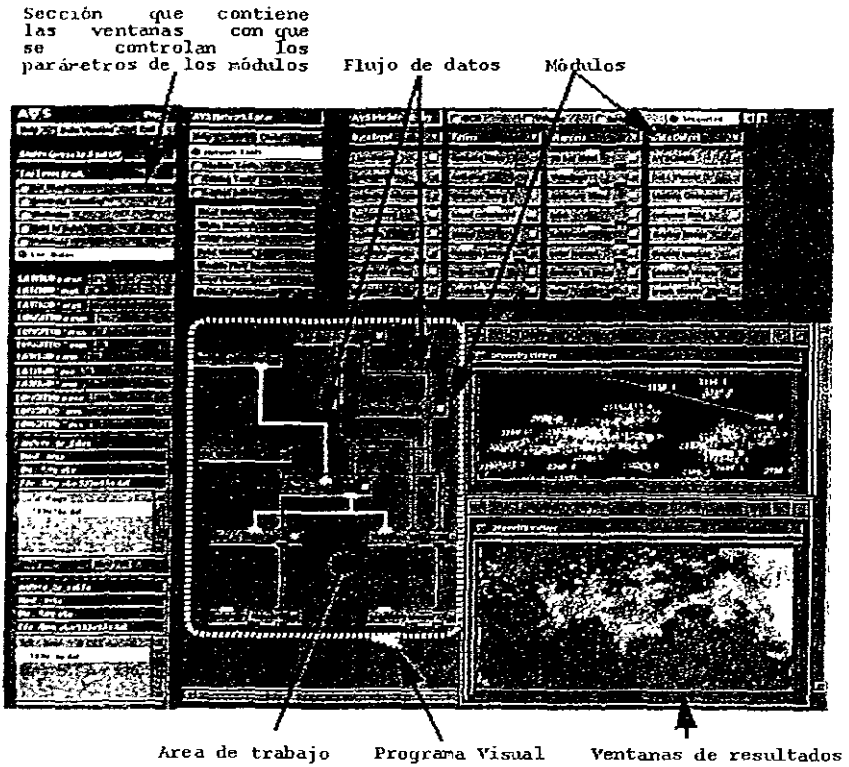


Figura 2.1 Elementos que componen a un ambiente de visualización modular

En muchas situaciones de programación o de visualización y análisis de datos es necesario interactuar con los diferentes módulos que constituyen el programa. Para lograr esta interacción es necesario capturar y procesar nuevos datos en forma inmediata y ajustar nuevos parámetros de visualización para identificar sus efectos en las imágenes o animaciones. Para que un programa visual se ejecute eficientemente de forma interactiva es necesario contar con resultados parciales en puntos estratégicos. Para lograr esto se almacenan en memoria lo cual los hace disponibles en cualquier momento de las ejecuciones y de manera inmediata Para evitar problemas relacionados con el acceso a datos no actualizados, es necesario, además, verificar si los parámetros de los módulos previos han sido modificados. En la práctica esto se lleva a cabo a base de iluminar los iconos de los módulos cuyos parámetros han cambiado. Este proceso de ilustra en la figura 2.2.

Finalmente, se podría decir que por la manera en que están diseñados, los programas visuales, producidos en los ambientes de visualización modular incorporan la capacidad de simular multiprocesamiento de datos, incluso en computadoras de un solo procesador escalar, debido a que los módulos se ejecutan de manera independiente y además se pueden tener varias redes de módulos que también se ejecutan de manera independiente.

Ejemplos de ambientes de visualización modular y de interfases gráficas, además del *AVS* utilizado en este trabajo, existen muchos, tanto del dominio público como comerciales tales como el *Khoros*, el *IDL*, el *XForms*, *Motif*, *Lesstif*, *Visual-Basic*, *Visual-C*, *Visual-Java*, *Delphi*, etcétera.

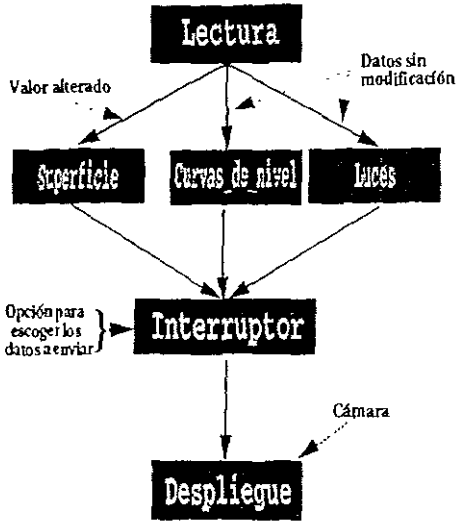


Figura 2.2: Esquema de un programa visual que condiciona el flujo de datos en el momento de escoger el tipo de entradas.

2.2 Relación que guardan los módulos que se utilizan.

Hasta ahora se ha explicado la relación que existe entre algunos de los módulos que se involucran en el trabajo, pero no su inserción en el sistema en general, por lo que acontinuación se describirá la estructura global del sistema. Haciendo un poco de memoria, la figura 1.2 muestra a todos los módulos que se utilizan, así como las ligas que mantienen entre algunos de ellos.

En la cima se encuentra el módulo *Lee_Datos* que se encarga de leer y ordenar en una malla los datos de la altimetría de tal forma que los sub-

secuentes módulos sólo se abocan a realizar cálculos más trascendentes. El funcionamiento de este módulo se describe con detalle en el Capítulo 3. Otro módulo que está en la cima es el *generate_colormap* encargado de generar una paleta de colores con el fin de proveer una gama de colores que ayude a distinguir los distintos valores de la altimetría en la superficie que representa a la topografía.

Inmediatamente abajo, se encuentran los módulos *LatLon_a_UTM* y *downsize* que como se observa están conectados a *Lee_Datos*. El *LatLon_a_UTM* se encarga de obtener la equivalencia de transformar las coordenadas geográficas $(longitud, latitud)$ a coordenadas cartesianas (x, y). En este caso, el nombre de la transformación utilizada es *UTM*. El módulo *downsize* es el encargado de decimar la malla que *Lee_Datos* genera, con el fin de no saturar la memoria de la máquina, porque ésto ocurre cuando se aumenta la densidad de la malla. En otras palabras, con este módulo se puede controlar el detalle de la topografía así como la velocidad del despliegue. Esto ayuda a sortear el dilema de obtener gran precisión en los cálculos de los parámetros geomorfológicos u obtener un despliegue y una manipulación de las imagenes más veloz.

Una vez que se obtiene el grado de resolución de la malla; que no es otra cosa mas que su dimensión y la ubicación en el globo terráqueo de cada uno de sus elementos, lo primero que se hace es enviar los datos al módulo *color range* con el fin de asignar a cada elemento de la malla un color conforme a su valor en altitud y al mapa de colores definido por *generate colormap*.

Posteriormente se envian en forma independiente a los módulos *Curvas_de_Nivel*, *Volcanes* y *field_to_mesh*, para realizar los cálculos más importantes, que para el caso del primer módulo son las curvas de nivel del terreno así como la superficie tridimensional que representa la topografía de la zona que se desea visualizar; el segundo se encarga de obtener en forma automática los parámetros geomorfológicos de los volcanes que estan en la zona, como son el diámetro de la base, el diámetro del cráter, la altura del

cráter sobre el terreno, la pendiente promedio de las laderas, el volumen del volcán y la comparación de la razón entre la altura y el diámetro de la base. El último únicamente obtiene una superficie tridimensional del terreno, al igual que el *Curvas_de_Nivel*, pero más rapido ya que no realiza cálculos muy complejos.

Por último se envían los resultados a un módulo *gometry viewer* que se encarga de desplegar los resultados que para todos los casos son geometrías compuestas de polígonos, en forma de imagenes. Como cada uno de los módulos *Curvas_de_Nivel*, *Volcanes* y *field_to_mesh* genera sus propios resultados se incluye varias veces. Por el contrario, solo basta con utilizar una sola vez al *gometry viewer* y despliega todos los resultados en una sola imagen. Esto proporciona independencia cuando únicamente se desea conocer la topografía de una determinada zona o una vez que se observó la región, obtener los parámetros de volcanes qué ahi se encuentran.

Con el fin de complementar la información que se obtiene en forma de imagen, se utiliza el módulo *Ejes* para mostrar una *Rosa_de_los_vientos* y obtener la orientación que la superficie tiene respecto a los puntos cardinales. El módulo *Color_del_fondo* como su nombre lo indica, cambia el color del fondo de la imagen con lo que se obtiene un mejor impacto visual. Finalmente el módulo *set_view* permite observar la superficie desde cualquier perspectiva.

Capítulo 3

Desarrollo del sistema

En este capítulo se explica cómo se desarrolló el programa para manejar información topográfica de una manera gráfica. Como se mencionó antes los datos originales están constituidos por arreglos de alturas sobre el nivel del mar que corresponden a puntos en coordenadas geográficas obtenidos cada tres segundos de arco y que cubren todo el territorio nacional. Estos datos se obtuvieron de un disco compacto publicado por el *INEGI* que contiene 255 archivos que contienen un grado cuadrado cada uno y que se elaboraron a partir de las cartas topográficas escala $1:250,000$, también del *INEGI*. En la figura 3.1 se muestran los 255 cuadrantes geográficos en que se dividió a la República para representar la topografía en forma digital.

A este tipo de archivos se les conoce como Modelos de Elevación Digital(MED), que como se mencionó en el parrafo anterior, contiene las elevaciones de un terreno de una zona del globo terraqueo, y representadas en una malla dividida en intervalos. Estos intervalos siempre deben estar referenciados con algun sistema de coordenadas geograficas. Los más usuales son el basado en latitud-longitud y el Universal Transversa de Mercator (*UTM*). Entre más densa sea la malla, se podra obtener un mayor detalle por que sólo se cuenta con el valor de z (valor de elevación). Para calcular la posición que ocupa cada valor en la Tierra, se debe inferir desde la esquina inferior izquierda del archivo, que como se verá más tarde se obtiene del nombre de

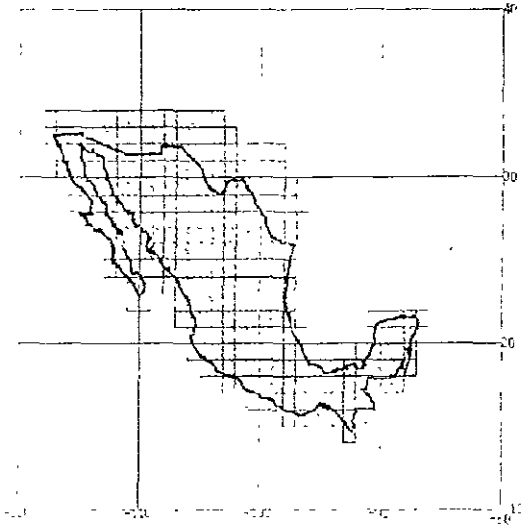


Figura 3.1: División de la República en 255 cuadrantes geográficos.

cada uno de los archivos. Los MED pueden contener información civil tales *cómo vías de comunicación* y edificios por eso hay que aclarar que no son imagenes obtenidas de un escáner ni mucho menos mapas de *bits*, tampoco contiene contornos de elevaciones. Únicamente contiene el valor promedio de la elevación (media ponderada)

En cada archivo se tiene la información de un grado de latitud por un grado de longitud en una matriz de 1,442,401 alturas del terreno. Las coordenadas geográficas de cada elevación se infieren por su posición dentro de la matriz

Los datos se recuperaron del disco compacto mediante un programa escrito *ad hoc* que los decodifica, los ordena y rota de tal manera que las columnas representan direcciones norte-sur y los renglones este-oeste. Con estas transformaciones elementales se corrige el problema que tiene el programa que

provee el *INEGI*, en el que las imágenes que se obtienen son imágenes espejo del terreno real.

3.1 Proceso preliminar

Para sistematizar la identificación de archivos, se creó un arreglo de caracteres en donde los nombres de los archivos que contienen los datos topográficos indican la latitud y la longitud separadas por "n" y "o" que indican norte y oeste respectivamente.

Así, los nombres de los archivos indican la latitud y la longitud a partir de las cuales se tiene la secuencia de puntos en coordenadas geográficas a los que corresponde la altura sobre el nivel del mar. Estos se puede considerar que son valores asociados a una malla regular, la figura 3.2 ilustra los nodos (alturas) que conforman la malla.

Para crear la malla se deben especificar las dimensiones iniciales de un *arreglo*, que para comenzar, debe ser, como mínimo, de 1201×1201 elementos, para reservar el espacio de memoria suficiente para almacenar un grado cuadrado. Posteriormente se escogen la longitud y la latitud de la zona que se quiere estudiar y se comienza la lectura de los datos

Para evitar la limitación de representar solamente un grado cuadrado, se creó un algoritmo que durante la lectura reasigna la dimensión de los arreglos para poder acceder a archivos de regiones vecinas y así poder construir regiones mayores que las correspondientes a un grado cuadrado o bien regiones que se encuentran en la intersección de meridianos y paralelos que delimitan grados cuadrados contiguos.

El problema de la reasignación de memoria y por tanto de las dimensiones de los arreglos es uno cuando las regiones a unir colindan en el sentido horizontal (longitud) y otro cuando lo hacen en el sentido vertical (latitud). La malla es entonces de $n \times m$ elementos, en donde $(n, m) \geq 1201$ y n corresponde a la longitud y m la latitud.

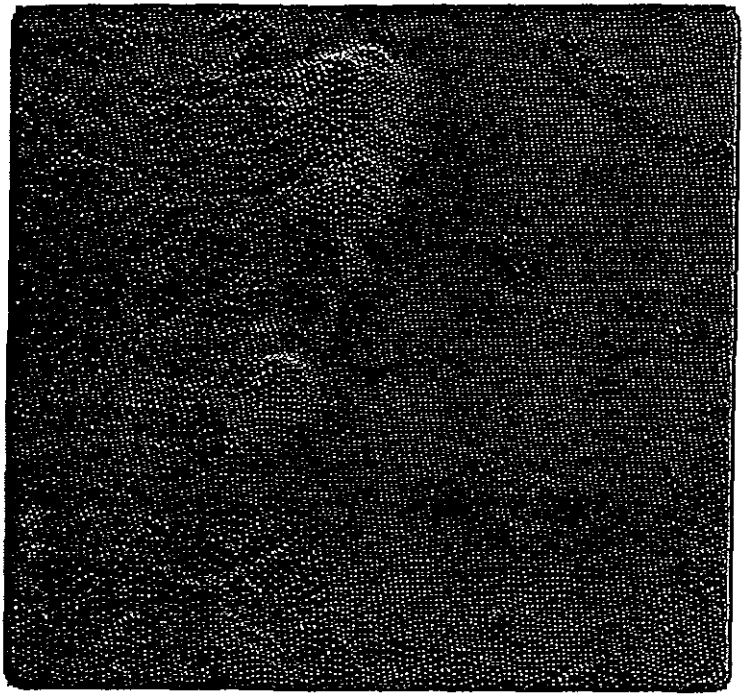


Figura 3.2: Imagen que muestra las alturas representadas por los puntos de colores, con que se construye la malla. La región que se observa comprende al Pico de Orizaba y a la Caldera de los Humeros en Veracruz.

Esto último se ilustra en la figura 3.3 en donde se muestran diferentes zonas contenidas en diferentes archivos. Como se puede observar la zona global está constituida por el alineamiento adecuado de las subzonas que la componen.

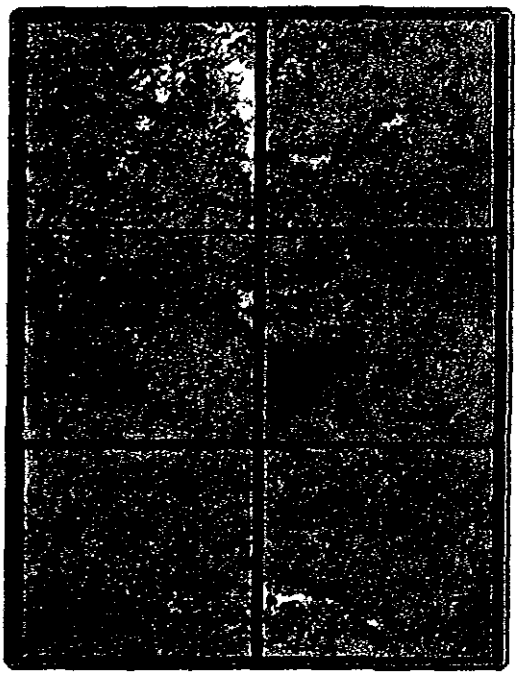


Figura 3.3: Imágenes que se usan para integrar una región muy extensa a partir de regiones menores

La manera de resolver este problema es la siguiente: Se crean dos ciclos, en donde el interior representa la longitud y el exterior la latitud. Ambos ciclos van desde 0 hasta la dimensiones máximas de longitud y latitud. Estas dos dimensiones pueden ser diferentes y se calculan dependiendo de las dimensiones de la zona que se quiere recuperar.

Para unir dos mallas en longitud, el problema no es complicado, puesto que basta leer el primer archivo y, a continuación, leer el que le sigue, por ejemplo, si se lee el archivo `19n100w.dat`, a continuación se debe leer el archivo `19n99w.dat`. El problema surge cuando se quieren unir dos mallas contiguas en latitud, ya que primero hay que leer 1201 campos del primer archivo, e inmediatamente después leer del archivo en el que continúa esa región hacia el norte o hacia el sur, con otros 1201 campos, para posteriormente regresar a leer otros 1201 campos del primer archivo y repetir el proceso hasta agotar la extensión en latitud. Si se quiere ver una zona que involucre más de tres archivos, obviamente primero hay que leer 1201 campos del primero, después 1201 del segundo y por último 1201 del tercero, hasta terminar el proceso de integración de la imagen.

Después de arreglar el orden en que se accederá a los archivos, se diseñó un algoritmo para que a cada altura le asignara su localización geográfica mundial, esto es, a todos y cada uno de los datos topográficos se les dió una localización tanto en longitud como en latitud en grados, minutos y segundos, de esta forma, es posible obtener después zonas más pequeñas.

Así, al tener identificados todos los datos, únicamente se necesita escoger el área de interés y representarlos gráficamente. En la figura 3.4 se muestra la imagen completa que resulta de unir todos los archivos de la figura 3.3. Esto además facilita, como es natural, la manipulación de la imagen.

Para obtener subconjuntos de la malla creada con el procedimiento anterior, se usa un módulo de *AVS* que cambia el número de elementos de la malla a partir de submuestras de datos. Esto es, extrae un elemento n del conjunto a lo largo de cada dimensión, en donde n es el valor obtenido por medio de un parámetro que se proporciona. Esta técnica preserva el aspecto de la relación de los datos de entrada, ya que la salida es de la misma dimensión que la de entrada, pero el número de elementos en cada dimensión se reduce. Para ilustrar esto, en la figura 3.5 se muestra cómo es reducido un conjunto. Cada elemento es representado por un punto y únicamente

Figura 3.4: Imagen completa a partir de todos los archivos ya unidos

los puntos encerrados en círculos serán los que se envíen como salida a los subsiguientes módulos del programa.

La orientación en que están las imágenes es la usual: *El norte hacia arriba* y el este hacia la derecha. Las elevaciones almacenadas en un registro dado corresponden a puntos sobre el mismo meridiano, ordenados de sur a norte; así, cada registro representa un perfil del terreno, orientado de sur a norte. Los registros se encuentran ordenados de oeste a este. Todas las elevaciones se representan con enteros binarios de 16 bits, justificados a la derecha con el bit de orden más alto indicando el signo. Los valores permitidos están en el intervalo $[-32767, +32767]$ metros sobre el nivel del mar. Los valores

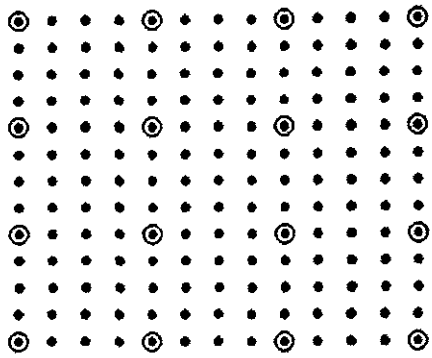


Figura 3.5: Obtención de subconjuntos de datos.

desconocidos se indican con todos los bits prendidos. Los valores negativos se ignoran.

La figura 3.6 muestra el despliegue de la pantalla del módulo que realiza la lectura de los datos, proporcionando además su localización geográfica. Al cambiar los valores que aparecen dentro de los rectángulos que se observan en la imagen, se puede obtener una zona más chica, siempre y cuando esté contenida en la malla principal.

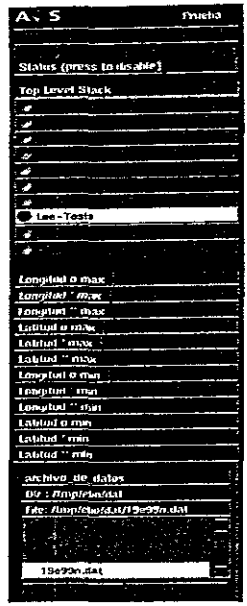


Figura 3.6: Imagen del módulo *Lee_Datos*

3.2 Representación digital

Debido a que la Tierra es aproximadamente esférica, cualquier representación de ésta en un plano, genera una distorsión, que en zonas muy pequeñas es insignificante, pero cuando se trata de regiones muy vastas como obtiene una gran distorsión. Es por eso, que durante varios siglos, se ha trabajado para representar la supercicie curva de la Tierra en mapas planos por medio de varios esquemas, conocidos con el nombre de proyecciones para mapas. Cada una de estas proyecciones tiene ventajas y desventajas y su utilización depende de los propósitos que se persiguen.

Para convertir las coordenadas geográficas de las alturas del terreno a

una forma adecuada, se utilizó la representación Universal Transversa de Mercator (UTM). Lleva el nombre de transversa porque el orden de los datos dentro de la malla es *en forma ascendente* de sur a norte y de este a oeste a lo largo del ecuador. Con esta representación es posible realizar mediciones de distancias, calcular áreas, etcétera.

La proyección UTM es un modelo matemático que se utiliza para inferir posiciones en la superficie curva de la Tierra sobre una superficie plana. Se forma proyectando las coordenadas de la Tierra sobre un cilindro tangente a la Tierra en un meridiano de longitud. Es una proyección comforme, lo que implica que se mantiene la forma de areas pequeñas. Los únicos parametros necesarios para definir esta proyección son la longitud del meridiano central y la latitud de referencia, aunque también pueden especificarse un factor de escala, el falso este, y el falso norte. A medida que uno se aleja del meridiano central, la distorsión aumenta. Esta proyección se utiliza para minimizar la distorsión que se produce en mapas de areas que se extienden de norte a sur.

La representación gráfica de los datos requiere que se mapee un grado cuadrado a una área cuadrangular. En virtud de que los datos están originalmente en coordenadas geográficas, las distancias entre dos puntos sobre dos meridianos dados son menores en el hemisferio norte y mayores en el hemisferio sur.

La conversión de coordenadas se hace sobre cada grado cuadrado por separado, utilizando las siguientes expresiones:

$$x = \frac{111276.1806\cos\phi\Delta\lambda}{(1-0.0067686579sen^2\phi)^{1/2}} + \frac{5.619\cos^3\phi(\Delta\lambda)^3(1-\tan^2\phi+0.006815\cos^2\phi)}{(1-0.0067686579sen^2\phi)^{1/2}} \qquad (1)$$

$$y = 6332500.489(0.0175424666\mathrm{A} - 0.0025601sen2\mathrm{A} + 0.0000027sen4\mathrm{A}) \quad (2)$$

Donde:

$$\mathrm{A} = \phi + (0.0087861165\tan\phi\cos^2\phi(\Delta\lambda)^2(1 - 0.0067686579sen^2\phi))$$

En donde λ y ϕ son las coordenadas geográficas del punto a transformar, medidas en grados y λ_0 es la longitud del meridiano central de la región. Con estas expresiones se obtiene finalmente $xe = fa + x$ si el punto está al este del meridiano central o $xo = fa - x$ si se encuentra al oeste. En estas últimas expresiones $fa = 500000$ es la falsa abscisa del meridiano central. Para una discusión más detallada de estas transformaciones se puede consultar por ejemplo la referencia[10](Bribiesca] que se cita al final.

Para realizar esta transformación se programó un módulo en lenguaje "C" que se incluyó en el *AVS*. La figura 3.7 muestra la ventana en donde se ejecuta este procedimiento.

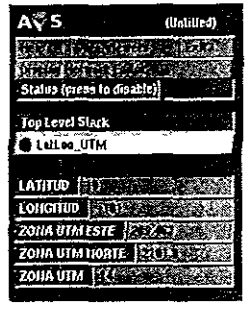


Figura 3.7: Imagen del módulo *LatLon_a_UTM*

Después de la transformación de coordenadas de los elementos que se desean visualizar se construye una malla de puntos (x, y) que corresponde a la localización de las alturas que se leyeron previamente.

Los coeficientes que se encuentran en las ecuaciones (1) y (2) corresponden a constantes establecidas por el cambio de coordenadas. Utilizando las ecuaciones (1) y (2) cualquier punto definido por sus coordenadas geográficas es transformado a coordenadas *UTM*.

Como ejemplo, en la figura 3.8 se proporcionan las coordenadas UTM de la cuenca de México utilizando el módulo *LatLon_a_UTM*.

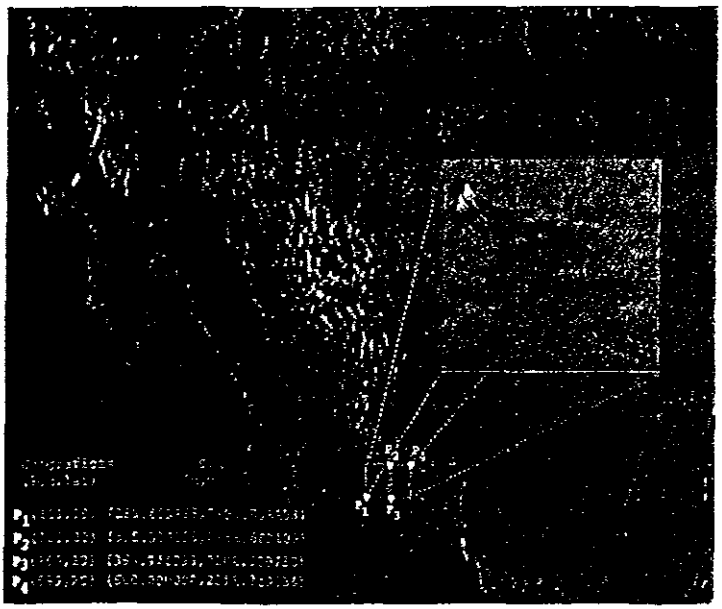


Figura 3.8 Ejemplo de conversión de coordenadas geográficas (longitud latitud) a coordenadas *UTM* (x,y).

3.2.1 Despliegue gráfico

El siguiente paso, que se tiene que dar después de leer todos los datos, es crear la malla y transformarla en una superficie para finalmente desplegarla en una imagen, esto se logra de dos maneras.

Módulo *field_to mesh*

La primera es usar un módulo llamado *field_to_mesh*, este convierte a un arreglo bidimensional en una superficie tridimensional en donde cada elemento es mapeado a un punto en una base plana. La altura de la malla sobre cada punto en este plano es proporcional al valor escalar del campo, además hay que escalar el mapa de colores, para que de esta forma, de acuerdo al valor

de cada dato se le proporcione una coloración.

El procesamiento para proporcionar esta coloración es un aspecto fundamental en la visualización, puesto que proporciona una estructura de datos para que sea usada por *módulos que necesiten transformar* sus datos en valores de color. La escala para hacer la coloración es la normal (0 a 255), pero uno mismo puede crear su propia escala de valores. En este caso el mapa de colores usa el *hue-saturation-bryhtness*(*HBS*) para crear el modelo de color. A continuación se dan los valores posibles de cada uno de los parámetros que se utilizan en el mapa de colores.

Matiz

0.00=rojo

0.16=amarillo

0.33=verde

0.50=cyan

0.66=azul

0.83=magenta

Saturación

0.00=blanco

1.00=matiz

Brillo

0.00=negro

1.00=matiz

Módulo *Curvas_de_nivel*

El más importante, después del módulo que lee los datos, es el que obtiene las curvas de nivel del terreno, este módulo tiene la capacidad de obtener diferentes números de niveles, dependiendo de el número que consideremos adecuado, puede obtener la altura de cada nivel, mostrarla y cambiar la escala para las alturas. Todo esto se encuentra representado en una superficie tridimensional. En la figura 3.9 se muestra la pantalla de este módulo.

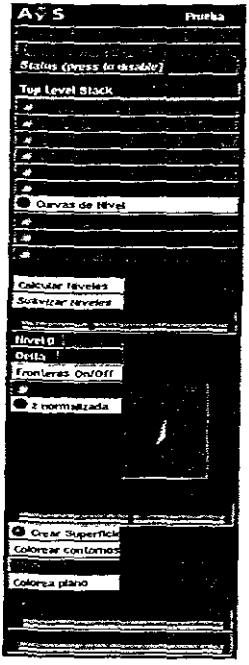


Figura 3.9: Imagen del módulo *Curvas_de_Nivel*

Para lograr lo anterior, se generaron contornos poligonales para realizar un despliegue veloz: se generó un único intervalo para los contornos que existan en un intervalo

El algoritmo para obtener estos polígonos es el siguiente

Se obtienen los valores de la altura más pequeña y la más grande

```
/* Obtencion de la altura mas alta y baja */
maxfld $=$ (char *) &maxval;
minfld $=$ (char *) &minval;
```

Se crea un arreglo tridimensional, los dos primeros valores son para definir el contorno poligonal (x,y), y el tercero es para guardar la altura (z) del polígono.

```
static float lineas_poligono[LIMITEMAX][3],tmp_poly[LIMITE][3],
```

A continuanción se debe calcular el promedio para que indique los diferentes intervalos por medio de un número de niveles, ya que apartir de este valor, es como se va a dividir el conjunto de polígonos que representa a las curvas de nivel.

```
promedio = (maxval - minval) / (float) (1+niveles);
```

El módulo *Curvas_de_Nivel* crea un mapa de contornos tridimensional, a partir de valores escalares en *2D*, para representar isoalturas de la superficie de terreno, además de generar una superficie tridimencional semejante a la que se obtiene con el módulo *field_to_mesh*, también se pueden establecer manualmente los isocontornos de las alturas máxima y mínima por medio

de niveles, en donde el nivel $N = 0$ y una cierta ΔN sirven para cubrir homogéneamente todo el intervalo. Cuando se escoge la opción de suavizar los niveles, estos se redondean a valores enteros, múltiplos de 1, 2 o 5 y de acuerdo con un factor de escala decimal.

Una vez que se ha dividido y se han obtenido las alturas que van a servir como intervalo en la búsqueda de los posibles valores que puedan formar una curva de nivel, recorremos la malla en forma horizontal y cuando se llega al final de un renglón, se continúa con el siguiente.

```
#define DIRECCION(i,j) (j)*dim0+(i)
```

Por medio de ésta y otras funciones se realiza la búsqueda de las alturas que están contenidas en un dominio definido por los intervalos anteriormente establecidos. Una vez que se encuentra, se realiza una búsqueda en las localidades vecinas y en sentido de las manecillas del reloj, para ilustrar esta búsqueda, se presenta la figura 3.10.

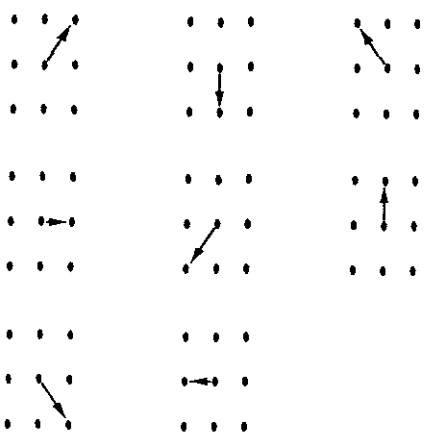


Figura 3.10 Comparaciones posibles con las localidades vecinas.

En la figura 3.11 se ejemplifica la creación de una curva de nivel, la búsqueda en forma breve consiste en encontrar una altura que esté en alguno de los intervalos que previamente fueron calculados, una vez que se localiza y ubica, se obtienen dos localidades vecinas en donde la primera señala a la localidad vecina más cercana en la malla y la segunda es el valor más cercano dentro del polígono que describe a la curva de nivel.

```
/* */
p0 = DIRECCION(i, j); p1 = DIRECCION(i + 1, j + 1); p2 = (posicion ?
DIRECCION(i + 1, j) : DIRECCION(i, j + 1));
```

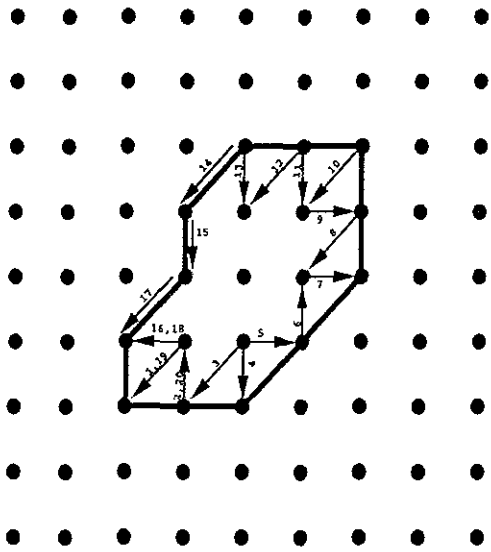


Figura 3.11: Búsqueda para construir una curva de nivel

A continuación se determina, si corresponde a una de las ocho opciones (ver figura 3.10) que valida la inclusión de dicha altura en la curva de nivel.

El siguiente segmento de código muestra cómo se implementó al procedimiento anterior.

```
switch (codigo_op) {
   case 0:
       break; /* ningun umbral cruza aqui */
   case 1: case 2: case 4: case 7:
       fprintf(stderr, "situacion imposible");
       break; /* ningun # suelto cruza */
   case 3:
       obtener_puntos_de_curvas(&longitud, p0, p1, p2, inf, nivel);
       if (posicion) {
           poligono_continuo(&longitud,i+1,j,MINIMO,1,inf,nivel);
           if (longitud > LIMITEMAX-1) break;
           poligono_inverso(longitud);
           poligono_continuo(&longitud,i,j,MINIMO,2,inf,nivel);
           if (longitud > LIMITEMAX-1) break;
       } else {
           poligono_continuo(&longitud,i,j+1,MAXIMO,1,inf,nivel);
           if (longitud > LIMITEMAX-1) break;
           poligono_inverso(longitud);
           poligono_continuo(&longitud,i,j,MAXIMO,2,inf,nivel);
           if (longitud > LIMITEMAX-1) break;
       }
   case 5.
       obtener_puntos_de_curvas(&longitud, p2, p0, p1, inf, nivel);
       if (posicion) {
           poligono_continuo(&longitud,i,j,MINIMO,2,inf,nivel);
           if (longitud > LIMITEMAX-1) break;
           poligono_inverso(longitud);
           poligono_continuo(&longitud,i,j-1,MINIMO,0,inf,nivel);
           if (longitud > LIMITEMAX-1) break;
       } else {
```

```
        longitud$=$poligono_continuo(&longitud,i,j,MAXIMO,2,inf,nivel);
          if (longitud > LIMITEMAX-1) break;
          poligono_inverso(longitud);
        longitud$=$poligono_continuo(&longitud,i-1,j,MAXIMO,0,inf,nivel),
          if (longitud > LIMITEMAX-1) break;
      }
      break;
  case 6:
      /* marca con un vector el inicio del poligono    */
      /* que representara una curva de nivel          */
      /* usando la funcion poligono inverso           */
      obtener_puntos_de_curvas(&longitud, p1, p2, p0, inf, nivel);
      if (posicion) {
          poligono_continuo(&longitud,i,j-1,MINIMO,0,inf,nivel);
          if (longitud > LIMITEMAX-1) break;
          poligono_inverso(longitud);
          poligono_continuo(&longitud,i+1,j,MINIMO,1,inf,nivel);
          if (longitud > LIMITEMAX-1) break;
      } else {
          poligono_continuo(&longitud,i-1,j,MAXIMO,0,inf,nivel),
          if (longitud > LIMITEMAX-1) break;
          poligono_continuo(&longitud,i,j+1,MAXIMO,1,inf,nivel);
          if (longitud > LIMITEMAX-1) break;
      }
      break;
  default:
      break;
  }
```

En ocasiones no se encuentra ningún valor que esté contenido en el intervalo de altura que se esta trabajando, por lo que se utiliza a un método de interpolación lineal por medio del uso de promedios pesados, para estimar las alturas intermedias:

$$Z_2 \; \overset{d_2}{- - - - - - - - - - - -} \; Z_\lambda \; \overset{d_1}{- - - - - - - - - - - -} \; Z_1$$

$$Z_\lambda = \frac{Z_1 d_2 + Z_2 d_1}{d_1 + d_2} \qquad (3)$$

Para la búsqueda en cuatro direcciones, a partir de un punto se recorre una abscisa dada en sentido positivo hasta encontrar una curva de nivel Z_1.

Se recorre ahora la abscisa en sentido negativo hasta encontrar una curva de nivel Z_2.

Con estos dos valores y con el cálculo de sus distancias respectivas, se calcula el valor de Z_λ usando la ecuación (3).

Si se repite el mismo proceso, pero en el sentido de las ordenadas, se encontrará otro valor para Z_ϕ con la elevación del punto deseado. Entonces, se promedian Z_λ y Z_ϕ, el método puede ser ampliado dependiendo el grado de resolución que se desee.

Módulo *Rosa_de_los_vientos* (ejes cardinales)

Otro módulo que se desarrolló, es el que muestra una rosa de los vientos, con el fin de indicar en la imagen que se obtiene de la topografía los ejes cardinales norte y este. En este módulo se pueden modificar diversos parámetros como son: el aumento de la longitud de los ejes para ajustarlos al tamaño de la superficie que define a la topografía, el número de divisiones en los ejes, su escala y etiquetas. En la figura 3.12 se muestra la pantalla de este módulo, así como los parámetros que pueden modificarse.

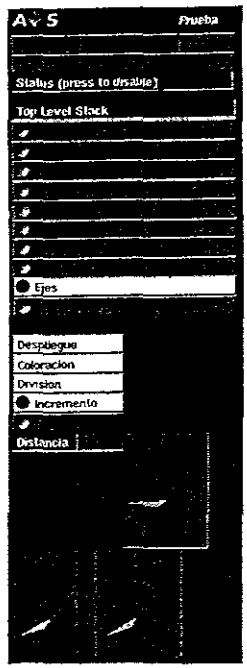


Figura 3.12: Imagen del módulo *Rosa_de_los_vientos* (ejes cardinales)

Para construir los ejes, se utilizaron las bibliotecas *GEOMcreate_obj* y *GEOMcreate_label* incluidas en *AVS* para hacer los distintos objetos geométricos a partir de triángulos, cuadrados y líneas. Para dibujar estos objetos, se cuenta con un arreglo bidimensional en donde se especifican las coordenadas de los vértices así como una lista que contiene el orden que deben llevar las conexiones. El primer entero en la lista contiene el número de vértices del primer polígono que define al polícdro. Enseguida se encuentran n enteros consecutivos (comenzando en 1), para representar a los elementos contenidos en el arreglo de vértices. Después, en el último índice del primer polígono se encuentra un entero, que se utiliza para representar el número de vértices del segundo polígono. Este patrón continúa hasta que el valor de

n es cero, con lo que se indica el fin de la lista.

Las etiquetas que se utilizan en el eje de las Z se pueden obtener de los valores contenidos en la malla, o hacer que el valor mínimo sea cero y el máximo será el valor más grande que se obtiene en el parámetro de la escala. Para el primer caso, ambos valores de escala se obtienen al utilizar como entrada una liga que provenga del módulo *Lee_Datos*.

Este módulo es opcional porque sólo se puede utilizar con los módulos que crean una superficie tridimensional

A continuación se muestran tres ejemplos de las visualizaciones que se pueden obtener con este sistema:

La figura 3.13 muestra la cuenca de México y sus alrededores.

Figura 3.13: Imagen que muestra la topografía de la cuenca de México ($120 \times 150 km^2$). (1) Volcán Popocatepetl (2) Volcán Iztaccihuatl (3) Volcán Ajusco (4) Sierra de las Cruces (5) Sierra de Guadalupe

La figura 3.14 contiene dos imágenes en donde la primera muestra la zona comprendida entre los meridianos $95^{o}.0',0''$ y $102^{o},14'.18''$ de longitud oeste y los paralelos 18^{o} $0'$ $0''$ y 21^{o} $6'$ $6''$ de latitud norte.

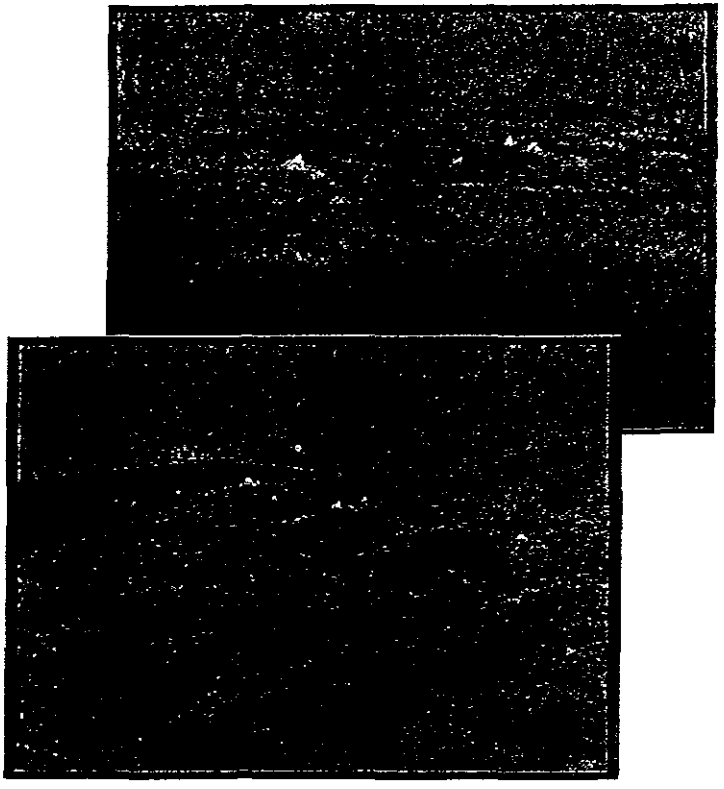


Figura 3.14. Perspectiva de la región que comprende los meridianos 95^{o} $0'$ $0''$ y $102^{o}14'$ $18''$ de longitud oeste y los paralelos 18^{o} $0'$ $0''$ y 21^{o} $6'$ $6''$ de latitud norte.

La figura 3.15 muestra la misma región pero incorporando las curvas de nivel de la zona.

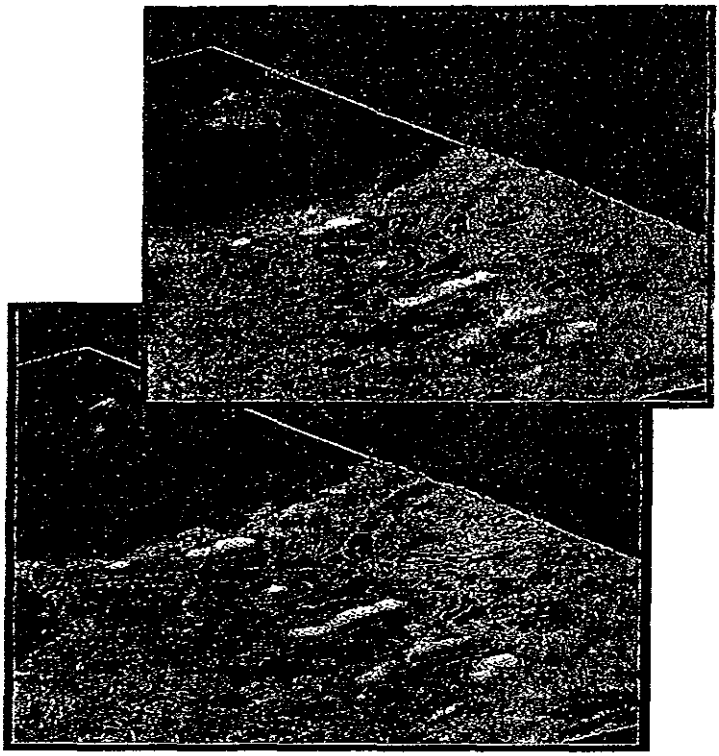


Figura 3.15. Perspectiva de la región que comprende los meridianos 95° 0′ 0″ y 102° 11′ 18″ de longitud oeste y los paralelos 18° 0′ 0″ y 21° 6′ 6″ de latitud norte; pero incorporando a las curvas de nivel que corresponden a esta zona

En todas las imágenes se observa la interacción de todos los módulos que se desarrollaron para este trabajo, incluyendo algunos que *AVS* trae implícitamente. Hay que resaltar, que estas imágenes se pueden rotar, escalar y trasladar dentro de la ventana que se utiliza para el despliegue, de está forma se pueden observar todas las perspectivas posibles.

3.2.2 Cálculo de la pendiente y el aspecto

Otro módulo importante es el que permite calcular la pendiente y el aspecto de la topografía, básandose en los valores vecinos al punto en el que se desea determinar el gradiente de su pendiente (máxima tasa de cambio en altitud) y la pendiende del aspecto (conseguir la dirección de la pendiente) de la elevación. Las fórmulas básicas son:

$$tan(pendiente) = \sqrt{(\frac{\partial z}{\partial x})^2 + (\frac{\partial z}{\partial y})^2}$$

$$tan(aspecto) = \frac{\frac{\partial z}{\partial x}}{\frac{\partial z}{\partial y}}$$

En donde $\frac{\partial z}{\partial x}$ y $\frac{\partial z}{\partial y}$ son las derivadas parciales en las direcciones del lado este-oeste y norte-sur correspondientes. Utilizando el siguiente método numérico es como se estiman estas derivadas.

$$[\frac{\partial z}{\partial x}]_{y,x} = \frac{Z_{y-1,x-1} + 2Z_{y,x-1} + Z_{y+1,x-1} - Z_{y-1,x+1} - 2Z_{y,x+1} - Z_{y+1,x+1}}{8\Delta x}$$

$$\left[\frac{\partial z}{\partial y}\right]_{y,x} = \frac{Z_{y-1\,x-1}+2Z_{y-1,x}+Z_{y-1\,x+1}-Z_{y+1,x-1}-2Z_{y+1,x}-Z_{y+1,x+1}}{8\Delta y}$$

La variable $Z_{y,x}$ es el valor de la elevación en el renglón y la columna x, Δ es el espacio que existe entre un dato y otro, que en el caso de la malla que se trabaja, es de 87.5 metros aproximadamente, así Δx es el espacio este-oeste y Δy el norte-sur. Esto último se puede expresar más claramente con la siguiente matriz de coeficientes:

$$\frac{\partial z}{\partial y} = \frac{\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}}{8\Delta x} \qquad \frac{\partial z}{\partial y} = \frac{\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}}{8\Delta y}$$

La sección de código que realiza estos cálculos es la siguiente.

```
pendiente(inf,pendientes)
AVSfield_float *inf;
AVSfield_float **pendientes;
{
/*                                        */
/* Procedimiento para calcular la pendiente del terreno      */
/*                                        */
int   ren, col, pos0, pos1, pos2, pos3, resolucion$=$1;
double az, bz;

for(ren$=$1; ren < dim0-2; ren++)
{
 for(col$=$1; col < dim1-2; col++)
 {
```

```
        pos0 = (col+1)+(ren*dim0);
        pos1 = (col-1)+(ren*dim0);
        pos2 = col+(dim0*(ren+1));
        pos3 = col+(dim0*(ren-1));
        az$= (float)(inf-> data[pos0]-inf-> data[pos1])/ (2.0 * resolucion);
        bz$= (float)(inf-> data[pos2]-inf-> data[pos3])/ (2.0 * resolucion);
        I2D(*pendientes,ren,col) = 360*atan(pow((pow(az,2.0)+pow(bz,2.0)),0.5))/
      (2.0*PI);
        I2D(*pendientes,ren-1,col-1) = 100*(pow((pow(az,2.0)+pow(bz,2.0)),0.5));
    }
   }
  }
```

Hay que hacer notar, que las variables *az* y *bz* se utilizan para retener temporalmente los valores intermedios que posteriormente se utilizarán en la expresión que calcula los datos finales. Al calcular la pendiente se usa la versión *atan*() que utiliza un solo argumento produciendo ángulos en un intervalo de 0^o a 90^o, sin embargo cuando el terreno tiene un relive muy bajo, estos ángulos no pueden dar información sobre la pendiente. Una modificación que puede hacerse es multiplicar el resultado por 10 y obtener diez tantos de grado. Otra opción es remplazar la funcion de *atan*() por la de *tan*()

En cambio, para calcular el aspecto se utiliza la version *atan*() con argumento doble, produciéndose ángulos en un intervalo de 0^o a 360^o, pero hay que tener cuidado por que el aspecto en ocaciones es indefinido, para el caso en que *az* o *bz* son cero (cuando el terreno es plano), por lo que hay que limitar los cálculos a ángulos que cubran el intervalo de 1^o a 360^o grados. El codigo para calcular el aspecto es el siguiente.

```
aspecto(inf,aspecto)
AVSfield_float *inf;
AVSfield_float **Aspecto;
```

```
{
/*                                  */
/* Procedimiento para calcular el aspecto del terreno      */
/*                                  */
int   ren, col, pos0, pos1, pos2, pos3, resolucion$=$1;
double az, bz;

for(ren = 1; ren < dim0-2; ren++)
{
 for(col$=$1; col < dim1-2; col++)
 {
   pos0 = (col+1)+(ren*dim0);
   pos1 = (col-1)+(ren*dim0),
   pos2 = col+(dim0*(ren+1));
   pos3 = col+(dim0*(ren-1));
   az = (float)(inf-> data[pos0]-inf-> data[pos1])/ (2.0 * resolucion),
   bz = (float)(inf-> data[pos2]-inf-> data[pos3])/ (2.0 * resolucion);
   I2D(*Aspecto,ren,col)$=$360*atan(az,bz))/(2.0*PI);
   I2D(*Aspecto,ren-1,col-1)$=$100*(pow((pow(az,2.0)+pow(bz,2 0)),0.5));
 }
}
}
```

La figura 3.16 muestra una imagen con las pendientes del terreno que se obtienen al aplicar a la malla que contiene la altimetría de la cuenca de México las fórmulas de la pendiente y el aspecto descritas en los párrafos anteriores. Los datos que se obtienen se almacenan en otra malla regular en donde cada punto, así como su ubicación, corresponde a la pendiente de la altura (x, y) contenida en la malla de la altimetría.

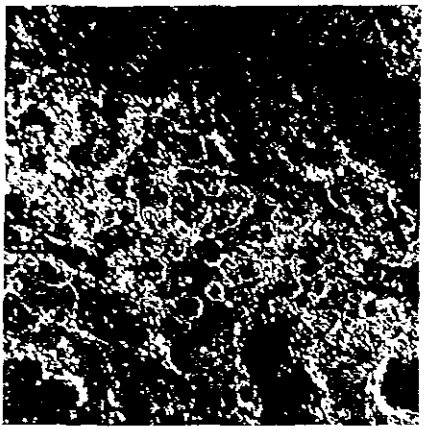


Figura 3.16: Imagen que muestra un mapa de las pendientes de la cuenca de México

Módulo *Volcanes*

El módulo más importante, en cuanto a cálculos se refiere, es el de *Volcanes*, con el que se obtienen el diámetro de la base, el diámetro del cráter y el volumen del volcán. Para obtener estos datos se aprovecharon los procedimientos que calculan las curvas de nivel. El primer paso que se realizó fue el identificar aquellas curvas de nivel que estuvieran cerradas, esto se obtiene cuando se utiliza la función *buscar_vertice_nuevo* que obtiene el conjunto de vértices que forma a una curva de nivel y dado que es una lista cerrada, se buscan e identifican aquellas listas que tengan los mismos valores al principio y al final. Esto indica que es una curva de nivel cerrada . Las operaciones que se diseñaron para identificar si una curva de nivel es cerrada o no se explican a continuación.

Cada vez que se realiza el dibujo en pantalla de una curva de nivel, ésta *debe calcularse y almacenarse previamente* antes de su despliege en un arreglo tridimensional $(x, y, altura)$, para que se libere en cuanto se envíen los datos a la pantalla. Para decidir si una curva de nivel es cerrada, el primer paso

consiste en guardar en dos variables temporales las coordenadas x e y del primer punto. Después, se van incorporando los vértices que serán parte del polígono y por medio de un condicionamiento se verifica el momento en que los dos primeros vértices se repiten, indicando con esa acción que el polígono actual representa una curva cerrada. Aunado a lo anterior, es necesario llevar una lista de las curvas de nivel cerradas, por lo que se utiliza un arreglo cuyo contenido son los índices que se asignan a cada una de estas curvas, en el momento que son detectadas.

Una vez que se ha detectado una curva cerrada, el siguiente paso es identificar cuáles de las curvas de nivel cerradas representan a la base como al cráter del volcán. Esto se hace por dos vías, una mediante el cálculo de los diámetros mayor y menor de cada curva para analizar relaciones de contención, y otra, simplemente comparando las alturas extremas de las curvas. Para calcular el diámetro de la base se localizan los puntos que están más al este, oeste, norte y sur, ya que al obtener estos cuatro puntos, se obtienen los cuatro puntos más lejanos que forman la curva de nivel, estos puntos se guardan en un arreglo. Hay que señalar que este arreglo lleva como índice el número que se asignó a la curva cuando se detectó que era cerrada. Estos dos procesos se llevan a cabo hasta que se han calculado todas las curvas de nivel. Con la información obtenida, se pueden identificar qué curvas de nivel cerradas están contenidas dentro de otras.

Este último procedimiento se realiza en la función *identificar_curvas_inter()* en donde se almacena el cálculo de los diámetros mayor y menor. Para ilustrar ésto, a continuación se da el listado que realiza la búsqueda y es parte de la función.

```
/*****************************************************************/
/*   C\'alculo del Diametro de la base de la montran~a     */
/*****************************************************************/
if(sqrt(pow(interior[busca[0]*9+6] - interior[busca[0]*9+2],2) +
    pow(interior[busca[0]*9+7] - interior[busca[0]*9+3],2) ) >
```

```
  sqrt(pow(interior[busca[0]*9+0] - interior[busca[0]*9+4],2) +
    pow(interior[busca[0]*9+1] - interior[busca[0]*9+5],2) ))
          {
diamMay=sqrt(pow(interior[busca[0]*9+6] - interior[busca[0]*9+2],2) +
       pow(interior[busca[0]*9+7] - interior[busca[0]*9+3],2) );
diamMen=sqrt(pow(interior[busca[0]*9+0] - interior[busca[0]*9+4],2) +
       pow(interior[busca[0]*9+1] - interior[busca[0]*9+5],2) ),
if(cuent == 1)
  fprintf(fp,"Diametro Base  May %f Men %f Altura %f Posic %d",diamMay,diamMen,
interior[busca[0]*9+8],busca[0]+1);
else
  fprintf(fp,"Diametro Base  May %f Men %f Altura %f Posic %d ",diamMay,diamMen,
       interior[busca[0]*9+8],busca[0]+1);
          }
          else
          {
diamMen=sqrt(pow(interior[busca[0]*9+6] - interior[busca[0]*9+2],2) +
       pow(interior[busca[0]*9+7] - interior[busca[0]*9+3],2) );
diamMay=sqrt(pow(interior[busca[0]*9+0] - interior[busca[0]*9+4],2) +
       pow(interior[busca[0]*9+1] - interior[busca[0]*9+5],2) );
if(cuent == 1)
  fprintf(fp,"Diametro Base  May %f Men %f Altura %f Posic %d ",diamMay,diamMen,
 interior[busca[0]*9+8],busca[0]+1);
else
  fprintf(fp,"Diametro Base  May %f Men %f Altura %f Posic %d ",diamMay,diamMen,
        interior[busca[0]*9+8],busca[0]+1);
          }
/*************************************************************/
/*   Calculo del Diametro de la cima de la montran~a     */
/*************************************************************/
```

```
if(sqrt(pow(interior[busca[1]*9+6] - interior[busca[1]*9+2],2) +
    pow(interior[busca[1]*9+7] - interior[busca[1]*9+3],2) ) >
  sqrt(pow(interior[busca[1]*9+0] - interior[busca[1]*9+4],2) +
    pow(interior[busca[1]*9+1] - interior[busca[1]*9+5],2) ))
            {
diamMay=sqrt(pow(interior[busca[1]*9+6] - interior[busca[1]*9+2],2) +
      pow(interior[busca[1]*9+7] - interior[busca[1]*9+3],2) );
diamMen=sqrt(pow(interior[busca[1]*9+0] - interior[busca[1]*9+4],2) +
      pow(interior[busca[1]*9+1] - interior[busca[1]*9+5],2) );
fprintf(fp,"Diametro Cima  May %f Men %f Altura %f",diamMay,diamMen,
  interior[busca[1]*9+8]);
            }
            else
            {
diamMen=sqrt(pow(interior[busca[1]*9+6] - interior[busca[1]*9+2],2) +
      pow(interior[busca[1]*9+7] - interior[busca[1]*9+3],2) );
diamMay=sqrt(pow(interior[busca[1]*9+0] - interior[busca[1]*9+4],2) +
      pow(interior[busca[1]*9+1] - interior[busca[1]*9+5],2) );
fprintf(fp,"Diametro Cima  May %f Men %f Altura %f",diamMay,diamMen,
  interior[busca[1]*9+8]);
            }
        }
    for(m = 0; m < cuent; m++)
    {
        for(l = 0; l < cta_global; l++)
        {
            if(global[l] == exterior[m]) global[l] = exterior[m]*-1;
        }
    }
```

Al examinar el código, se puede observar que se utilizan dos arreglos

interior y busca, el primero contiene a todas las curvas de nivel que se detectaron como cerradas y el segundo es el que indica las bases, y los cráteres. Las variables `diamMay` `diamMen` contienen los diámetros que se calculan en dos direcciones una este-oeste y la otra norte-sur y se hace una comparación con el fin de obtener el mayor asignándolo a la variable `diamMay` y el menor a `diamMay`. De esta forma finaliza el cálculo de los diámetros.

Despues sólo resta calcular el volúmen, para lo que es necesario localizar a todas las curvas que definen la base de un volcán, para lo que se utilizan las pendientes cercanas, y si se encuentra una variación de entre 20 y 30 unidades, se puede decir con certeza que se trata de la base de un volcán. Una vez que se localiza cada una de estas curvas, lo que se hace es fijar el área que delimita y extraer la altimetría que se encuentra en cada área.

Como la altimetría que compone a cada zona se encuentra en la malla regular que se utiliza para representar la topografía, se pueden construir prismas que están contenidos dentro de cada volcán, haciendo de su base y su tapa un área cuadrada de aproximadamente 90×90 metros. La altura del prisma depende de las alturas definidas por las coordenadas (x, y) contenidas dentro de los vertices de las curvas de nivel que definen a la base del volcán.

A continuación se describe cómo se recuperan las distintas alturas que se encuentran en la malla y que están delimitadas por la curva que define la base para multiplicar esta altura por el área que se definió anteriormente. Así se obtiene el volúmen del prisma.

El procedimiento anterior se puede observar gráficamente en la figura 3.17.

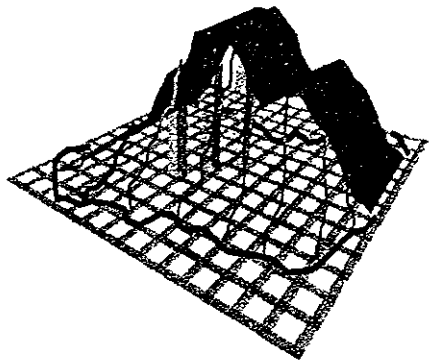


Figura 3.17: Obtención de la altimetría que está delimitada por alguna curva de nivel cerrada que define la base de un volcán.

Por último sólo resta realizar la suma de todos aquellos prismas que estén contenidos dentro del volcán (ver figura 3.18) para obtener el volúmen total de cada uno de los volcanes. Esto último se puede expresar con la siguiente fórmula $\sum_{FO \geq i \leq FE, FS \geq j \leq FN} VolPrisma_{ij}$, en donde las variables FO, FE son la frontera oeste-este de la base del volcán y las variables FN, FS definen la frontera sur-norte respectivamente

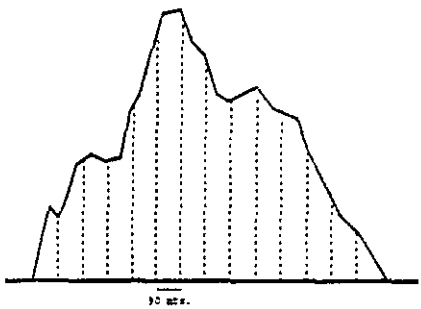


Figura 3.18: Obtención del volumen total.

Para mostrar algunos de los resultados que se obtuvieron al estudiar el campo volcánico que se encuentra en los estados de Michoacán y Guanajuato, se muestra la tablas 3.1, 3.2, 3.3, 3.4. En donde además se comparan con los resultados de un estudio previo de Hasenaka y Carmichael [1985][19][Hasenaka-Carmichael] sobre el mismo campo. Los parámetros geomorfológicos que se presentan y comparan son los siguientes. El diametro de la base, D_b y el diámetro del cráter, D_c, tablas 3.1, 3.2, la altura del cráter sobre el terreno h, el volumen del volcán v, la comparación de la razón h/D, y la pendiente promedio de las laderas, θ tablas 3.3, 3.4, además de incluir tanto el nombre y la ubicación geográfica.

Nombre	latitud	longitud	D_b km	D_c km	latitud	longitud	D_b km	D_c km
C. El aire	19°31.23′	102°3 85′	1.392	0.567	19°31 22′	102°3.87′	1.40	0.60
C El Varal	19°31.49′	102°3 01′	1 090	0.347	19°31 51′	102°3 07′	1 08	0.40
El Cerrito	19°31.60′	102°2.15′	0 550	0.259	19°31 62′	102°2 18′	0.53	0.30
C. Isingo	19°29 42′	102°17.72′	0 750	0.160	19°29 40′	102°17.79′	0.28	0 28
	19°29 95′	102°17.12′	0 480	0.089	19°29 95′	102°17 13′	0.58	0.08
	19°29 80′	102°6 75′	0 838	0 235	19°29 81′	102°6 72′	0 85	0 25
	19°29 54′	102°16.23′	0.590	0 270	19°29.57′	102°16 26′	0 60	0.28
	19°29.77′	102°15 85′	0 980	0.263	19°29 77′	102°15 87′	0 90	0.28
Paricutin	19°29.55′	102°15 05′	0 910	0.190	19°29 55′	102°15.07′	0 95	0 25
	19°29 02′	102°16.76′	0 700	0.230	19°29 03′	102°16.78′	0.65	0.25
	19°28.60′	102°16.95′	0 501	0 278	19°28.59′	102°16 98′	0 55	0.30
C.San Pedro	19°27.20′	102°16 57′	1 110	0 211	19°27.19′	102°16 55′	1.00	0.25
C San Pedro	19°27 35′	102°16 45′	1 090	0.100	19°27.35′	102°16 47′	1 00	0.10
C El Cebo	19°27 45′	102°15 92′	0.700	0.090	19°27 46′	102°15 98′	0 68	0.05
La Escondida	19°27.60′	102°15 40′	0.090	0 0	19°27 64′	102°15.40′	0.00	0.00
C. El Tizne	19°27.69′	102°15 01′	0 561	0.210	19°27 65′	102°15.09′	0.50	0.20
C Cirahapan	19°28.40′	102°14.02′	0 793	0 370	19°28 43′	102°14.00′	0.88	0 40
	19°28.35′	102°13.75′	0 660	0 190	19°28 38′	102°13.71′	0.65	0 20
C. Pancingo	19°29.76′	102°11.82′	0 603	0 300	19°29.76′	102°11 81′	0.63	0.25
	19°26 50′	102°14.99′	1.096	0 318	19°26.53′	102°14 96′	1.03	0.35
C Prieto	19°26 23′	102°13 14′	0.480	0 090	19°26.23′	102°13 17′	0 50	0.08
	19°27 05′	102°12 68′	0.601	0 164	19°27.10′	102°12 69′	0 65	0.20
C LaPerita	19°27.19′	102°12 36′	0 909	0 310	19°27.21′	102°12.32′	0.90	0.33
LL Juritzicuaro	19°27 50′	102°12 33′	0.479	0.094	19°27.57′	102°12 30′	0.45	0.10
C Estiradero	19°26 58′	102°11 77′	0.900	0.257	19°26 58′	102°11.78′	0 89	0.28
LL. Tacadero	19°27.99′	102°11 89′	0.450	0.281	19°27 95′	102°11.85′	0.50	0.30
C. Pario	19°28 17′	102°11 02′	0 879	0.100	19°28.14′	102°11.06′	0.90	0 10
C. Tumbiscatillo	19°27 80′	102°10.37′	0 926	0 189	19°27.79′	102°10 34′	0 93	0 23
	19°28.70′	102°10.72′	0.410	0 090	19°28.70′	102°10 69′	0.45	0.08
C. Las Varas	19°27.70′	102°07 70′	0.728	0 130	19°27.75′	102°07 74′	0.78	0.15
(C. El Jabali)	19°26 60′	102°07 28′	0.703	0 520	19°26.93′	102°06 76′	0 93	0.38
	19°26.24′	102°06 90′	0.659	0 209	19°26.62′	102°07 23′	0 70	0.23
C Sapien	19°26 90′	102°06 17′	0 734	0 230	19°26 26′	102°06 91′	0 65	0 28
C. El Jabali	19°26.91′	102°06 74′	0.938	0.482	19°26 92′	102°06.10′	0.73	0.50
(Costo)	19°26.59′	102°04.10′	1 360	1.095	19°28 02′	102°01.93′	1.25	1.15
C. Copitiro	19°29.11′	102°03.01′	0 898	0.224	19°26.61′	102°01.13′	0.95	0.25

Tabla 3.1. Comparación de los datos geomorfológicos del campo volcánico Michoacán-Guanajuato, latitud, longitud, D_b, D_c. Los datos a la izquierda de las tres lineas verticales, son los que se obtuvierón en este trabajo.

Nombre	latitud	longitud	D_b km	D_c km	latitud	longitud	D_b km	D_c km
(C. Cheranguaran	19°28 00′	102°01.95′	1.200	0.382	19°28 02′	102°01 93′	1 25	0 35
(C. El Cajete	19°28.20′	102°02.57′	0 890	0 300	19°28 26′	102°02 52′	0 85	0 28
(C. El Puerto	19°28 50′	102°02.11′	1.103	0.299	19°28 45′	102°02.11′	0 93	0 33
(El Las Paredes)	19°28 95′	102°09.93′	1 480	0 170	19°28 95′	102°09.90′	1.28	0 20
(C. San Pedro)	19°26 33′	102°16.89′	0 675	0 169	19°26 35′	102°16 90′	0.65	0 18
El Tepetate	19°21.94′	102°15.10′	0.500	0 137	19°21 90′	102°15 99′	0.45	0.15
C. La Chimenea	19°21 63′	102°15 55′	0 800	0 346	19°21 10′	102°15 56′	1.20	0 33
El Tejunanil	19°24 40′	102°15.03′	0.00	0 0	19°24 43′	102°15 03′	0.00	0 00
El Ucasuaro	19°25 44′	102°14.36′	0 520	0 110	19°25 18′	102°14 33′	0.60	0 13
C. Curucuti	19°25.89′	102°10.55′	0 790	0 695	19°25 93′	102°10 59′	0.75	0 65
C. El Colorado	19°25 37′	102°10.17′	0 850	0 273	19°25 33′	102°10 11′	0.83	0.25
C. El Rosario	19°26.25′	102°09.34′	0 747	0.097	19°26 20′	102°09.31′	0.73	0 08
C. La Trinidad	19°28 57′	102°08 18′	0.563	0 090	19°28 56′	102°08 15′	0.58	0 10
C. La Alberca	19°29 02′	102°00.68′	1 460	0.314	19°29 05′	102°00.66′	1.28	0 30
C. Cinco	19°21 69′	102°06 37′	1 289	0.503	19°21 66′	102°06 36′	1.38	0 48
C. Jacalun	19°23 30′	102°04.69′	1.390	0 478	19°23 33′	102°04 68′	1.40	0 50
	19°21 58′	102° 33′	1.243	0 407	19°21 61′	102°00 34′	1.15	0 35
C. La Alberca	19°21.70′	102° 33′	1.130	0 325	19°21 73′	102°17 90′	1 03	0 35
El Cerrito	19°21.50′	102°17.33′	0 876	0 180	19°21 51′	102°17 31′	0 95	0.20
(La Soledad)	19°21.05′	102°17 03′	0 959	0.190	19°21 05′	102°17.06′	0.93	0.23

Tabla 3.2: Comparación de los datos geomorfológicos del campo volcánico Michoacán-Guanajuato, latitud, longitud, D_b, D_c. Los datos a la izquierda de las tres lineas verticales, son los que se obtuvierón en este trabajo.

Nombre	A km	V_v m^3	A/D_b	Θ	A km	V_v	A/D_b	Θ
C. El aire	2 21328	0 16090	0 159	27	.205	170	0.15	27
C. El Varal	1.711	0 07410	0 160	23	.160	074	0.15	25
El Cerrito	0 59950	0 07950	0.100	20	.550	008	0.10	26
C Isingo	1.065	0 02106	0.112	25	.105	.023	0 14	24
	0 78720	0 08055	0.161	22	090	009	0.16	20
	0 92180	0 22107	0.110	20	095	.025	0.11	18
	0 65018	0 09800	0 1102	25	.060	010	0 10	21
	1 69510	0 11007	0 173	25	155	046	0 17	27
Paricutin	2 12949	0 660706	0 231	32	220	069	0.23	32
	0.567	0 00760	0.081	13	.050	008	0.08	14
	0 56613	0 00802	0.113	23	060	.009	0.11	26
C San Pedro	2.220	0 06405	0 20	28	195	067	0.19	27
C San Pedro	1 907	0 03990	0.175	21	170	019	0.17	21
C El Cebo	1 631	0 01850	0 233	26	155	020	0.23	26
La Escondida	0 00	0.0	0.00	5	.000	000	0.00	-
C El Tizne	0 67881	0 00179	0 121	21	060	006	0.12	22
C Cirahapan	1.14985	0 03560	0 145	29	120	.040	0.14	27
	0 67980	0 00082	0 103	16	.065	.010	0 10	16
C Pancingo	2 50981	0 00860	0 101	17	.060	.010	0 10	18
	0.68160	0 06710	0 157	24	.150	061	0 15	21
C Prieto	1 08116	0 00870	0 226	28	.110	.009	0 22	28
	1 19079	0 15070	0 142	29	090	014	0 14	22
C. LaPerita	0 78556	0 50070	0 179	29	155	049	0 17	29
LL Juritzicuaro	0 333	0 00369	0 131	17	055	001	0 12	17
C. Estiradero	0 95850	0 36010	0 161	20	130	038	0.15	23
LL. Tacadero	1 51515	0 01950	0 037	7	015	002	0.03	9
C. Pario	2 07121	0 12605	0.213	24	180	043	0.20	24
C. Tumbiscatillo	0 779	0 50101	0.185	25	170	050	0 18	26
	1 28856	0 00583	0.224	28	100	.006	0.22	28
C. Las Varas	0 86469	0.02409	0 190	27	110	027	0.18	24
(C El Jabalí)	1 138	0.01030	0.177	22	.160	057	0.17	30
	0.51380	0.00970	0 123	21	.075	.012	0 12	20
C Sapien	1.53832	0.02701	0 200	31	.130	.028	0 18	30
C El Jabalí	2 23010	0 05980	0 070	32	045	.013	0 06	24
(Costo)	0 00	0 0	0 161	33	030	000	0 17	33
C. Copitiro	1 49266	0 04906	0 167	25	160	050	0 17	25

Tabla 3.3. Comparación de los datos geomorfológicos del campo volcánico Michoacán-Guanajuato. A, V_v, A/D_b, Θ. Los datos a la izquierda de las tres lineas verticales, son los que se obtuvierón en este trabajo.

Nombre	A km	V_v m^3	A/D_b	Θ	A km	V_v	A/D_b	Θ
C. Cherangnaran	1 968	0 10993	0 164	22	200	.111	0 16	24
C. El Cajete	1 157	0 02504	0 130	8	100	027	0 12	-
C. El Puerto	1 89716	0 04607	0 172	25	145	049	0 16	26
(El Las Paredes)	3 1089	0 04910	0 119	19	550	005	0 11	21
(C. San Pedro)	1 76120	0 15070	0 162	23	100	015	0 15	23
El Tepetate	1 69350	0 00398	0 134	22	650	.005	0 14	23
C. La Chimenea	1 496	1 07052	0 187	24	215	.109	0.18	26
El Tejamanil	0 00	0.0	0 00	3	000	.000	0.00	-
El Escurro	1 08680	0 01261	0 209	27	120	.014	0.20	27
C. Cuiseo	0 790	0.09608	0 100	15	.016	.016	0 09	18
C. El Colorado	1 53850	0.03901	0.181	24	145	.036	0.17	27
C. El Rosario	1 17279	0.14089	0 157	9	.110	170	0.15	-
C. La Irandad	0 92628	0 00669	0.124	13	.070	.007	0 12	16
C. La Alberca	2 67180	0 01195	0.183	27	.220	.122	0.17	24
C. Chino	2 02373	0.13440	0 157	27	210	151	0.15	32
C. Jaralito	1 70070	0.12094	0.123	31	170	.130	0.12	33
	1 61590	0 06040	0.130	7	135	.065	0.12	-
C. La Alberca	2 91530	0 05905	0.181	27	.170	.069	0.17	27
El Cerrito	1 77828	0 01902	0.203	24	.180	.053	0.19	26
(La Soledad)	11 16276	0 03068	0.164	26	.135	.040	1.15	21

Tabla 3.4: Comparación de los datos geomorfológicos del campo volcánico Michoacán-Guanajuato, A, V_v , A/D_b, Θ. Los datos a la izquierda de las tres lineas verticales, son los que se obtuvierón en este trabajo.

La importancia de este campo volcánico reside en que es una zona del país con más de mil volcanes monogenéticos (hacen erupción una vez y se extinguen)

3.3 Comparación con otro sistema

La visualización de los datos topográficos también se realizó en otro programa llamado *IDL* (*Interactive Data Language*, por las siglas en ingés), este programa también utiliza un tipo de programación visual basado en una serie de instrucciones en Fortran 90 y se escriben en una ventana que proporciona. Otro modo de hacer esto, es escribir todas las instrucciones en un archivo, para que una vez que se utiliza el *IDL* se lean y ejecuten.

Existen varias ventajas entre las que destaca la gran capacidad de análisis y funciones que ya vienen incorporadas. Los análisis que se hicieron fueron: obtener las curvas de nivel de una región comprendida en un cuadrante geográfico, recortar subzonas dentro de un cuadrante geográfico, generar imágenes de la superficie del terreno y obtener imagenes en donde se observe el relcve tridimensional del terreno.

Para esto, se incluye el listado de las instrucciones que se utilizaron, comentando los resultados que se obtienen, así como las imágenes.

Lectura y obtención de las curvas de nivel

Las instrucciones que se utilizaron para leer la altimetría son las siguientes:

```
openr,1,'102w19n.dtt'
arreglo = intarr(1201,1201)
readf,1,arreglo
close,1
```

Las dos primeras líneas definen el archivo de donde se van a extraer los datos de la altimetría, se le asigna una etiqueta con el fin de establecer una referencia. Posteriormente se inicializa una variable `arreglo` como un arreglo bidimensional de 1201×1201 elementos, que como se recordará son las dimensiones que tiene un cuadrante geográfico. Una vez que se obtiene el nombre del archivo y en dónde se van a almacenar los datos, se procede a leer la información y cerrar el acceso al archivo.

Por último, se utiliza la sentencia `contour, arreglo` que sirve para calcular las curvas de nivel conforme a un intervalo de altura de 250 metros, la imagen que se obtiene se encuentra en la figura 3.19.

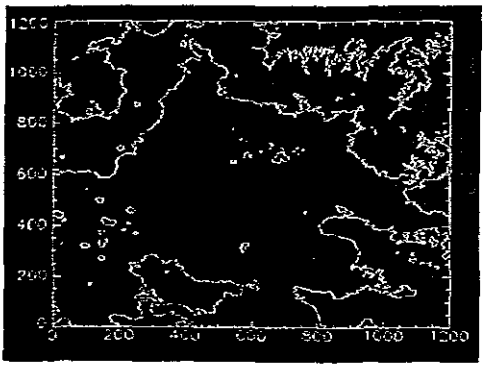


Figura 3.19. Imagen que muestra una curva de nivel

Obtención de un subconjunto de datos y el cálculo de distintas curvas de nivel

Las instrucciones del paso anterior se almacenaron en un archivo, cuyo nombre es `lectura.pro`. Esto auxilia en el momento de leer los datos en otros programas, con sólo invocar el nombre del archivo, se ejecutan las instrucciones que están en él y envía los resultados a los subsecuentes comandos.

Las siguientes líneas son las instrucciones que se usaron para decimar la información y generar distintas curvas de nivel.

```
@lectura
x = 326.850 + .090 * findgen(1201)
y = 326.850 + .090 * findgen(1201)
contour, arreglo, x, y, levels = 2000 + findgen(9) * 250.0,
```

Con las instrucciones previas se leen los datos, después se indica el tamaño del conjunto que definirá a la subzona, que es de 326.850 × 326.850 metros.

estos valores marcan la posición (x, y) donde finaliza el corte, el inicio es el $(0, 0)$; el valor 0.090 indica la distancia en metros entre un dato y sus vecinos. por último se calculan las curvas de nivel de esta subregión cada 250 metros. La figura 3.20 muestra los resultados obtenidos.

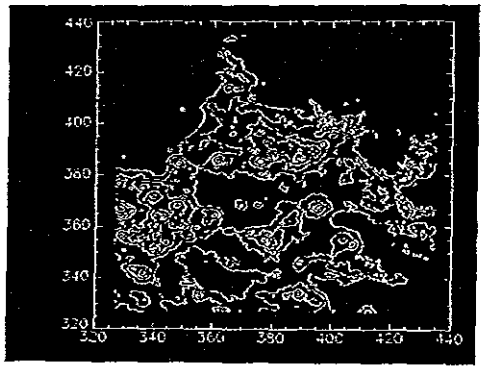


Figura 3.20: Imagen que muestra distintas curvas de nivel del terreno.

La figura 3.21 muestra la superposición de dos distintos niveles de curvas de nivel, así como la incorporación de texto en la imagen las instrucciones que se emplearon son las siguientes.

```
xstyle = 1, ystyle = 1, ymargin = 5, max_value = 3000,
c_linestyle = [1,0],
c_thick = [1,1,1,1,1,3],
title='Region de la cuenca de Mexico',
subtitle='contorno de 250 metros',
xtitle = 'Coordenadas UTM (KM)'$
```

Obtención de imágenes en tonos de gris y la superposición de distintas curvas de nivel

```
@lectura
```

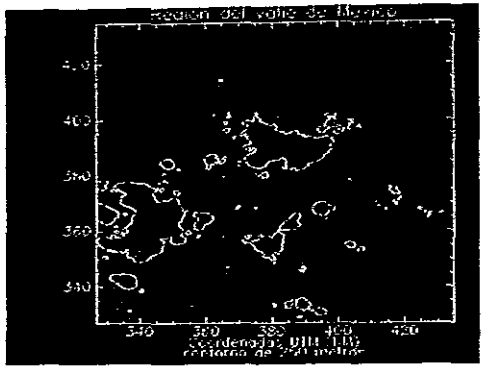


Figura 3.21: Imagen que muestra la superposición de distintos niveles de curvas de nivel.

```
x = 326.850 + .030 * findgen(1201)
y = 326.850 + .030 * findgen(1201)
image = bytscl(arreglo, min=2000, max=3000)
contour, arreglo, x, y, levels = 2250 + findgen(8) * 250.0,
max_value=50000, xstyle=1, ystyle=1,
title='Region de la cuenca de Mexico',
subtitle='contorno de 250 metros',
xtitle = 'Coordenadas UTM (KM)', /nodata
```

Las siguientes instrucciones son las que utilizan para obtener la imagen en la subregión.

```
px='x.window * !d.x_vsize
py='y.window * !d.y_vsize
sx=px[1]-px[0]+1
sy=py[1]-py[0]+1
tvscl, congrid(image,sx,sy),px[0],py[0]
```

Para superponer las curvas de nivel a la imagen de tonos de gris, se incluye lo siguiente

```
contour, arreglo, x, y, levels = 2250 + findgen(8) * 250.0,
max_value=50000, xstyle=1, ystyle=1,
title='Region de la cuenca de Mexico',
subtitle='contorno de 250 metros',
xtitle = 'Coordenadas UTM (KM)', /noerase
```

La imagen con únicamente los tonos de gris se muestra en la figura 3.22 y la que contiene la superposición de esta última con sus respectivas curvas de nivel es la figura 3.23.

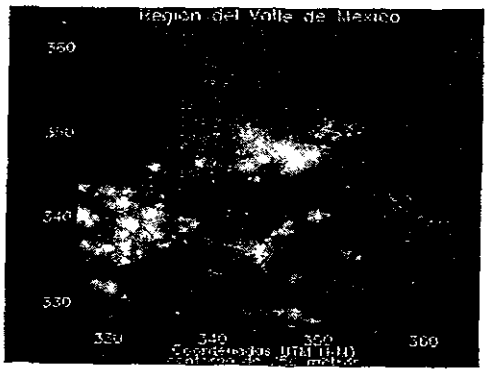


Figura 3.22: Imagen en tonos de gris

Las dos imágenes anteriores son subregiones que se obtuvieron de un cuadrante geográfico, por lo que falta realizar los mismos cálculos pero esta vez para un cuadrante geográfico completo, por lo que se escribieron las siguientes líneas para hacerlo.

```
@lectura
x = 326.850 + .030 * findgen(1201)
y = 326.850 + .030 * findgen(1201)
image = bytscl(arreglo, min=2000, max=3000)
px='x.window*!d.x_vsize
```

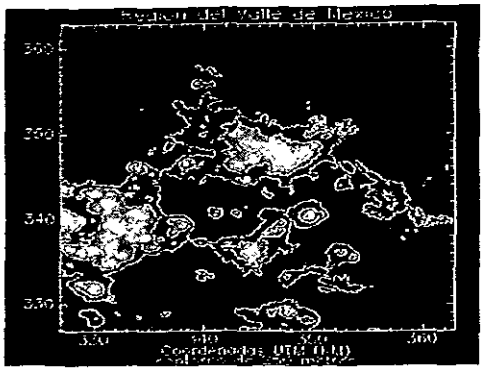


Figura 3.23: Imagen con la superposición de los tonos de gris y sus curvas de nivel

```
py=!y.window*!d.y_vsize
sz=size(image)
tvscl,image,px[0],py[0]
contour, arreglo, xstyle=1, ystyle=1,
position=[px[0],py[0],px[0]+sz[1]-1,py[0]+sz[2]-1],
levels=2250+findgen(8) * 250., max_value=5000,
title='Region de la cuenca de Mexico, Ajusco y Nevado de Toluca',
subtitle='contorno de 250 metros',
xtitle = 'Coordenadas UTM (KM)', /noerase, /device
```

La figura 3.25 muestra las imágenes que se obtienen.

Realce y superposición del las curvas de nivel

Por último se trató de generar el realce de la altimetría y obtener una imagen tridimensional además de incluir las curvas de nivel del terreno, pero desgraciadamente la imagen que se obtiene es muy pobre, esto se puede observar en la figura 3.26.

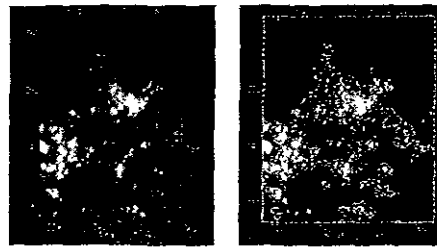


Figura 3.24: Imagen con tonos de gris y superposición de curvas de nivel. de una subregión del cuadrante (326.850,326.850)

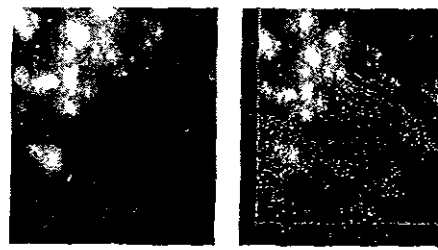


Figura 3.25: Imagen con tonos de gris y su superposición de sus curvas de nivel del cuadrante completo (1201.1201)

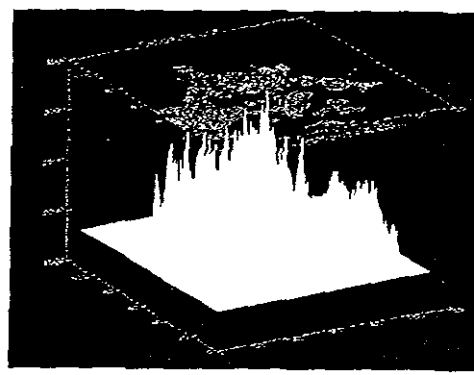


Figura 3.26: Realce de la altimetría y sus curvas de nivel

Conclusiones

En este trabajo se presentan los resultados del desarrollo de un sistema de visualización de datos topográficos del territorio nacional, en el que se involucran aspectos de la computación que están íntimamente relacionados: visualización, supercómputo y programación visual.

El sistema que se presenta cumple con las características necesarias para el fin que fue originalmente pensado ya que se puede representar la topografía digital del territorio nacional en forma tridimensional, desarrollar estudios de geomorfología de volcanes asistidos por computadora al calcular en forma automática los parámetros geomorfológicos de un campo volcánico como son el diametro de la base, D_b, el diámetro del cráter, D_c, la altura del cráter sobre el terreno, A, la pendiente promedio de las laderas, θ, el volumen del volcán v, y la comparación de la razón A/D_b

La calidad de los datos obtenidos es muy alta, debido a que los parámetros calculados se obtienen a partir de las curvas de nivel que se calculan y los datos brutos del modelo digital. Tomando en cuenta este último comentario, se puede afirmar, que se abren nuevas capacidades de análisis geoespaciales utilizando la representación gráfica como uno de los principales pilares que dichos estudios requieren.

La programación visual actualmente está relacionada con lenguajes de programación, interfases gráficas, sistemas expertos, sistemas operativos distribuidos y más recientemente, la programación orientada a objetos. Lo que provee un nueva manera de mejorar la creación, mantenimiento y empleo de los programas. De esta manera se cambia un poco la forma de progra-

mar, aumentando la velocidad en las aplicaciones y su capacidad de utilizar distintas plataformas, así como la depuración, codificación y diseño.

Este trabajo fué pensado originalmente con fines geológicos y geofísicos como una base de información geográfica. Sin embargo las aplicaciones potenciales de este tipo de sistemas son muy variadas. Entre ellas se pueden mencionar estudios de contaminación, estudios geológicos, de planeción urbana, de transporte, botánicos y ecológicos, sistemas e información geográfica y fenómenos atmosféricos.

Bibliografía

[1] Ephraim P. Glinert.
Visual Programing Environments Applications and Issues.
IEEE Computer Press Tutorial. 1990
Los Alamitos, California.
Páginas 720.

[2] Ephraim P. Glinert
Visual Programing Environments Paradigms and Systems.
IEEE Computer Press Tutorial. 1990
Los Alamitos, California.
Páginas 730.

[3] Shi-kuo Chang,
Tadao Ichikawa,
Panos A. Ligomenides.
Visual Languages.
Plenum Press. 1986
New York.
Páginas 460

[4] **AVS Developerś Guide.**
Advanced Visual Systems Inc Mayo 1992
Waltham, MA 02154.
Páginas 500.

[5] **AVS Userś Guide.**
Advanced Visual Systems Inc. Mayo 1992
Waltham. MA 02154.
Páginas 600.

[6] **Visualization Case Studies Plus.**
IEEE Computer Graphics and Applicatons.
Julio 1995.
Volúmen 15, Número 4.

[7] **Modular Visualization Environments Past, Present and Future.**
IEEE Computer Graphics and Applications.
Mayo 1995.
Volúmen 29, Número 2.

[8] James D. Foley.
Andries van Dam.
Steven K. Feiner.
John F. Hughes.
Computer Graphics Principles and Practice.
Addison-Wesley Systems Programming Series.
Washigton D.C. 1990.
Páginas 1180

[9] James D. Foley.
Andries van Dam.
Fundamentals of Interactive Computer Graphics.
Addison-Wesley The Systems Programming Series
Washigton D.C. 1983.
Páginas 770

[10] Ernesto Bribiesca.
La Topografía del Valle de México Representada en forma Digital.
Reportes de Desarrollo.
Volumen 3, No 7, Marzo 1993.
IIMAS, UNAM.

[11] David McIntyre.
Visual Languages.
URL http://union.ncsa.uiuc.edu/HyperNews/get/computing/visual.html.
Diciembre 1994.

[12] **IEEE Computer Graphics and Applications.**
URL http://www.computer.org/pubs/cga/cga.htm
Febrero 1996.

[13] Benjamin C. Summers
Margaret M. Burnett.
Visual Programming Language Bibliography
URL http://www.cs.orst.edu/ burnett/vpl.html
Noviembre 29. 1995.

[14] IAC International AVS Center.
URL http://iac.ncsc org/
1995.

[15] IAC International AVS Center The IAC Public-Domain FTP Site

URL http://iac.ncsc.org/IAC/ftpsite.html.
1995.

[16] **Advanced Visual Systems.**
URL http://www.avs.com/
1993.

[17] **What is AVS.**
URL ftp://testavs.ncsc.org/avs/Info/WHAT_IS_AVS
1995.

[18] **IAC International AVS Center AVS and Visualization World Wide Web Sites.**
URL http://iac.ncsc.org/IAC/friends.html
1995.

[19] T. Hasenaka e I.S.E. Carmichael.
A compilation of location, size, and geomorphological parametres of volcanoes of the Micho acan-Guanajuato Volcanic field, central México.
Geofísica Internacional,
24-4, 577-607,85.