## Introducción

El problema consiste en implementar en CUDA la obtención de los vectores normales correspondientes a un conjunto amplio de puntos 3D en función de los vecinos más cercanos. El cálculo de normales para una superficie tridimensional es un aspecto muy importante en el campo de los gráficos por computador, visión artificial y robótica, entre otros.

**Cálculo de las normales de una superficie**

El cálculo de las normales de una superficie puede ser abordado desde diferentes puntos de vista. En el ejercicio que se presenta a continuación, se propone obtener las normales correspondientes a una superficie que está representada por un conjunto de puntos 3D. Para cada uno de los puntos de la superficie, se calculará el vector normal correspondiente utilizando los puntos de la vecindad del mismo.

Superficie representada por puntos

Vector normal a un punto

Se considera que los puntos de la superficie están organizados en una estructura de malla que permite determinar la vecindad de un punto dado. Para el cálculo de la normal en ese punto, se establece que será el vector normal medio de las normales de las rectas que pueden establecerse entre el punto y cada uno de los puntos de su vecindad. El algoritmo podría ser el siguiente:

para toda P de los puntos de la superficie

vectorMedio(P)=(0,0,0);

para todo punto Q de la vecindad de P

recta=CalcularRecta(P,Q);

vectorMedio=vectorMedio+CalculaNormal(recta);

fin para

vectorMedio(P)=vectorMedio(P)/numPuntosVecindad;

fin para

## Funciones

El algoritmo que resuelve el cálculo de las normales se aporta, en su versión secuencial, en la función *CalculaNormalesCPU* dentro del fichero *calculaNormales.cu*. Se plantea implementar la función *CalculaNormalesGPU.* Esta función debe ser funcionalmente equivalente a la función CPU del mismo nombre, realizando operaciones equivalentes para determinar el vector normal que corresponde a un conjunto de puntos de entrada correspondientes a una superficie. En esta función deberá usarse la programación basada en CUDA® para aprovechar la capacidad de cómputo paralelo de las GPUs de NVIDIA®.

De forma análoga a la función CPU, el resultado de la función deberá devolver en los vectores *NormalUGPU, NormalVPU, NormalWGPU* los componentes de los vectores normales normalizados (entre 0 y 1) de cada uno de los puntos de la superficie.

Estos vectores serán posteriormente comparados en la función *runTest* (ya implementada) con el generado por la solución CPU para comprobar su corrección. Los tiempos que serán tenidos en cuenta para la competición por equipos serán exclusivamente los generados para la solución GPU y que muestra la función *runTest* por pantalla.

### Función runTest

La función principal del programa tiene como objetivo, leer un fichero con la información de la superficie que se pasará como argumento y realizar el cálculo de las normales propuesta en el apartado anterior.

Una vez leída la información del SOM y de los patrones, mediante las funciones *LeerSOM* y LeerPatrones (ya implementadas) se lanzan dos algoritmos, uno basado en CPU-Secuencial (sin llamadas a CUDA) y que es aportado en la práctica y otro basado en GPU que debe ser implementado por el grupo. El tiempo de ejecución es obtenido para cada uno de ellos y, tras la conclusión de ambos, se comprueba si la solución generada es idéntica. En tal caso, el tiempo de GPU será tenido en cuenta a efectos de la competición.

### Funciones auxiliares

*LeerSOM*. Implementada en calculaNormales.cu

Esta función lee un fichero (.som) que es aportado en la práctica y construye un mapa SOM. El mapa se organiza en la estructura TSOM SOM y contiene:

* Ancho 🡪 Ancho del mapa
* Alto 🡪 Alto del mapa
* Dimensión 🡪 Dimensión de cada neurona del mapa
* Neurona🡪 Array bidimensional de neuronas float de tamaño Ancho\*Alto\*Dimension. Ejemplo, para acceder a la neurona de coordenadas (v,u), se accederá mediante el código SOM.Neurona[v][u]

*CrearMapa.* Implementada en clasificacionSOM.h. Gestiona la creación del mapa.

*BorrarMapa* Implementada en clasificacionSOM.h. Gestiona la liberación de memoria del mapa.

### Funciones de simulación

***CalculaNormalesCPU*.** Implementada en *calculaNormales.cu*

Para clasificar el conjunto de patrones de entrada, implementa el algoritmo comentado anteriormente. Para mayor información, la función proporcionada comenta paso a paso cada línea de código.

El resultado final de la función es el cálculo de las etiquetas como vector global (ya creado en la función principal) que tienen el tamaño del número de patrones a clasificar. Este vector *EtiquetaCPU* contiene para cada posición, la etiqueta correspondiente al patrón de entrada.

***CalculaNormalesGPU*.** NO implementada en *calculaNormales.cu*

Esta es la función que tiene que ser implementada por el grupo. El objetivo es que sea funcionalmente equivalente a la descrita anteriormente, con la diferencia de que los resultados se devolverán en el vector global (ya creados en la función principal) *EtiquetaGPU*. Como ya se ha dicho, este vector será comparado a efectos de corrección en la función *runTest*.

La función debe incluir la gestión de memoria para el paso de datos desde la CPU a la GPU y viceversa, así como la llamada a una o varias funciones kernel de CUDA que efectúen la paralelización de las operaciones siguiendo el paradigma SIMD usando la GPU del computador. Esta función o funciones kernel deberán ser definidas por el grupo y se da libertad sobre su contenido y definición.

**Entrada y salida de fichero**

Se proporcionará la lectura de dos ficheros. Uno contendrá la estructura del mapa en el que se indicará el alto y ancho del mapa así como la dimensión de los pesos de cada neurona. Por ejemplo, para un mapa 3x2 con un vector de pesos de dimensión 3:

Estructura.som

Alto: 3

Ancho: 2

Dimensión: 3

N1,1: 45.6 76.3 78.7

L1,1: 1

N1,2: 45.8 65.0 56.0

L1,2: 2

N2,1: 34.2 78.3 78.0

L2,1: 3

N2,2: 78.3 87.8 56.3

L2,3: 4

N3,1: 45.8 65.0 56.0

L3,1: 1

N3,2: 45.8 67.0 56.0

L3,3: 1

El otro fichero de entrada, contendrá los patrones de entrada. Se indicará el número de patrones y la dimensión. Por ejemplo:

patrones.pat

Numero: 5

Dimension: 3

P1: 78.0 67.7 37.7

P2: 74.0 67.5 47.7

P3: 78.0 67.1 27.7

P4: 72.0 67.3 17.7

P5: 79.0 67.5 37.7

Finalmente, la salida se escribirá en fichero y contendrá las etiquetas asociadas a los patrones de entrada. El fichero contendrá el número de etiquetas y el valor de las etiquetas correspondientes a los patrones de entrada. Por ejemplo:

salida.txt

Numero: 5

1

2

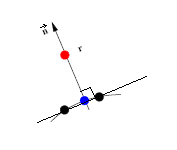
3

4

1

A fin de que la comparativa entre grupos sea lo más justa posible, no será posible salvo causa justificada expresamente, modificar ninguna de las funciones ya implementadas en la práctica.

**Simulación del movimiento de una herramienta sobre**



*Figura 1 - Normal a un punto*