Tarea 1 - Computación Científica y Ciencia de los Datos
Vectorización en Python: Proyección 3D en una malla 2D

Prof: Pablo Román

20 de Marzo 2024

1 Objetivos

Construir programas vectorizados que permitan ejecución eficiente en python. Esta técnica permite lograr velocidades comparables a un programa compilado en C e incluir aceleración vía GPU. La aceleración típica en relación a códigos realizados con loop de python son de 3 ordenes de magnitud, esto se hace al costo de un mayor uso de memoria. Para fines de este ejercicio de resolverá el problema de proyectar un conjunto de mediciones dispersas 3D en una malla 2D. Este problema se presenta al visualizar un conjunto de datos medidos en un volumen en forma de imagen bidimensional.

2 Contexto

Se dispone del valor $\{V_k\}$ de una medición que se hace en distintas ubicaciones en un volumen 3D y un peso estadístico $\{w_k\}$. Por ejemplo V puede ser una temperatura y w al inverso del error cuadrático de la medición. Sin embargo los instrumentos no localizan las mediciones en forma regularmente espaciadas sino que de una manera bien irregular. Para fines de visualización, desde el punto de vista de una cámara se requiere proyectar sobre un cierto plano. En este caso se observa un objeto situado muy lejos del observador por lo que los rayos caen perpendicularmente en el plano de observación (Fig. 1).

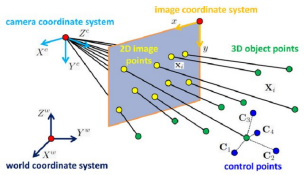

Figure 1: Relación entre puntos 3D y su proyección 2D Fuente:
`https://doi.org/10.1016/j.isprsjprs.2011.09.010`

La proyección en el plano produce una tabla de coordenadas $[x_k]$ e $[y_k]$ dispersas en el plano. Para muchos procesamientos científicos es necesario disponer de una estimación de los valores estimados de V en una malla regular (valor por cada píxel (i, j)). Por ejemplo para procesamiento de señales o simplemente para visualizar. El procedimiento más básico para encontrar un valor de $V^G_{i,j}$ en el píxel (i, j) se denomina gridding. Este consiste en que a cada punto de la grilla (i, j) se asigna el valor de un promedio ponderado de los valores vecinos ubicados a un radio menor o igual que r. En la ecuación (1) tenemos dicha suma, donde A^r_{ij} es el conjunto de ubicaciones con índices k que se encuentran a una distancia menor que r.

$$V^G_{ij} = \frac{\sum_{k \in A^r_{ij}} w_k V_k}{\sum_{k \in A^r_{ij}} w_k} \tag{1}$$

3 Datos y cálculos necesarios

Existen $N = 10000$ mediciones $\{V_k\}$ distribuidas en un volumen acotado en posiciones (vectores 3D) X_k. El plano donde se proyecta la nube de puntos 3D esta representado por su vectores unitarios como:

(`https://en.wikipedia.org/wiki/Rotation_matrix`)

$$\hat{x} = [\cos(\alpha)\cos(\beta), \sin(\alpha)\cos(\beta), -\sin(\beta)]$$
$$\hat{y} = [-\sin(\alpha), \cos(\alpha), 0]$$

Los ángulos α y β son la parametrización de la orientación de dicho plano. Para ello se solicita crear una función que reciba los arreglos de las posiciones 3D, los ángulos α y β, y retorne arreglos que representan las proyecciones a coordenadas 2D de las mediciones anteriores. Para encontrar los valores de x e y, solo deben efectuar el producto punto entre los puntos 3D y los vectores unitarios $\hat{x}$ e $\hat{y}$. Es decir si se tiene una medición en X_k, entonces $x_k =< X_k, \hat{x} >$ y $y_k =< X_k, \hat{y} >$. Dichos datos de proyección se procesan para generar estimaciones por cada pixel (malla regular) de una visualización.

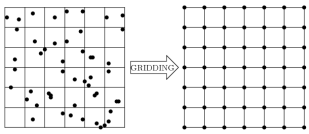

Figure 2: El procedimiento de gridding

La malla regular esta equis-espaciada a intervalos Δ, la esquina superior izquierda corresponde al punto (-1,1), la esquina inferior derecha corresponde al punto (1,-1), y existen $n = 256$ pixeles por fila y columna. Es decir es una matriz de $n \times n$. El radio r es un parámetro, por ejemplo se puede testear con $r = 2 * \Delta$. La operación de gridding se entrega como una función que recibe como argumentos la proyección de la nube de puntos, los valores medidos, sus pesos asociados, Δ y n. Esta función retorna una matriz de $n \times n$ con los valores promediados en los entornos de cada pixel.

Pueden chequear su algoritmo utilizando lo siguiente:

```
N=10000 # numero de mediciones
n=256  # tamaño de grilla
low=-1
high=1
delta=(high-low)/n
r=2*delta
X= np.random.uniform(low=low,high=high,size=(3*N)) # 3 vectores de coordenadas de las mediciones
X=np.reshape(X,(N,3)) # N filas de mediciones y 3 columnas de coordenadas 3D entre -1 y 1
w= 1+np.random.uniform(low=-0.1,high=0.1, size=(N)) # pesos de las mediciones
```

Los valores medidos corresponderá a otro arreglo de tamaño N y queda a su elección. La idea es que sea validante de su algoritmo, es decir sirva para propósitos demostrativos.

4 Experimentos a realizar

Debe concebir un conjunto de datos por ud mismo que sea demostrativo para la implementación que debe realizar.

1. (1 pto) Documente en forma clara y prolija en jupyter notebook la resolución de sus tarea y experimentos realizados. Describa en detalle su enfoque utilizado e interprete sus resultados.

2. (2 pts) Implemente la proyección 3D a 2D sin utilizar loop bajo la forma de una función. Realice experimentos que demuestren la validez de su experimento.

3. (2 ptos, 0 puntos si no tiene nada de la anterior) implemente la operación de gridding sobre los puntos proyectados. Para ello implemente una función que retorne la matriz de valores sin utilizar ciclos de python. Realice experimentos que demuestren la validez de su experimento.

4. (1 pto) Utilice una distribución inicial 3D que puede samplear para generar los datos y grafique el resultado del gridding. Indique el tiempo usado para dos casos: cupy vs numpy. Interprete resultados comparando con el tiempo usado en una implementación basada en ciclos de python.

5 Entrega

Debe compartir enlace a dicho Jupyter Notebook en classroom con plazo al día Jueves 11 de Abril antes de las 23:55 hrs.