Transformata Fourier Rapida pe GPU

Moraru Radu-Andrei, Grupa 462
Blogu Toma, Grupa 462
Matache Alexandru, Grupa 462

Motivatie

- Transformata Fourier este esentiala in procesarea imaginilor ⟶ importanta intr-un graphics pipeline
- FFT este un algoritm cu potential inalt de paralelizare ⟶ poate fi scris ca un set de instructiuni paralelizate (cu operatii SIMD)

Algoritmul FFT

$W_{N/2} = e^{-2j\pi/(N/2)}$ $\quad f^e(x) = f(2x)$

$F`(u) = NF(u)$ $\quad f^o(x) = f(2x+1)$

$$F`(u) = \sum_{x=0\ldots N-1} f(x)(W_N)^{xu} =$$

$$\sum_{x=0\ldots N/2} f^e(x)(W_{N/2})^{xu} + (W_N)^u \sum_{x=0\ldots N/2} f^o(x)(W_{N/2})^{xu} =>$$

$$F`(u) = F`^e(u) + (W_N)^u F`^o(u)$$

Definitie recurenta → F` poate fi calculat printr-un algoritm divide et impera / programare dinamica

Partitionarea sumei pentru TF

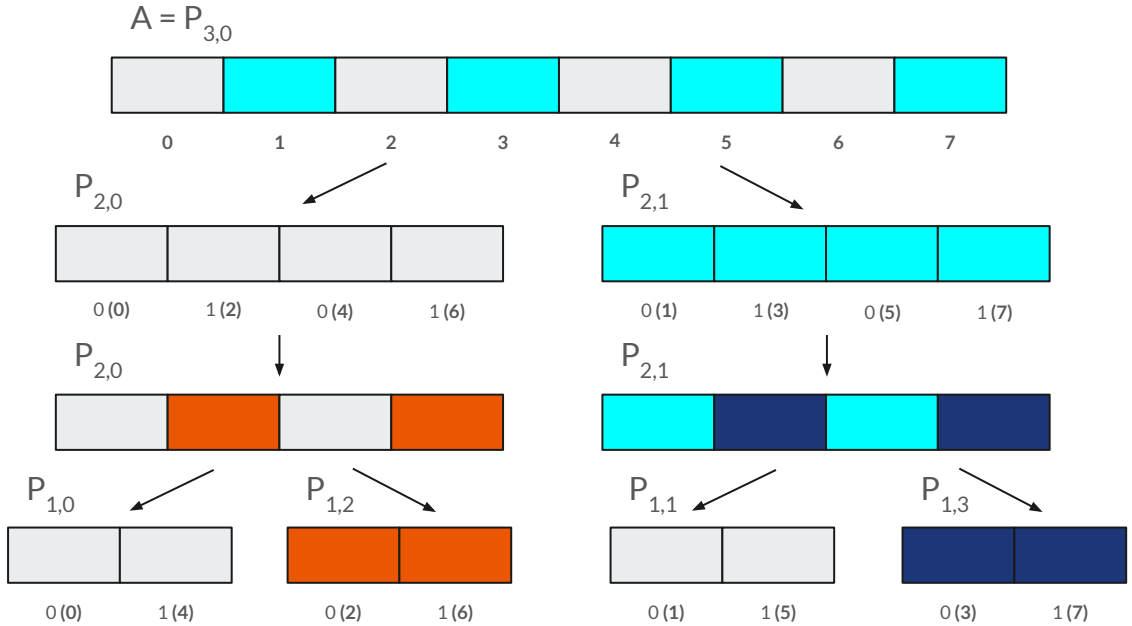
Definim partitiile dupa $P_{i,p}[u] \equiv A[n]$

$i \leftarrow 0 \ldots \log N$	Numarul iteratiei
$p = n \bmod N/2^i$	Numarul partitiei
$u = n \operatorname{div} N/2^i$	Indexul in partitie

Solutie DP bottom-up:

La iteratia i, salvam in A[n] transformata Fourier a partitiei $P_{i,p}$ in u.

La iteration i = logN, $A=P_{\log N,0}$, deci algoritmul va calcula transformata Fourier a lui A.



*Indexul original din matricea A notat in paranteze **(i)**

Solutie DP - formula de recurenta

Fie $M = N/2^i$, $L = 2^i$

Rescriem formula de recurenta:

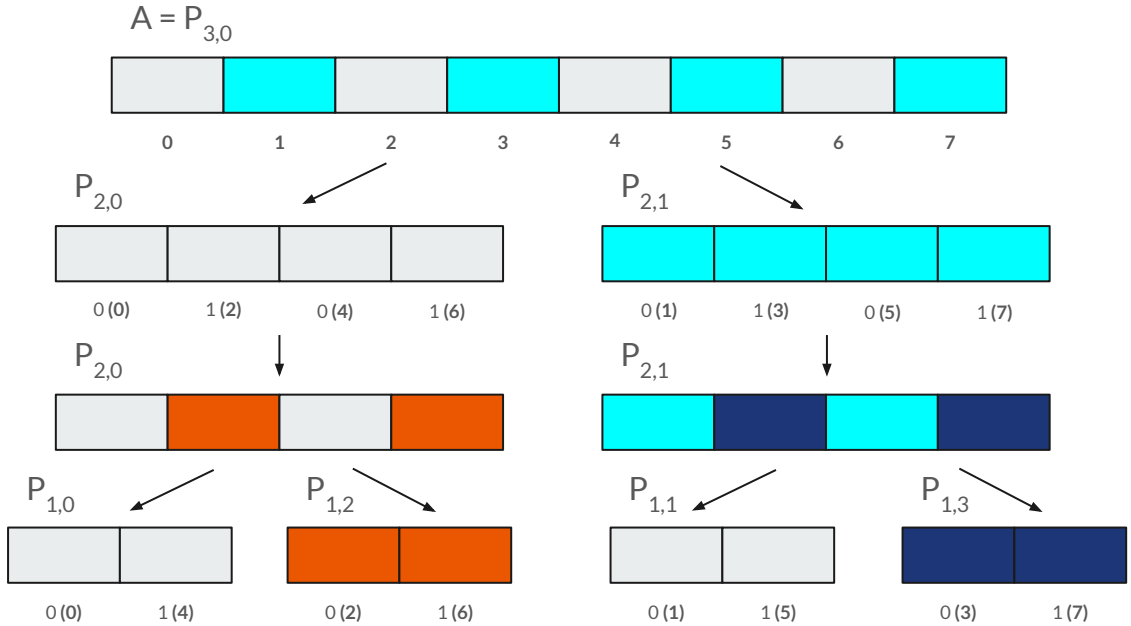
$P_{i,p}[u] \leftarrow P_{i-1,p}[u] + (W_L)^u P_{i-1,p+M}[u]$

Rescrisa in functie de A:

$A[n] \leftarrow A[n+uM] + (W_L)^u A[n+uM + M]$

Formula aplicata la fiecare iteratie $i \leftarrow 1 \ldots \log N$

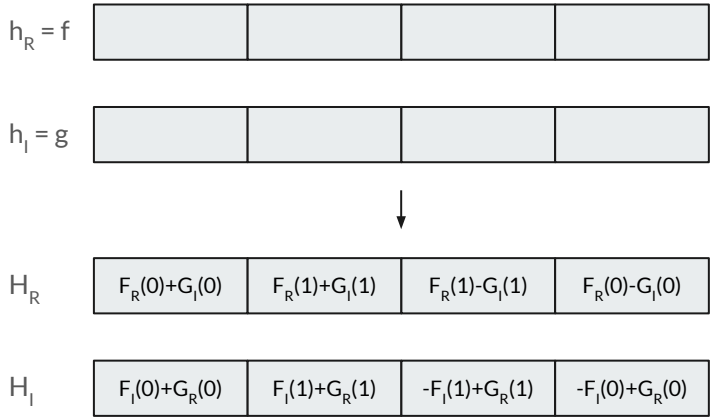
- Trebuie aplicata simultan pe toti indexii lui A,
- indexarea se face modulo N.



*Indexul original din matricea A notat in paranteze **(i)**

Frequency Compression - Tangling

- Fie f si g doua randuri de pixeli din imagine (valori reale)
- Fie h(x) = f(x) + jg(x) o functie complexa
- Transformata Fourier este o transformare liniara => H(x) = F(x) + jG(x)
- Putem sa ne folosim de faptul ca valoarea transformatei Fourier in n este complementul valorii din N-n pentru a scoate valorile lui F si G din H

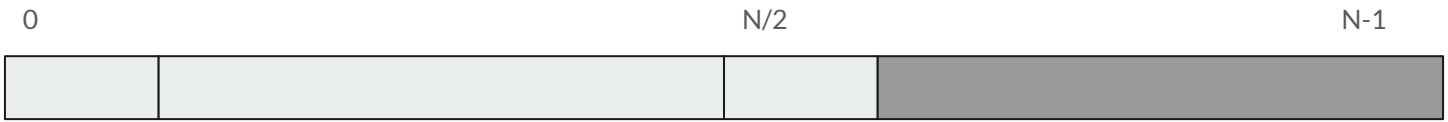


Frequency Compression - Packing

- Pentru fiecare vector F, in loc sa salvam valorile lui F in cate doua variabile, putem folosi proprietatea de complementaritate:

  $F_R[n] = F_R[N-n]$ si

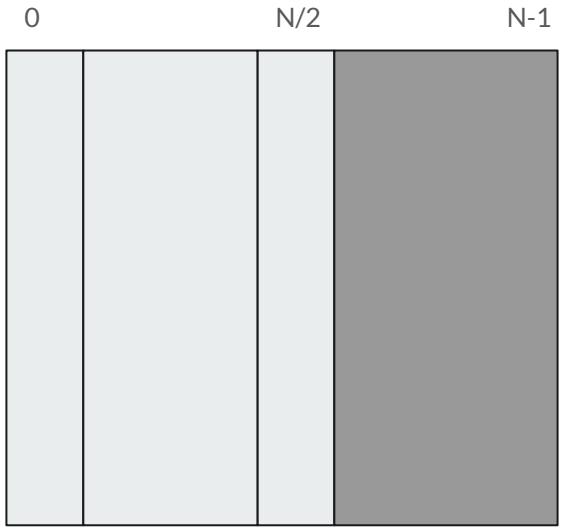
  $F_I[n] = -F_I[N-n]$, cu exceptia cazului n=0 sau n=N/2



- Pentru n > N/2 salvam partea imaginara
- Pentru n <= N/2 salvam partea reala

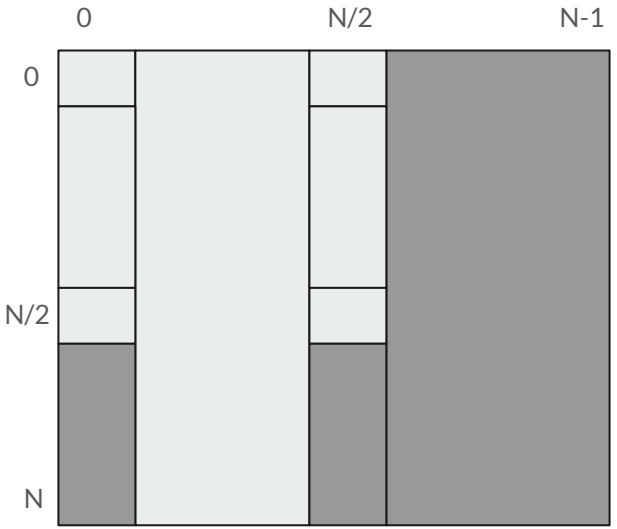
Rezultat dupa calculul pe randuri

- Coloanele 0 si N/2 contin valori “reale”, care nu trebuie sa fie unpacked. Pot fi imperecheate si calculate la fel ca doua randuri.
- Restul coloanelor trebuie imperecheate n cu N-n. Ele vor fi partea reala, respectiv cea imaginara si se poate calcula FFT fara tangling.



Rezultat final

- Coloanele 0 si N/2 contin valori “reale”, care nu trebuie sa fie unpacked. Pot fi imperecheate si calculate la fel ca doua randuri.
- Restul coloanelor trebuie imperecheate n cu N-n. Ele vor fi partea reala, respectiv cea imaginara si se poate calcula FFT fara tangling.



Bibliografie

- Kenneth Moreland and Edward Angel, The FFT on a GPU (2003)