Filtros de Convolução em Imagens Gigantescas

Bayard da Rocha

INTELIGÊNCIA NA WEB E BIG DATA

Prof. Fabrício Olivetti de França

Maio/2018

Objetivo

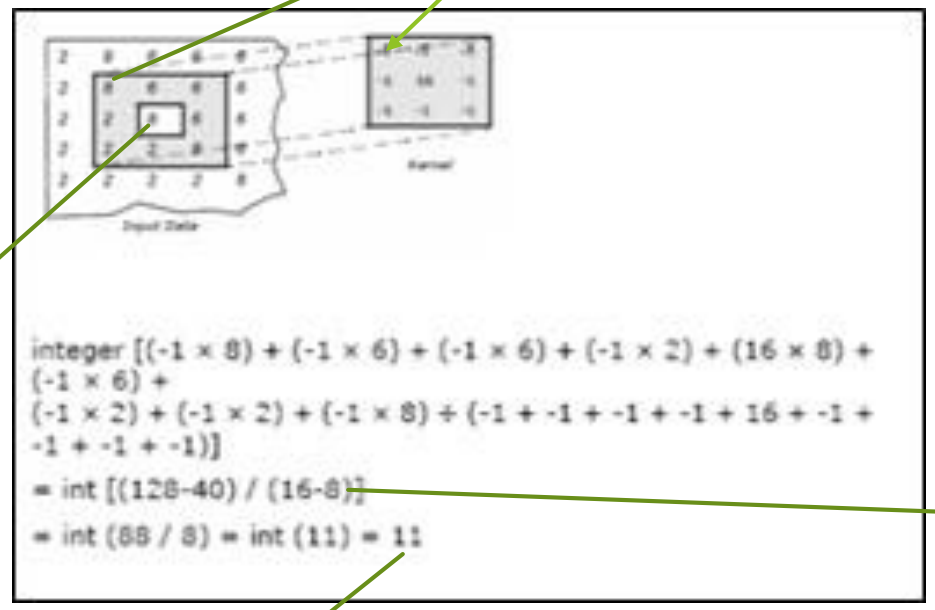
- Desenvolver Algoritmo em Python para Filtros de Convolução utilizando 2 técnicas distintas :
  - Técnica Sequencial
  - Técnica de Paralelismo
- Aplicar os Algorítmos em Imagens Gigantescas
- Avaliar comportamento do tempo de processamento aplicando as 2 técnicas

Conceitos

- **Convolução** → efeito de filtro de propósito geral para imagens (desfocagem, nitidez, mapa de relevo, dentre outros)
- **Técnica** → aplicar uma matriz que chamamos de Kernel a uma matriz geradora da Imagem
- **Imagens Gigantescas** → acima de 1.000 pixels em suas dimensões

Aplicação do Filtro de Convolução

A técnica



(-1x8)

Alvo

Somatória do Kernel

Novo valor do Alvo

$$V = \left| \frac{\sum_{i=1} \left(\sum_{j=1} f_{ij} d_{ij} \right)}{F} \right|$$

Equação Matemática

Algoritmo sequencial

- Converter Imagem em matriz
  - Resultante 3 matrizes r,g,b
- Varredura sequencial nas três matrizes criando submatrizes com mesmas dimensões do Kernel (3x3)
- Aplicar somatórias para calculo do novo valor do Pixel Alvo

Algoritmo sequencial

```
for a in range (np.array(img_foto).shape [2]):
    matrizConvolu = np.copy(matrizBGR[a])
    for x in range (matrizBGR[a].shape [0]-2):
            for y in range (matrizBGR[a].shape [1]-2):
                    subMatriz = matrizBGR[a][slice(x,x+3),slice (y,y+3)]
                    matrizConvolu [x+1][y+1] = int(sum(np.reshape(subMatriz*kernel,-1)) / somaKernel)
                    if  matrizConvolu [x+1][y+1] < 0:
                            print (matrizConvolu [x+1][y+1])
                            matrizConvolu [x+1][y+1] = 0
                    qtdeOperac = qtdeOperac + 1
    matrizBGR[a] = matrizConvolu
```

Onde aplicar o paralelismo

Algoritmo com Paralelismo

- criar de um RDD no formato ((x,y), (r,g,b))
- criar submatrizes 3x3 com estrutura

  (x,y), (coordenada-submatriz, posição na submatriz) ,rgb)
- aplicar **map** para incorporar as coordenadas das submatrizes no RDD e na sequencia **reduce**
- aplicar Kernel

Resultado

- Utilizei imagem com dimensão 2041X1080 pixels
- Algoritmo sequencial executado em tempo médio de 65,5segs
- Algoritmo em paralelo não concluído
- Não foi possível realizar estudo comparativo

Obrigado