Projeto 4 – Segmentador de Imagem na GPU

André De Marco Toyama

https://github.com/AToyama/ImageSegmentation-GPU

INTRODUÇÃO

O projeto trata-se de um segmentador de imagem na GPU utilzando a ferramenta da nvidia Nvgraph para o algoritmo de caminho mínimo utilizado no processo de segmentação e a API da Nvidia de computação paralela para realizar o filtro de bordas na imagem.

SSSP (Single Source Shortest Path)

De forma simples, o algoritmo encontra o custo mínimo necessário para todos os pixels chegarem até um pixel semente, onde o custo de um pixel para o outro é o módulo da diferença de seus valores. Assim damos uma semente de frente e uma de fundo, cálculamos os custo para ambos e verificamos qual é mais barato, ou seja qual tem o caminho com cores mais similares a semente, o que pode indicar que os pixels fazem parte do mesmo elemento.

FUNCIONAMENTO

Dando-se uma imagem de entrada e definindo suas sementes de frente e semente de fundo, ele realiza a segmentação da imagem. Primeiro a imagem é tratada com um filtro de borda Laplaciano, para que haja mais facilidade no algoritmo de caminho mínimo utilizado para a segmentação. Tanto para a versão sequencial quanto a versão com nvgraph, esse filro é realizado através de um kernel de CUDA (API da NNIDIA de computação paralela). Já o cálculo do caminho mínimo, na sequencial é utilizado uma função sequencial comum, já a versão com Nvgraph utiliza uma função pronta do mesmo.

O objetivo das diferentes versões é ao final realizar um teste de desempenho entre as duas versões, avaliando o tempo de execução das diferentes partes do código.

As medidas de tempo são tiradas utilizando uma ferramente do CUDA Toolkit chamado “cuda_runtime” .

Foi-se utilizado a biblioteca “thrust” para que os cálculos do filtro de bordas realizado no CUDA, seja feito na GPU (unidade de processamento gráfico), onde o processo pode ocorrer mais raídamente. Basicamente, a GPU passa a operar como se fosse mais uma CPU, aumentando a performance do sistema, aproveitando também do fato que o CUDA realiza a função em multiplos blocos com multiplas threads executando em paralelo diferentes seções da imagem.

DADOS

Para a comparação final foram tiradas as seguintes medidas de tempo:

- Tempo para construção do grafo (Apenas na versão Nvgraph)
- Tempo para o cálculo de caminho mínino
- Tempo para geração da imagem de saída
- Tempo Total para execução do programa

A imagem de saída terá o nome de acordo com o nome colocado no início da execução do programa. Para observarmos a segmentação, os pixels com o custo do caminho menor referente a semente de frente, será colorido de branco, já se for referente a semente de fundo, será colorido de preto.

SISTEMA

CPU: Intel Core i5-4210U CPU @ 2.7GHz
GPU: Mesa DRI Intel(R) Haswell Mobile
RAM: 8GB DDR3

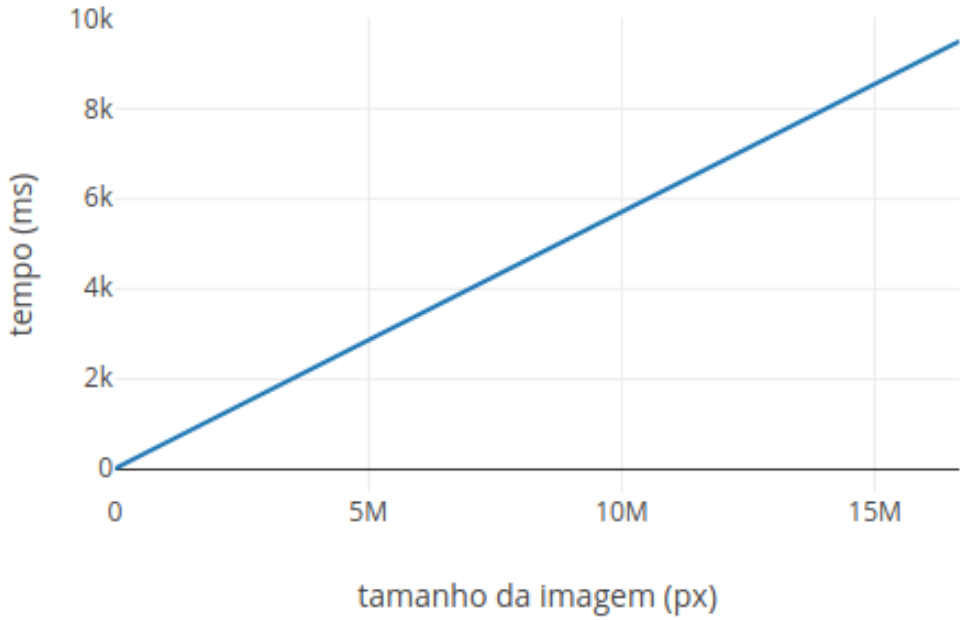
RESULTADOS

Abaixo temos o resultado comparando o tempo de execução para a versão sequencial e a versão com nvgraph com imagens de 3 diferentes tamanho.

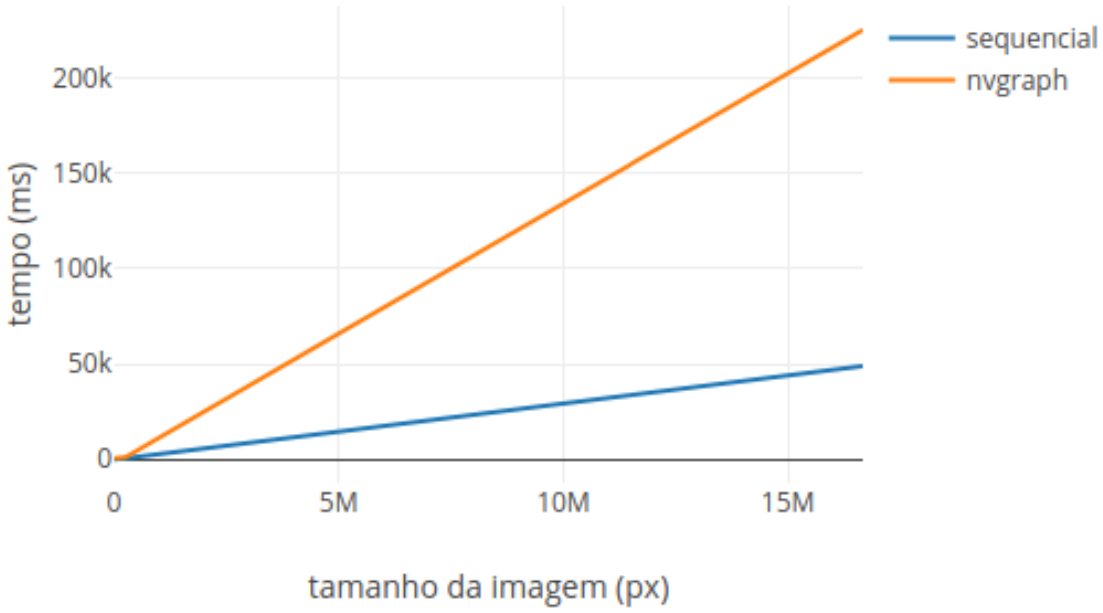
- teste1.pgm (10X10)
- teste.pgm (500X500)
- lemon.pgm (5000X3333)

As imagens utilizadas estão no formato PGM, que é basicamente uma escala em preto e branco onde o valor do pixel varia de 0 (preto) a 255 (branco).

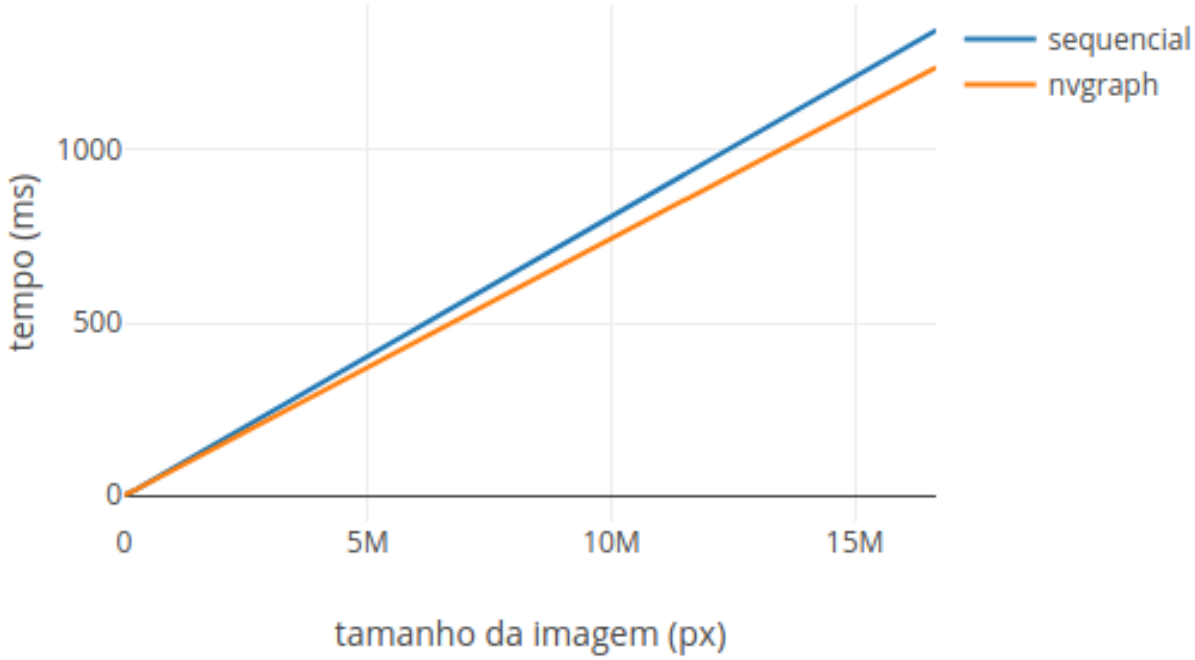
Tempo para Construção do Grafo:



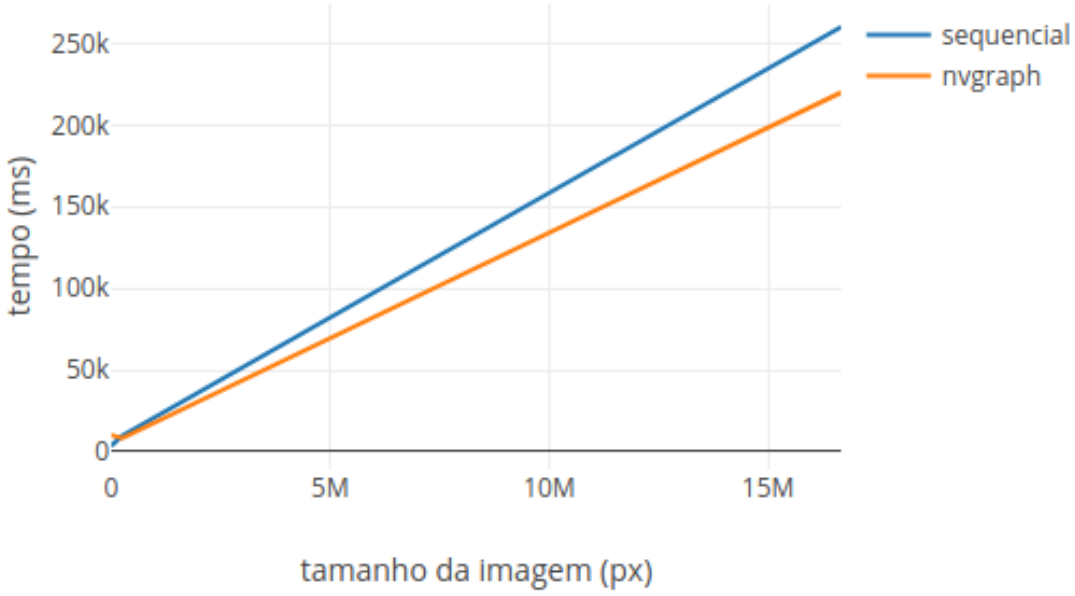
Tempo para execução da função SSSP (Single Source Shortest Path):



Tempo para geração da imagem de saída



Tempo Total de Execução do Programa



Embora a função do nvgraph esteje funcionando, ainda não há uma gritante diferença de performance apresentada. A seção do cálculo SSSP apresentou um tempo melhor na versão sequencial, mas no tempo total de execuçaão do programa, vemos uma melhora na versão do nvgraph crescendo diretamente proporcional ao tamanho da imagem.

REPRODUTIBILIDADE

No README do repositório do projeto tem as instruções necessárias para rodar e realizar os teste por conta própria, basta acessar o link disponível no começo do relatório.

MELHORIAS

A melhora de desempenho menor do que esperada pode estar relacionada a uma perda de bytes presentes na versão do nvgraph, descobertas através da ferramente de debug e análise de memória chamada “Valgrind”. Como melhorias espero tratar esse vazamento de memória para obter uma melhora de desempenho mais interessante.