Concorrência aplicada a Segmentação de Imagens pelo método Limite (Thresholding)

Programação Concorrente (ICP-361) - Relatório Parcial

Diego Vasconcelos Schardosim de Matos

June 2024

Abstract

The image segmentation problem is one of several problems studied in the area of Computer Vision, is the process of dividing a digital image into multiple regions (set of pixels) or objects, with the aim of simplifying and/or changing the representation of an image to facilitate its analysis. It is typically used to locate objects and shapes (lines, curves, etc.) in images.

There are several Segmentation methods, in this project I will be dividing the image in several Regions of Interests (ROI), iterate though each ROI and appling a manual Thresholding preprocessing method to it, which is one of the simplest algorithms. This will be made in sequential and concurrency mode and at the end compare the results.

1 Descrição do problema

Em visão computacional, segmentação se refere ao processo de dividir uma imagem digital em múltiplas regiões (conjunto de pixels) ou objetos, com o objetivo de simplificar e/ou mudar a representação de uma imagem para facilitar a sua análise. Segmentação de imagens é tipicamente usada para localizar objetos e formas (linhas, curvas, etc) em imagens.

O resultado da segmentação de imagens é um conjunto de regiões/objetos ou um conjunto de contornos extraídos da imagem (detecção de borda). Como resultado, cada um dos pixels em uma mesma região é similar com referência a alguma característica ou propriedade computacional, tais como cor, intensidade, textura ou continuidade. Regiões adjacentes devem possuir diferenças significativas com respeito a mesma característica(s).

1.1 Classificação

Os processos de segmentação podem ser classificados como automáticos, manuais ou semiautomáticos, sendo diferenciados pela fato de haver ou não intervenção humana. Nos processos automáticos todas as etapas são realizadas sem

a ação humana. Geralmente são muito utilizados por robôs que tomam decisões a partir da interpretação das imagens obtidas, como exemplo tem-se a aplicação nos carros autônomos. Nos processos de segmentação manual, por outro lado, toda a segmentação é feita com o auxílio do usuário, cabendo a ele utilizar as ferramentas e técnicas que melhor se adequem a imagem e a sua necessidade, é frequentemente usada em situações muito específicas onde as condições exteriores como luminosidade e ruídos prejudiquem o uso dos algoritmos tradicionais, também é muito utilizada ao se fazer testes que buscam a melhoria e aperfeiçoamento das técnicas.

1.2 Método de pré-processamento por Limite (Thresholding)

Um dos métodos mais simples de segmentação de imagens, o thresholding é um algoritmo que transforma uma imagem em escala de cinza para uma imagem binária, onde cada pixel possui intensidade mínima ou máxima (ex: 0 ou 1, 0 ou 255, etc).

Figure 1: Display de temperatura - Imagem original

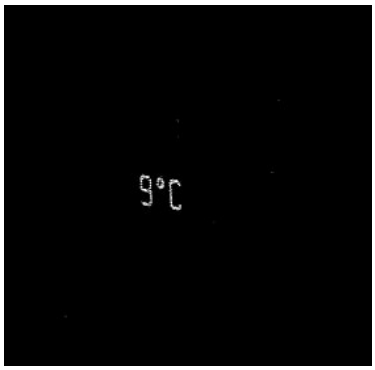


Figure 2: Display de temperatura - Threshold aplicado invertido

O fator principal para bons resultados com esse algoritmos é encontrar um bom valor de limite de intensidade, isso pode ser feito de forma manual ou automática.

1.3 Baseados em Histograma

Podemos combinar essa ideia com o métodos de histogramas, são muito eficientes em comparação com outros métodos de segmentação de imagem, porque eles normalmente exigem apenas uma passagem pelos pixels. Nesta técnica, um histograma é calculado a partir de todos os pixels da imagem, e os picos e vales no histograma são usados para localizar os clusters na imagem. Cor ou intensidade pode ser usada como medida.

Uma desvantagem do método de busca de histograma é que pode ser difícil identificar picos e vales significativos na imagem.

1.4 Como funciona

Recebendo como entrada uma imagem em escala de cinza representada por uma matriz M_{2x2}, seja L o valor máximo que um pixel pode ter, L_{ij} a intensidade do pixel na posição i, j da matriz, cada pixel da imagem pertence ao intervalo [0,L]. Após determinar um valor constante k de intensidade, analisar cada pixel da imagem. Se a intensidade do pixel for maior que a constante definida o pixel passará a ser branco, se for menor será preto (ou ao contrário, não importa), fazendo assim uma imagem binária.

$$\begin{cases} 1, se L_{ij} <= k \\ 0, casocontrario \end{cases} \tag{1}$$

Para melhor eficácia desse método é necessário um bom valor para constante, no geral isso é feito analisando um Histograma onde o eixo x pertence ao intervalo [0,L] e o y a contagem de pixels da imagem para cada valor de intensidade. É comum neste histograma existir um vale formando duas classes C_0 e C_1, e a constante separe bem essas classes. Geralmente são o fundo da imagem e o objeto.

Este processo também pode ser feito de forma automatizada, a forma mais comum é pelo método de Otsu (variância máxima) que não será abordado neste trabalho.

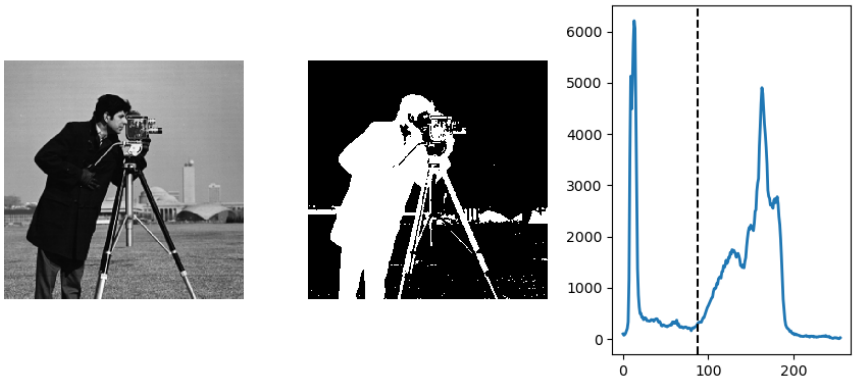


Figure 3: Demonstração do método Thresholding para geração de uma imagem binária, destacando o fotógrafo da imagem

1.5 Concorrência

Para uma solução concorrente este algoritmo pode se beneficiar da representação matricial da imagem e também do fato do método ser independente dos pixels vizinhos para ser disparado a leitura e escrita em fluxos de execução distintos.

2 Projeto da solução

Este projeto será implementando em C++ usando a biblioteca Qt6 para GUI e gerenciamento das Threads, OpenCV para o gerenciamento das imagens. Estarei aplicando o método Limite manual encontrado na biblioteca OpenCV.

O usuário através da interface gráfica irá selecionar uma imagem como entrada (pode ser colorida) e ao clicar para executar (Run) irá visualizar todas as etapas realizadas:

1. imagem em escala de cinza
2. histograma
3. imagem binária

Para contextualizar um pouco a finalidade do método, ao final do programa a imagem binária será usada como máscara na imagem original com o intuito de remover uma das classes da cena, como por exemplo: remover o fundo da imagem.

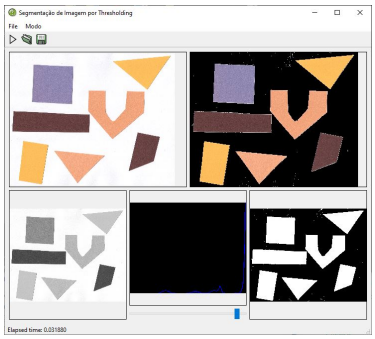


Figure 4: Demonstração para separação de objetos do fundo

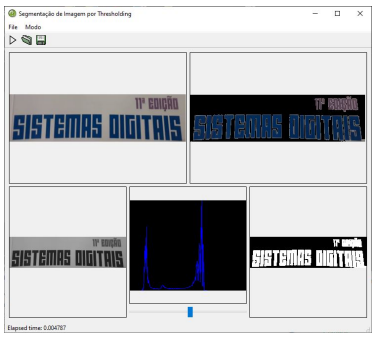


Figure 5: Demonstração da separação do texto do fundo da imagem

O usuário também poderá através da Gui alternar entre os modos de execução sequencial e concorrente do algoritmo, além ajustar o valor do limite aplicado.

2.1 Implementação do modo concorrente

Neste projeto a concorrência será feita usando o módulo QtConcurrency do Qt6, a vantagem de chamar o módulo da própria biblioteca é a garantia de funcionar em qualquer dispositivo (portabilidade) e as abstrações que a biblioteca trás para resolver o problema, as seguintes estratégias são válidas:

- Disparar N Threads usando QtConcurrency, e com o OpenCv iterar pixel por pixel da imagem de forma alternada, cada thread usando exclusão mutua para aplicando ler o pixel, aplicar método e escrever numa imagem resultado o pixel modificado.
- Disparar N Threads usando QtConcurrency e passando como argumento o início e o fim de leitura, cada thread fica responsável por uma região diferente usando exclusão mútua para escrever numa imagem resultado o pixel modificado.
- Dividir a imagem em N regiões de interesse (ROI) horizontais completas, e atribuir para cada Thread uma região para aplicar o método.

Entre todas alternativas pensados pelo autor, o mais simples e direto pareceu ser o 3 item. Como cada região é uma cópia verdadeira da matriz horizontal não é necessário uso de exclusão mútua ou sincronização por barreira pois cada thread estará escrevendo em regiões de memória distintas.

Para isso o Qt possui um método chamado blockingMapped, que recebe uma lista, uma função que deve ser executada e aguarda o fim da execução das threads. Esse método garante que executada em concorrenci e é uma função com efeito colateral, ou seja, não irá retornar nada e irá modificar o item da lista original.

```
QtConcurrent::blockingMap(imageParts, [&](cv::Mat &imagePart) {
    cv::threshold(imagePart, imagePart, this->thresholding, 255, cv::THRESH_BINARY_INV);
});
cv::vconcat(imageParts, this->binaryImage);
```

Figure 6: Uso do método blockingMapped. Recebe uma lista de ROIs e cada thread aplicada o método de thresholding para uma região diferente

Ao final as regiões devem juntadas para formar a imagem final. Além disso, este método de divisão da imagem em ROI e aplicar o método também será usada para converter a imagem colorida em escala de cinza e transforma-la em imagem binária.

Não é necessário uso sincronização por barreira pois cada etapa é disjunta e como uma depende do resultado da anterior, o programa só avança quando a anterior completar.

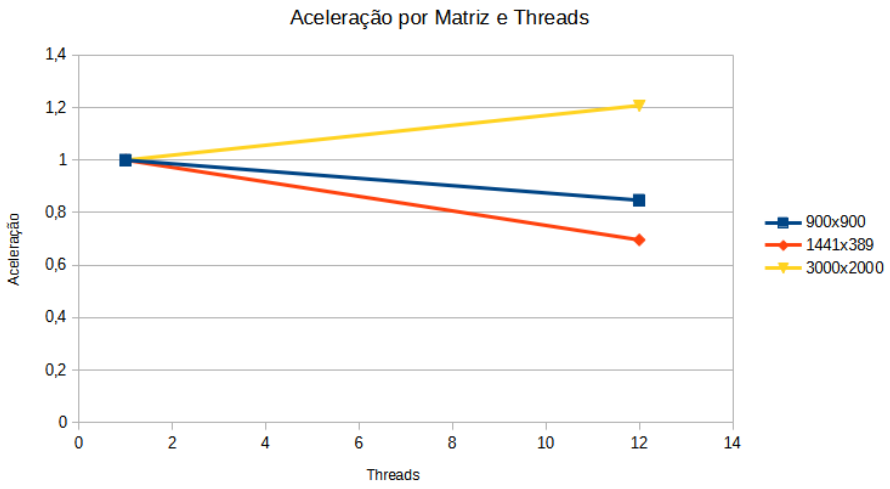
2.2 Implementação do modo sequencial

Para garantir uma boa comparação de resultados entre o modo sequencial e o concorrente, devemos tentar fazer eles o mais próximo possível. Para isso, o modo sequencial do programa irá fazer as mesmas coisas que o concorrente mas ao invés de disparar as threads para cada região da imagem, irá iterar sequencialmente entre cada região e aplicar o método.

3 Casos de teste de corretude e desempenho

Para o benchmark do programa estarei usando uma função própria do OpenCV para calcular o intervalo de tempo, exibindo a duração do algoritmo no rodapé do programa. Como entrada imagens de artigos, sites, vídeos da internet onde estarei tentando replicar os seus resultados visuais.

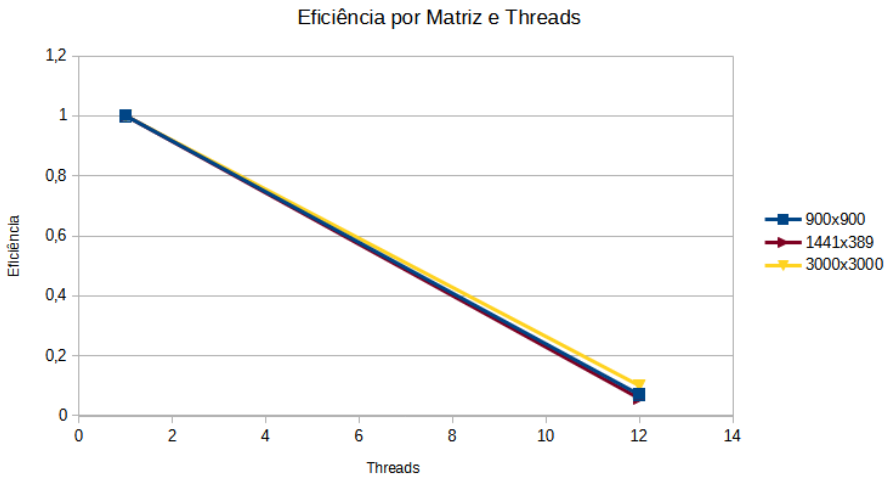
Os testes foram feitos numa máquina Windows 10 64bits com 12 threads, foram coletadas a média de 3 amostras de execução para cada modo diferente do programa: sequencial e concorrente. Usando como entradas uma imagem pequena (900x900), média (1441x389) e grande (3000x2000)



4 Discussão

Era esperado que nem para todos os casos o custo do overhead para criação das threads ganharia do sequencial, porém não pensei que começaria a ter alguma vantagem a partir de imagens de alta resolução (grandes).

A partir da análise dos resultados conseguimos identificar melhora na aceleração no modo concorrente para imagem grande e que para imagens pequenas e



médias o custo do overhead da criação das threads não vence o modo sequencial do programa para nenhum quesito.

O programa apresenta 2 bugs: ao executar Run várias vezes a máscara da imagem binária aplicada na imagem original não é atualizada e também a janela do programa aumenta de tamanho a cada execução.

Seria interessante também implementar um algoritmo que não aloca um novo espaço de memoria e realiza a cópia das regiões de interesse, mas sim que itere pixel a pixel e vá direto para implementação concorrente.

4.1 Dificuldades

Esta foi minha primeira vez desenvolvendo um programa uma Gui em c++ usando Qt, então foi um grande aprendizado desde entender como configura o ambiente de desenvolvimento (compilando as bibliotecas, etc), aprender a usar as bibliotecas, aplicar para meu interesse e no fim exportar um binário para windows e linux (que falhei).

5 Referências bibliográficas

- https://pt.wikipedia.org/wiki/Segmenta%C3%A7%C3%A3o_(processamento_de_imagem)
- https://datacarpentry.org/image-processing/index.html
- https://www.ibm.com/br-pt/topics/image-segmentation
- https://www.tensorflow.org/tutorials/images/segmentation?hl=pt-br
- https://visaocomputacional.com.br/identificacao-deteccao-reconhecimento-e-segmentacao-de-imagem-e-objetos/

- https://en.wikipedia.org/wiki/Otsu%27s_method#Limitations_and_variations
- https://doc.qt.io/qt-6/qtconcurrentmap.html