Relatório AED

Universidade de Aveiro

João Pinto, Leonardo Luís

Relatório AED

Dept. de Eletrónica, Telecomunicações e Informática
Universidade de Aveiro

João Pinto, Leonardo Luís
(113093) joaop1@ua.pt, (114093) leonardo.luis@ua.pt

26 de novembro de 2023

Conteúdo

1 Análise de Funções **1**
1.1 ImageLocateSubImage() 1
1.1.1 Análise Formal 1
1.1.2 Dados experimentais 2
1.2 ImageBlur() 3
1.2.1 Análise Formal 3
1.2.2 Dados experimentais 4
1.2.3 Análise Comparativa Algoritmo Básico/Algoritmo Melhorado 5

Capítulo 1

Análise de Funções

Neste capitulo iremos abordar a analise formal, entre esta temos a analise do melhor e pior caso possível para cada função, o cálculo da expressão matemática que dará a complexidade teorica (que será comparada com valores testados) e os dados experimentais da função ImageLocateSubImage() e da função ImageBlur().

1.1 ImageLocateSubImage()

A função ImageLocateSubImage é usada para encontrar uma subimagem específica dentro de uma imagem maior. O objetivo desta função é identificar a localização de uma imagem menor (subimagem) dentro de outra imagem maior. Muito resumidamente, esta função vai percorrendo a imagem maior e comparando cada região dela com a subimagem, utilizando algum critério de comparação, neste caso, a igualdade dos valores dos pixels em hexadecimal. Quando uma correspondência é encontrada, a função retorna as coordenadas da localização da subimagem na imagem maior. Esta função é particularmente útil em aplicações de reconhecimento de padrões ou em situações onde é necessário identificar a presença e a posição de um objeto específico dentro de uma imagem.

1.1.1 Análise Formal

Nesta função, começamos por analisar o pior e o melhor caso possível da função.

- O **melhor caso** será quando a subimagem que pretendemos encontrar em uma outra imagem pretendida é encontrada na primeira posição verificada.

- O **pior caso** será quando a subimagem que pretendemos encontrar em uma outra imagem pretendida não existe ou é encontrada na última posição possível.

- A análise da complexidade teorica é nos dada por:

    1. **Iteração por Cada Pixel em** `img1`: A função precisa verificar cada posição em `img1` onde `img2` poderia começar. Isso envolve iterar sobre todas as posições (x, y) em `img1`. Se `img1` tem dimensões $W_1 \times H_1$ (largura W_1 e altura H_1), o número de posições a verificar é proporcional a $W_1 \times H_1$.
    2. **Verificação de Correspondência em Cada Posição (**`ImageMatchSubImage`**)**: Para cada posição em `img1`, a função `ImageMatchSubImage` é chamada. Esta função verifica se `img2` corresponde à subárea de `img1` começando naquela posição. Se `img2` tem dimensões $W_2 \times H_2$, então a função `ImageMatchSubImage` tem uma complexidade de tempo de $O(W_2 \times H_2)$ para cada chamada.
    3. **Complexidade Total**: A complexidade total de tempo da função `ImageLocateSubImage` é, portanto, $O(W_1 \times H_1 \times W_2 \times H_2)$.

    Com isto, podemos concluir que a complexidade de tempo da função `ImageLocateSubImage` é quadrática em relação às dimensões das imagens. Isso significa que o tempo de processamento aumenta rapidamente com o aumento do tamanho das imagens. Isso significa que o tempo necessário para executar a função aumenta rapidamente à medida que o tamanho das imagens aumenta. Em casos práticos, especialmente com imagens grandes ou quando a subimagem é uma fração significativa do tamanho da imagem maior, esta operação pode ser bastante custosa em termos de tempo de processamento.

1.1.2 Dados experimentais

Utilizando o ficheiro `testes2.c`, realizamos uma série de testes para avaliar o desempenho da função `ImageLocateSubImage`. Esta função é essencial em várias aplicações de processamento de imagem, como no reconhecimento de padrões e na busca por características específicas dentro de imagens maiores.

Metodologia

Nos testes, utilizamos uma imagem principal de tamanho médio (`tools_2_765x460.pgm`) em conjunto com diversas subimagens de tamanhos variados. O objetivo era determinar a eficácia da função em localizar estas subimagens dentro da imagem maior, medindo tanto o número de comparações necessárias quanto o tempo de execução.

Resultados Obtidos

Os resultados indicaram uma variação no número de comparações e nos tempos de execução, dependendo do tamanho e da complexidade da subimagem. Por exemplo, ao testar com a subimagem `bird_256x256.pgm`, foram necessárias

104550 comparações, com um tempo de execução de 0.010778 segundos. Já com a subimagem `art4_300x300.pgm`, o número de comparações foi de 75026, com um tempo de execução de 0.050653 segundos. Um teste adicional com a subimagem `art3_222x217.pgm` mostrou 132736 comparações e um tempo de execução de 0.037245 segundos.

Discussão

Os testes demonstraram que a função `ImageLocateSubImage` é sensível ao tamanho e à complexidade da subimagem. A variação no número de comparações e no tempo de execução sugere que a otimização desta função pode ser necessária, especialmente para aplicações que exigem rapidez e eficiência na busca por subimagens em imagens de grandes dimensões.

Tabela 1.1: Resultados do Teste da Função `ImageLocateSubImage`

Imagem Principal	Subimagem	Número de Comparações	Tempo de Execução (s)
tools_2_765x460.pgm	bird_256x256.pgm	104550	0.010778
tools_2_765x460.pgm	art4_300x300.pgm	75026	0.050653
tools_2_765x460.pgm	art3_222x217.pgm	132736	0.037245

1.2 ImageBlur()

A função ImageBlur é utilizada para aplicar um efeito de desfocagem numa imagem. Este processo consiste em substituir o valor de cada pixel na imagem pelo valor médio dos pixels na sua vizinhança, criando assim um efeito visual suavizado. Na implementação mais básica, isto é feito calculando a média dos valores dos pixels numa janela ao redor de cada pixel (definida pelos parâmetros dx e dy). Esta média é depois aplicada ao pixel central da janela, efetivamente desfocando a imagem. Este processo suaviza as transições de cor e textura, resultando numa imagem com aspeto mais 'macio' e menos definido.

1.2.1 Análise Formal

- função `ImageBlur` aplica um efeito de desfoque em uma imagem utilizando um filtro de média. A análise da sua complexidade teórica é feita em termos de complexidade de tempo e é nos dada por:

  1. **Iteração Sobre Cada Pixel da Imagem**: Para cada pixel na imagem, um loop duplo é utilizado para iterar sobre uma área retangular ao redor do pixel, definida por dx e dy. Se a imagem tem dimensões $W \times H$, o número de iterações é $W \times H$.
  2. **Cálculo da Média para Cada Pixel**: Para cada pixel, um segundo par de loops itera sobre uma região retangular ao redor do pixel, de tamanho $(2dx+1) \times (2dy+1)$. A complexidade para calcular a média para um pixel é proporcional a $(2dx + 1) \times (2dy + 1)$.

3. **Complexidade Total**: A complexidade total de tempo da função `ImageBlur` é $O(W \times H \times (2dx + 1) \times (2dy + 1))$.

A complexidade de tempo da função `ImageBlur` é quadrática em relação às dimensões da imagem e ao tamanho do filtro de média. O tempo de processamento aumenta com o aumento do tamanho da imagem e do filtro.

1.2.2 Dados experimentais

Utilizando o ficheiro `testes.c`, foi realizada uma série de testes para avaliar o desempenho da função `Blur` em diferentes cenários. Os testes envolveram a aplicação da função em três imagens de tamanhos distintos, utilizando um filtro de desfoque de tamanho 15x15.

Metodologia de Teste

Os testes foram executados em imagens de tamanhos variados - uma pequena (`art4_300x300.pgm`), uma média (`ireland-03_640x480.pgm`) e uma grande (`ireland-06-1200x1600.pgm`) - para medir o tempo de execução da função `Blur`.

Resultados Obtidos

Os resultados demonstraram um aumento progressivo do tempo de execução em função do tamanho da imagem. Para a imagem de menor tamanho, o tempo de execução foi de apenas 0.269474 segundos. No caso da imagem de tamanho médio, esse tempo subiu para 0.935286 segundos, enquanto a imagem grande registrou um tempo de 6.139191 segundos.

Discussão

A análise dos resultados indica que o desempenho da função `Blur` é fortemente influenciado pelo tamanho da imagem, evidenciando um aumento no tempo de processamento conforme o tamanho da imagem cresce. Isso ressalta a importância de considerar o tamanho da imagem em aplicações práticas que envolvem o uso da função `Blur`.

Tabela 1.2: Resultados do Teste da Função `ImageBlur`

Imagem	Dimensão da Imagem	Tamanho do Filtro	Tempo de Execução (s)
pgm/small/art4_300x300.pgm	300x300	15x15	0.269474
pgm/medium/ireland-03_640x480.pgm	640x480	15x15	0.935286
pgm/large/ireland-06-1200x1600.pgm	1200x1600	15x15	6.139191

1.2.3 Análise Comparativa Algoritmo Básico/Algoritmo Melhorado

Para realizar uma análise comparativa entre um algoritmo básico e um algoritmo melhorado da função `ImageBlur()`, consideramos as características distintivas de cada um.

Algoritmo Básico

O algoritmo básico utiliza um método de filtragem por média para desfocar a imagem.

- **Complexidade de Tempo**: $O(W \times H \times (2dx + 1) \times (2dy + 1))$, onde W e H são a largura e a altura da imagem, respectivamente, e dx e dy são os raios do filtro de média.
- **Desvantagens**:
  - Pode ser lento para imagens grandes ou filtros de média grandes.
  - Qualidade de desfoque básica, sem consideração por preservação de bordas.

Algoritmo Melhorado

Um algoritmo melhorado pode incluir otimizações como filtro de média ponderada ou preservação de bordas.

- **Complexidade de Tempo**: Varia de acordo com as otimizações, podendo se aproximar de $O(W \times H)$.
- **Vantagens**:
  - Maior eficiência para imagens grandes ou filtros de média grandes.
  - Melhor qualidade de desfoque, com técnicas como preservação de bordas.

Análise Comparativa

- **Desempenho**: O algoritmo melhorado tende a ser mais eficiente, especialmente para imagens de alta resolução ou filtros grandes.
- **Qualidade de Desfoque**: O algoritmo melhorado pode oferecer um desfoque de melhor qualidade.
- **Complexidade de Implementação**: O algoritmo melhorado requer um entendimento mais aprofundado de processamento de imagem e algoritmos de filtragem.

A escolha entre o algoritmo básico e o melhorado depende das necessidades específicas do projeto, como a importância do desempenho versus a qualidade do desfoque e os recursos disponíveis para a implementação.