



deti

universidade de aveiro
departamento de electrónica,
telecomunicações e informática

TAD image8bit

Trabalho realizado por:

Eduardo Lopes Nº 103070

Roberto Fontoura Nº 115178

Algoritmos e Estruturas de Dados

Prof. Joaquim Madeira

Prof. Pedro Lavrador

Ano Letivo 2023/2024



Índice

INTRODUÇÃO.....	3
ImageLocateSubImage	4
Dados experimentais	4
Análise formal.....	6
ImageBlur	6
Dados experimentais	6
Análise formal.....	7
CONCLUSÃO.....	8
BIBLIOGRAFIA	9

INTRODUÇÃO

Este projeto teve como objetivo a implementação de diversas funções de forma a completar o script “image8bit”. Este script permite a criação e manipulação de imagens do tipo “.pgm”, isto é, imagens em que cada pixel pode tomar um valor de intensidade entre 0 e 255 (tom de cinzento dos pixéis).

As funções implementadas permitem-nos manipular as imagens que passamos como argumentos ao script. Assim é possível transformar a imagem na sua versão negativa (ImageNegative); aplicar threshold, ou seja, transformar todos os pixéis com um valor inferior ao threshold para preto e os restantes para branco (ImageThreshold); iluminar a imagem através da multiplicação dos pixéis por um fator, sem ultrapassar o nível máximo de cinzento (ImageBrighten); rodar a imagem em 90° no sentido contrário aos ponteiros do relógio (ImageRotate); criar uma versão espelhada da imagem, no sentido esquerda-direita (ImageMirror); recortar uma parte da imagem através das coordenadas do primeiro pixel, e do tamanho pretendido de comprimento e altura (ImageCrop); colar uma imagem por cima de outra (ImagePaste); misturar uma imagem com uma imagem de tamanho maior (ImageBlend); procurar uma imagem dentro de uma imagem maior através da comparação de pixéis (ImageLocateSubImage e ImageMatchSubImage) e aplicar blur a uma imagem através de um filtro de tamanho variável (ImageBlur).

Para concluir, foram realizados testes para verificar se as funções foram implementadas de forma correta, e testar a eficiência das mesmas.

ImageLocateSubImage

Dados experimentais

De seguida, apresentam-se as tabelas de resultados para a função ImageLocateSubImage. Esta função foi testada com 3 imagens diferentes, de dimensões 300x300, 512x512 e 940x940. Para cada imagem, foram utilizadas dimensões diferentes para a janela de procura (subimagem). Para efeitos de comparação, foram registados os valores de acesso à memória (pixmem), o número de comparações efetuadas (pcomp) e o número de iterações executadas (itr).

Também se fez a análise do melhor e do pior caso. O melhor caso é quando a janela de procura corresponde a só um pixel e este encontra-se logo no início da imagem (canto superior esquerdo). Para pior caso, considerou-se que a imagem e a janela de procura só diferenciavam num pixel, sendo este o último pixel da imagem (canto inferior direito). Também se verificou que quando a janela de procura era um quarto da imagem original, os valores de acesso à memória, comparação e iterações atingiam um valor máximo. Isto é possível observar nos resultados seguintes.

- Teste com a imagem de 300x300

```
# Teste da função ImageLocateSubImage
```

#	IMAGELOCATESUBIMAGE	BEST CASE	300x300 (size: 50)		
#	time	caltime	pixmem	pcomp	itr
	0.000013	0.000020	5000	2500	2500
#	IMAGELOCATESUBIMAGE	WORST CASE	300x300 (size: 50)		
#	time	caltime	pixmem	pcomp	itr
	0.591287	0.882714	315005000	157502500	157502500
#	IMAGELOCATESUBIMAGE	BEST CASE	300x300 (size: 150)		
#	time	caltime	pixmem	pcomp	itr
	0.000087	0.000131	45000	22500	22500
#	IMAGELOCATESUBIMAGE	WORST CASE	300x300 (size: 150)		
#	time	caltime	pixmem	pcomp	itr
	1.898982	2.834931	1026045000	513022500	513022500
#	IMAGELOCATESUBIMAGE	BEST CASE	300x300 (size: 250)		
#	time	caltime	pixmem	pcomp	itr
	0.000238	0.000356	125000	62500	62500
#	IMAGELOCATESUBIMAGE	WORST CASE	300x300 (size: 250)		
#	time	caltime	pixmem	pcomp	itr
	0.602741	0.899814	325125000	162562500	162562500

Figura 1- SubLocate com imagem de 300

- Teste com a imagem de 512x512

```
# Teste da função ImageLocateSubImage
```

#	IMAGELOCATESUBIMAGE	BEST CASE	512x512 (size: 150)		
#	time	caltime	pixmem	pcomp	itr
	0.000085	0.000133	45000	22500	22500
#	IMAGELOCATESUBIMAGE	WORST CASE	512x512 (size: 150)		
#	time	caltime	pixmem	pcomp	itr
	10.600842	16.556035	5929605000	2964802500	2964802500
#	IMAGELOCATESUBIMAGE	BEST CASE	512x512 (size: 250)		
#	time	caltime	pixmem	pcomp	itr
	0.000248	0.000388	125000	62500	62500
#	IMAGELOCATESUBIMAGE	WORST CASE	512x512 (size: 250)		
#	time	caltime	pixmem	pcomp	itr
	15.233094	23.790530	8646125000	4323062500	4323062500
#	IMAGELOCATESUBIMAGE	BEST CASE	512x512 (size: 350)		
#	time	caltime	pixmem	pcomp	itr
	0.000452	0.000707	245000	122500	122500
#	IMAGELOCATESUBIMAGE	WORST CASE	512x512 (size: 350)		
#	time	caltime	pixmem	pcomp	itr
	11.921441	18.618502	6509405000	3254702500	3254702500

Figura 2 - SubLocate com imagem de 512

- Teste com a imagem de 940x940

```
# Teste da função ImageLocateSubImage
```

#	IMAGELOCATESUBIMAGE	BEST CASE	940x940 (size: 350)		
#	time	caltime	pixmem	pcomp	itr
	0.000514	0.000802	245000	122500	122500
#	IMAGELOCATESUBIMAGE	WORST CASE	940x940 (size: 350)		
#	time	caltime	pixmem	pcomp	itr
	151.704801	236.600524	85573845000	42786922500	42786922500
#	IMAGELOCATESUBIMAGE	BEST CASE	940x940 (size: 450)		
#	time	caltime	pixmem	pcomp	itr
	0.000748	0.001166	405000	202500	202500
#	IMAGELOCATESUBIMAGE	WORST CASE	940x940 (size: 450)		
#	time	caltime	pixmem	pcomp	itr
	173.402573	270.440616	97637805000	48818902500	48818902500
#	IMAGELOCATESUBIMAGE	BEST CASE	940x940 (size: 550)		
#	time	caltime	pixmem	pcomp	itr
	0.001113	0.001737	605000	302500	302500
#	IMAGELOCATESUBIMAGE	WORST CASE	940x940 (size: 550)		
#	time	caltime	pixmem	pcomp	itr
	165.274019	257.763230	92493005000	46246502500	46246502500

Figura 3 - SubLocate com imagem de 940

Análise formal

Na figura seguinte encontra-se a análise formal da função ImageSubLocateImage, tal como o melhor e o pior caso, como foram descritos anteriormente. Assim a análise feita está de acordo com os resultados obtidos.

→ ImageLocateSubImage

w_1 → width da img 1
 w_2 → width da img 2
 h_1 → height da img 1
 h_2 → height da img 2

complexidade de ImageMatchSubImage, usada em ImageLocateSubImage

$$\sum_{i=0}^{w_1-w_2} \sum_{j=0}^{h_1-h_2} \sum_{i_2=0}^{w_2-1} \sum_{j_2=0}^{h_2-1} 1 = (w_1-w_2)(h_1-h_2)(w_2-1)(h_2-1) =$$
$$= (w_1 h_1 - w_1 h_2 - w_2 h_1 + w_2 h_2)(w_2 h_2 - w_2 - h_2 + 1)$$
$$= \underbrace{w_1 h_1 w_2 h_2}_{\text{Pascal + Significativa}} - w_1 h_1 w_2 + \dots + w_2 h_2$$

Seja $m_i = w_i h_i$, $i \in \{1, 2\}$ → m_i é o número de pixels da imagem i

$O(m_1 m_2)$

Best case:

A subImage é localizada no início (canto superior esquerdo).

$$B(m_1, m_2) = m_2 \rightarrow \text{são só comparados os pixels de img 2 com os da img 1}$$

Worst case:

A subImage é localizada só no final (canto inferior direito), e todos os pixels de img 2 são idênticos aos de img 1, exceto o último.

$$W(m_1, m_2) = (w_1 - w_2)(h_1 - h_2)(w_2 - 1)(h_2 - 1)$$

Figura 4 - Análise formal do SubLocate

ImageBlur

Dados experimentais

De seguida, apresentam-se as tabelas de resultados para a função ImageBlur antiga (primeira versão) e para a otimizada. Os testes foram realizados com 3 imagens de dimensões 300x300, 512x512 e 940x940, respetivamente. Também foram utilizados diferentes tamanhos para as janelas que aplicam o blur. Para efeitos de comparação, foram utilizadas principalmente as variáveis “pixmen” e “itr” referidas anteriormente, uma vez que a função otimizada não efetua comparações.

- Teste com a imagem de 300x300

```
# Teste da função Blur não otimizada
```

# BLUR_Old image (size: 90000 - window 7x7)				
#	time	caltime	pixmem	pcomp itr
	0.048359	0.077179	20019136	20250000 20250000
# BLUR_Old image (size: 90000 - window 50x50)				
#	time	caltime	pixmem	pcomp itr
	1.922274	3.067883	770332500	918090000 918090000
# BLUR_Old image (size: 90000 - window 100x100)				
#	time	caltime	pixmem	pcomp itr
	6.706785	10.703797	2520310000	3636090000 3636090000
# BLUR_Old image (size: 90000 - window 200x200)				
#	time	caltime	pixmem	pcomp itr
	20.146750	32.153518	6416280000	14472090000 14472090000

Figura 5 – Função Blur não otimizada

```
# Teste da função Blur otimizada
```

# BLUR_otimizado image (size: 90000 - window 7x7)				
#	time	caltime	pixmem	pcomp itr
	0.000889	0.001418	180000	0 180000
# BLUR_otimizado image (size: 90000 - window 50x50)				
#	time	caltime	pixmem	pcomp itr
	0.000884	0.001411	180000	0 180000
# BLUR_otimizado image (size: 90000 - window 100x100)				
#	time	caltime	pixmem	pcomp itr
	0.000864	0.001380	180000	0 180000
# BLUR_otimizado image (size: 90000 - window 200x200)				
#	time	caltime	pixmem	pcomp itr
	0.000806	0.001287	180000	0 180000

Figura 6 - Função Blur otimizada

- Teste com a imagem de 512x512

```
# Teste da função Blur não otimizada
```

# BLUR_Old image (size: 262144 - window 7x7)				
#	time	caltime	pixmem	pcomp itr
	0.129697	0.207268	58911808	58982400 58982400
# BLUR_Old image (size: 262144 - window 50x50)				
#	time	caltime	pixmem	pcomp itr
	5.954916	9.516482	2417688676	2674130944 2674130944
# BLUR_Old image (size: 262144 - window 100x100)				
#	time	caltime	pixmem	pcomp itr
	20.806677	33.250906	8614853776	10590879744 10590879744
# BLUR_Old image (size: 262144 - window 200x200)				
#	time	caltime	pixmem	pcomp itr
	71.729894	114.630699	27262758976	42153017344 42153017344

Figura 7 – Função Blur não otimizada

```
# Teste da função Blur otimizada
```

# BLUR_otimizado image (size: 262144 - window 7x7)				
#	time	caltime	pixmem	pcomp itr
	0.003050	0.004875	524288	0 524288
# BLUR_otimizado image (size: 262144 - window 50x50)				
#	time	caltime	pixmem	pcomp itr
	0.003302	0.005277	524288	0 524288
# BLUR_otimizado image (size: 262144 - window 100x100)				
#	time	caltime	pixmem	pcomp itr
	0.003343	0.005343	524288	0 524288
# BLUR_otimizado image (size: 262144 - window 200x200)				
#	time	caltime	pixmem	pcomp itr
	0.003176	0.005076	524288	0 524288

Figura 8 - Função Blur otimizada

- Teste com a imagem de 940x940

```
# Teste da função Blur não otimizada
```

# BLUR_Old image (size: 883600 - window 7x7)				
#	time	caltime	pixmem	pcomp itr
	0.472320	0.746783	199884736	198810000 198810000
# BLUR_Old image (size: 883600 - window 50x50)				
#	time	caltime	pixmem	pcomp itr
	19.822706	31.341604	8538562900	9013603600 9013603600
# BLUR_Old image (size: 883600 - window 100x100)				
#	time	caltime	pixmem	pcomp itr
	73.696887	116.521863	31986396400	35698323600 35698323600
# BLUR_Old image (size: 883600 - window 200x200)				
#	time	caltime	pixmem	pcomp itr
	280.964552	444.231965	113396478400	142083763600 142083763600

Figura 9 - Função Blur não otimizada

```
# Teste da função Blur otimizada
```

# BLUR_otimizado image (size: 883600 - window 7x7)				
#	time	caltime	pixmem	pcomp itr
	0.010977	0.017356	1767200	0 1767200
# BLUR_otimizado image (size: 883600 - window 50x50)				
#	time	caltime	pixmem	pcomp itr
	0.012011	0.018990	1767200	0 1767200
# BLUR_otimizado image (size: 883600 - window 100x100)				
#	time	caltime	pixmem	pcomp itr
	0.016686	0.026382	1767200	0 1767200
# BLUR_otimizado image (size: 883600 - window 200x200)				
#	time	caltime	pixmem	pcomp itr
	0.014224	0.022489	1767200	0 1767200

Figura 10 - Função Blur otimizada

Análise formal

Nas figuras seguintes encontram-se as análises formais da função ImageBlur otimizada e antiga (primeira versão). A grande diferença é que na versão otimizada, a função não depende da janela, tornando esta bem mais eficiente e consistente.

→ Image Blur (Otimizada)

$w \rightarrow$ width da img

$h \rightarrow$ height da img

$$\sum_{x=1}^w \sum_{y=1}^h 1 + \sum_{x_2=1}^w \sum_{y_2=1}^h 1 = wh + wh = 2 \underbrace{wh}_m = 2m$$

$O(m)$

Vemos claramente que esta versão de Image Blur é mais eficiente, pois a sua complexidade depende exclusivamente do número de pixels da img, e não da janela em que o Blur é aplicado.

Figura 11 – Análise formal do ImageBlur otimizada

Análise Formal

→ Image Blur (CME)

w → width da img

h → height da img

dx → quantidade de pixels de distância do pixel central no eixo dos xx

dy → " " " " " " " " " " " "

$$\sum_{x=1}^w \sum_{y=1}^h \sum_{i=x-d_x}^{x+d_x} \sum_{j=y-d_y}^{y+d_y} 1 = w h (2 d_x + 1) (2 d_y + 1) = (2 w h d_x + w h) (2 d_y + 1) = 4 w h d_x d_y + 2 w h d_x + 2 w h d_y + w h$$

Pouca + significativa

Considerando:

m = wh

$O(w h d_x d_y) \rightarrow O(m d_x d_y)$

Figura 12 - Análise formal do ImageBlur não otimizada

CONCLUSÃO

Em conclusão, a realização deste trabalho permitiu-nos melhorar as nossas capacidades de otimização de código, necessárias especialmente na realização da função “ImageBlur”.

Foi bastante interessante trabalhar com ficheiros do tipo “.pgm”, uma vez que nos permite manipular os valores de cinzento de todos os pixéis, de forma a alcançar resultados muito curiosos.

Este projeto também nos permitiu melhorar as nossas capacidades de trabalho em equipa, importantíssimas na nossa área.



BIBLIOGRAFIA

- <https://www.youtube.com/watch?v=4Eh0y3LHTNU&t=596s>
- <https://stackoverflow.com/questions/3437404/min-and-max-in-c>
- <https://stackoverflow.com/questions/2290509/debug-vs-ndebug>