

Licenciatura em Engenharia Informática

# Algoritmos e Estruturas de Dados

## O TAD image8bit



André Ribeiro  
Nº Mec.:112974

Daniel Ramos  
Nº Mec.:113170

26 de novembro de 2023

# **Índice**

Introdução.....	1
ImageLocateSubImage.....	1
Análise Experimental.....	1
Análise Formal.....	2
ImageBlur.....	3
Análise Formal.....	3
Primeiro Algoritmo – Lento.....	3
Segundo Algoritmo – Rápido.....	3
Comparação dos algoritmos.....	4
Análise Experimental.....	4
Conclusão.....	5

# **Índice de Figuras**

Figura 1: Gráfico de teste da aleatoriedade.....	2
Figura 2: Gráfico do pior caso da ImageLocateSubImage.....	3
Figura 3: Gráfico do melhor caso da ImageLocateSubImage.....	3
Figura 4: Comparação entre os dois algoritmos de Blur.....	5

# **Índice de Tabelas**

Tabela 1: Tabela do teste da aleatoriedade de ImageLocateSubImage.....	7
Tabela 2: Tabela de dados do Algoritmo Rápido de Blur.....	8
Tabela 3: Tabela de dados do Algoritmo Lento de Blur.....	8

# **Apêndices**

Apêndice I.....	7
-----------------	---

# Introdução

O presente trabalho tem como tema a análise de complexidade de algoritmos, mais concretamente algoritmos de procura de sub-imagens e de “Blur”.

São objetivos deste trabalho compreender a forma como os algoritmos se comportam computacionalmente e descobrir formas de os otimizar.

Está organizado em 2 partes principais. Na primeira parte debruçamo-nos no estudo da função `ImageLocateSubImage`. Na segunda parte debruçamo-nos no estudo das funções de “Blur”.

## ImageLocateSubImage

A utilização desta função permite encontrar uma imagem menor numa imagem maior. Podemos considerar a sub-imagem como um “recorte” da imagem maior.

O objetivo do algoritmo é encontrar este recorte numa imagem e saber as coordenadas do primeiro pixel da imagem. O processo de localização normalmente envolve comparar a sub-imagem com várias regiões da imagem maior. Ao encontrar uma correspondência, podemos garantir que o recorte é de facto uma sub-imagem da imagem maior.

Para analisar a complexidade do algoritmo definimos como objeto de estudo as comparações realizadas no decorrer da função.

## Análise Experimental

Criámos um ambiente de teste para estudar a evolução do objeto de estudo, usando uma imagem de 10000x10000 externa às imagens facultadas, recortada 300 vezes de forma aleatória (tanto a posição inicial como as dimensões são aleatórias).

Os dados utilizados no gráfico estão na tabela 1 do **Apêndice I**. Com estes dados obtivemos o seguinte resultado:

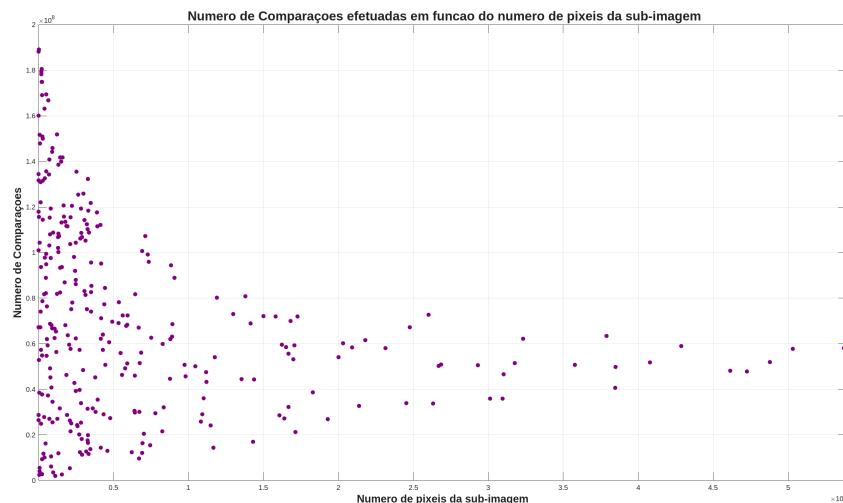


Figura 1: Gráfico de teste da aleatoriedade

Neste gráfico está representado o número de comparações necessárias para encontrar a sub-imagem da imagem de 10000x10000.

O resultado obtido é previsível. Numa situação aleatória é mais difícil encontrar imagens com dimensões menores do que imagens de maior dimensão, isto deve-se ao facto de existirem mais posições válidas para sub-imagens menores do que para maiores, no entanto, durante a análise dos pixels da sub-imagem é estatisticamente provável que o algoritmo encontre uma diferença nos primeiros pixels do que nos últimos pixels, independentemente do tamanho da sub-imagem.

Na situação da aleatoriedade, o tamanho da sub-imagem revela-se pouco importante e a complexidade do algoritmo passa a ser proporcional às posições possíveis.

## Análise Formal

Para além da análise experimental, para antever o pior e o melhor caso, realizámos a análise formal do algoritmo. Da análise do código obtivemos a seguinte expressão:

$$\sum_{i=0}^{w-w_2} \sum_{j=0}^{h-h_2} \left( 1 + \sum_{k=0}^{w_2-1} \sum_{l=0}^{h_2-1} 1 \right) = (h - h_2 + 1)(w - w_2 + 1)(h_2 w_2 + 1)$$

$h$  := altura

$w$  := comprimento

$h_2$  := sub-imagem altura

$w_2$  := sub-imagem comprimento

$h_2 * w_2$  simboliza o número de comparações necessárias para saber se a imagem menor é sub-imagem da original (número de pixels da imagem menor).

$(h-h_2 + 1)(w - w_2 + 1)$  representa todas as posições válidas para a imagem menor.

Após a formulação matemática, conseguimos identificar facilmente o melhor e o pior caso para este algoritmo.

No pior caso, o número de comparações efetuadas será igual à expressão calculada. Significa que todas as posições possíveis vão ser verificadas e a cada posição serão executadas  $h_2 * w_2$  comparações. Para  $h_2 = w_2$  e  $h = w$  a complexidade será  $O(n^4)$ .

No melhor caso a sub-imagem está na primeira posição (0,0) e são apenas efetuadas  $h_2 * w_2$  comparações, neste cenário a complexidade para  $h_2 = w_2$  será  $O(n^2)$ .

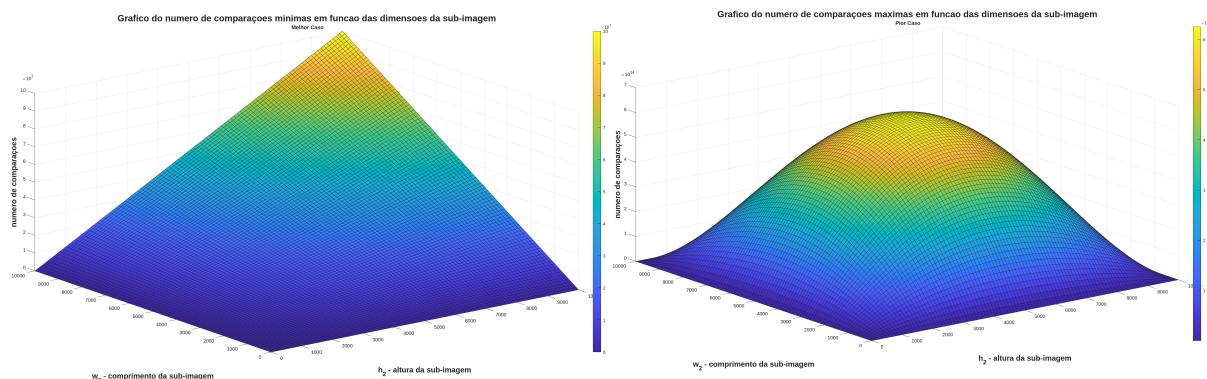


Figura 3: Gráfico do melhor caso da ImageLocateSubImage

Figura 2: Gráfico do pior caso da ImageLocateSubImage

## ImageBlur

De forma simples, o "Blur" (desfoque) em imagens refere-se a uma técnica que suaviza as transições entre pixels da imagem, reduzindo os detalhes e tornando-a menos nítida. Para isto é necessário misturar as cores dos pixels vizinhos. O resultado é uma imagem mais suave, onde as transições entre diferentes áreas são menos perceptíveis.

Existem diferentes métodos para aplicar o desfoque, sendo um deles, usar uma média dos valores dos pixels numa vizinhança retangular, ao redor de cada píxel da imagem, criando a sensação de suavidade.

Para a análise da complexidade dos algoritmos que implementam a função de "Blur", escolhemos como objeto de estudo o número de somas e subtrações realizadas ao longo da execução da função.

Durante o desenvolvimento do algoritmo de "Blur" surgiram duas abordagens diferentes.

## Análise Formal

### Primeiro Algoritmo – Lento

Na primeira implementamos diretamente o algoritmo "Blur" (mean filter), sem qualquer tipo de otimização.

Da análise do código obtivemos a seguinte formulação matemática:

$$\sum_{i=0}^{w-1} \sum_{j=0}^{h-1} \left( 1 + \sum_{k=i-\Delta x}^{i+\Delta x} \sum_{l=j-\Delta y}^{j+\Delta y} 2 \right) = 8hw\Delta x\Delta y + 4hw\Delta x + 4hw\Delta y + 3hw$$

$h :=$  altura

$w :=$  largura

$\Delta y :=$  metade da altura do retângulo da janela "Blur"

$\Delta x :=$  metade largura do retângulo da janela "Blur"

A soma total da largura do retângulo será  $(2\Delta x + 1)$ , e a soma total da altura será  $(2\Delta y + 1)$ .

Com a análise desta expressão concluímos que a complexidade do algoritmo é proporcional a  $h * w * \Delta x * \Delta y$  para valores muito grandes de  $h$ ,  $w$ ,  $\Delta x$  e  $\Delta y$  a complexidade do algoritmo aproxima-se de  $O(n^4)$ .

Com isto concluímos que o algoritmo tem um desempenho melhor para vizinhanças de "Blur" menores, o mesmo se aplica às dimensões da imagem, se fixarmos um valor de vizinhança, e variarmos as dimensões da imagem o desempenho mostra-se muito reduzido.

### Segundo Algoritmo – Rápido

Ao contrário do primeiro algoritmo, este demonstrou ser muito mais eficiente, através do uso de uma função auxiliar (sumTable).

sumTable é uma função que calcula a soma cumulativa do valor da cada retângulo possível numa imagem, assim, permite-nos realizar um "Blur" que não depende de  $\Delta x$  nem de  $\Delta y$ , mas apenas das dimensões da imagem. A nossa solução passa a utilizar a estratégia de programação dinâmica.

Na execução da função, a imagem original é percorrida píxel por píxel. Para cada píxel, a função calcula a média dos valores dos pixels numa vizinhança, sendo a largura e altura dessa região, respetivamente,  $\Delta x$  e  $\Delta y$ . Este cálculo da média é otimizado pela utilização da tabela de soma cumulativa para que com apenas 3 somas encontre o valor correto do "Blur" para cada píxel.

Com esta estratégia chegamos à seguinte fórmula matemática:

$$(h - 2) + (w + 2) + \sum_{i=1}^{w-1} \sum_{j=1}^{h-1} 3 + \sum_{k=0}^{w-1} \sum_{l=0}^{h-1} 4 = 7hw - 2h - 2w + 3$$

$h :=$  altura

$w :=$  largura

Após a interpretação da expressão matemática concluímos que a complexidade do algoritmo é proporcional a  $h * w$ , sendo a parte com mais 'peso' na expressão.

Para quando  $h = w$  a complexidade do algoritmo é proporcional a  $O(n^2)$ .

Esta relação é consistente para todos os casos, independentemente da configuração específica dos pixels.

Portanto, para este algoritmo, não há distinção clara entre melhor e pior caso. O desempenho é determinado principalmente pelo tamanho total da imagem. Se  $w$  e  $h$  forem grandes, o desempenho será afetado proporcionalmente.

Em resumo, a complexidade do algoritmo é consistente em todos os casos, e não há um caso que seja significativamente melhor ou pior do que os outros.

## Comparação dos algoritmos

Uma das maiores diferenças entre os dois algoritmos é o tempo de execução e o número de somas/subtrações. Sendo que o primeiro demora muito mais a produzir o mesmo resultado que o segundo, no entanto, após análise formal de ambos algoritmos tornou-se relativamente fácil entender esta grande diferença de execução. Como a diferença de complexidade é cerca de  $O(n^2)$  do número de operações, assim como também o tempo de execução é muito maior para o primeiro algoritmo comparado com o segundo.

## Análise Experimental

Para análise dos algoritmos começámos por definir, filtros de  $\Delta x \times \Delta y$  respetivamente, 3x3, 7x7, 9x9, 11x11, 15x15 e 17x17 e tamanho das imagens, 300x300, 512x512, 940x940, 1600x1200.

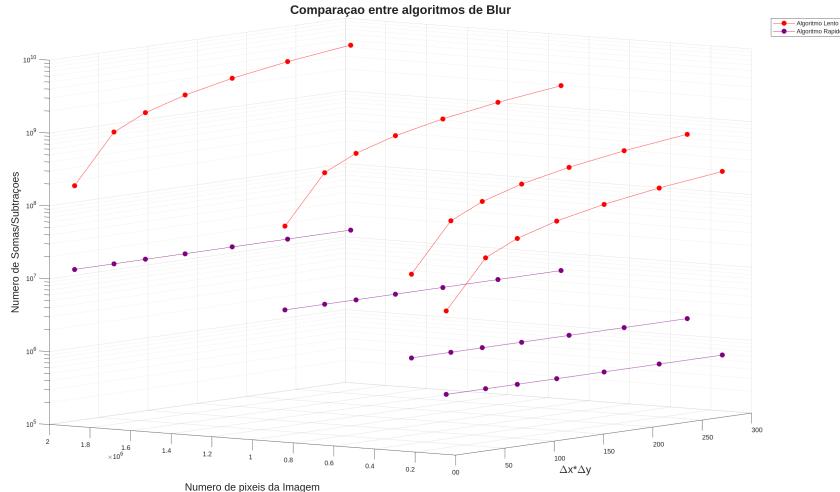


Figura 4: Comparação entre os dois algoritmos de Blur

Na tabela 2 e 3 do Apêndice I podemos encontrar os dados utilizados na Figura 4. Nesta Figura, para visualizar mais facilmente a quantidade de somas em cada algoritmo, o eixo das quotas tem escala logarítmica.

Através dos testes que fizemos, foi possível recolher dados para ilustrar graficamente a forma como o número de pixels se relaciona com o tamanho do filtro, assim como também, como o primeiro e segundo algoritmo se comparam.

A análise experimental complementa a análise formal.

Verificamos que no algoritmo rápido, se fixarmos o tamanho da imagem, e fizermos variar a janela de filtragem, a complexidade mantém-se constante.

Para além disso, também é trivial verificar que a complexidade só varia consoante a variação do tamanho da imagem e que se aproxima de um crescimento linear.

Relativamente ao algoritmo lento, quanto maior a janela de filtro e maior o tamanho da imagem, maior é o número de somas/subtrações realizadas.

É possível verificar a diferença brusca de operações entre os dois algoritmos, o lento tem ordem de grandeza entre  $10^8$  e  $10^{10}$ , enquanto o rápido tem ordem de grandeza entre  $10^5$  e  $10^7$ .

Por fim, podemos concluir que enquanto o segundo algoritmo é independente da janela de filtro, o primeiro não, conforme o tamanho da janela aumenta, aumenta também o número de somas.

## Conclusão

Com a realização deste trabalho, podemos concluir, que a análise da complexidade de algoritmos é extremamente útil, para de forma eficiente compreendermos o comportamento dos algoritmos computacionalmente e assim, entender a melhor forma de os otimizar.

Todos os objetivos deste trabalho foram concluídos.

## Apêndice I

Comps	pixels	37277124	649392	103770237	2119822	35439131	3955641	47559619	11184396
28771509	1410	166824478	652856	115482297	2144520	112108050	4132953	43335872	11201920
67238316	1450	134337623	697358	21523033	2147817	14475090	4157794	24232912	11484066
101027579	5796	140909581	738522	57753360	2155978	62133627	4161612	14486760	11668320
160094889	6266	103202823	739252	75155356	2192250	71245211	4183685	54040536	11774148
188119883	12928	27059851	742995	25086071	2199746	95164432	4191423	80206527	11893503
26466723	15438	115283344	743808	120458985	2241451	57378371	4293384	72999312	12983760
134482854	17184	108084421	768000	78046004	2272710	64107050	4309620	44536166	13549272
131750374	18135	49299913	781550	98152351	2382492	29066788	4345186	80769149	13807665
118005537	20195	68696691	788256	42839990	2404896	77304010	4398038	68939451	14154690
115684012	27531	45286824	803590	91965153	2452428	84510614	4446648	17050127	14323116
67175279	32238	97609314	813750	86193550	2482949	50734776	4465320	44333980	14376420
189164745	33368	119314131	829194	39247822	2487375	13070871	4603620	72114198	15005760
52842239	45976	6211333	846523	62310035	2490114	60677359	4715100	71924211	15820892
38499410	62080	10571219	848500	104314250	2494064	27429529	4784736	28623260	16074009
2473469	64878	68210105	872046	88064304	2503826	69667194	4934055	59598858	16230225
104337164	71688	40776537	882000	135464800	2533000	69127558	5348412	27162667	16399152
5485570	81004	66802551	913680	24290783	2578218	78213919	5366255	58595889	16513952
4053928	87669	144140482	927081	23805416	2617090	55996524	5490100	55705023	16671594
151628199	90241	34504656	930272	125444539	2649690	46375217	5583000	32255266	16686180
147922261	110605	25560907	932690	20174407	2734754	72414065	5617185	70003979	16810794
67236530	131692	145943893	950130	57380464	2738540	49174280	5782434	53264641	17000895
121992889	142956	3620633	988790	39795434	2758950	67875901	5861304	59282075	17073960
74131955	153400	108810652	997854	12389341	2766224	68245592	5916680	21260826	17136360
130972269	160930	62580462	1071894	106160614	2788940	51292516	5932176	71925466	17275896
93685073	164320	66702248	1079120	33903468	2829222	72450608	5956851	38658071	18307432
24915057	180681	2007752	1134648	119367393	2843856	12456798	6217750	26914877	19303636
57349435	185880	65354400	1163915	25452049	2849652	30572753	6417437	54053919	20007108
179314478	189210	56448851	1197188	108670274	2873680	29771386	6425552	60209949	20307741
178233597	202062	81830213	1231072	18236501	2895802	45968306	6427836	58400959	20914912
180612454	217467	151850589	1241315	11324540	2908311	81685915	6454728	32760226	21374689
174849071	220935	27012157	1255800	106853352	2910672	67066248	6681245	61530688	21794984
169095566	240530	102124874	1303362	48390271	2968329	9707122	6720376	58090495	23148640
174968189	245358	106783508	1311856	125829307	3012672	30199793	6727672	34030800	24526320
37722351	246064	100247309	1325728	83074100	3063792	51453397	6765266	67217854	24773049
2838503	250887	108370122	1331796	114212912	3078500	56046076	6841604	72695132	26014716
54836414	253506	138541373	1342656	81470088	3137230	12181868	6916864	33790709	26325776
9349226	253739	11909360	1345344	105338404	3152440	100666390	6923136	50333248	26699617
150994543	257868	107208526	1381003	12730523	3186201	16405405	6941396	50886433	26856486
78766425	262599	31721618	1425956	112454575	3233514	20531667	7029000	50532835	29302934
131481912	282144	82546189	1451946	75231161	3234518	107265013	7135044	35936258	30122424
150065099	283878	141744991	1457808	110245655	3270000	99185036	7286372	36017197	30950464
114383490	293506	93284778	1458945	31534428	3272670	96019788	7369596	46583340	31033440
11802877	348000	139939162	1515531	17664709	3282608	15546983	7444504	51505750	31772406
81825333	386540	113170135	1548807	16613400	3295552	62721237	7521207	62186658	32319225
27777728	392472	93730729	1561977	132267837	3306150	29592536	7802049	50793299	35775240
163114930	399672	2731475	1570204	19950710	3315372	21658813	8269085	63416237	37893219
10165551	400536	141695039	1614536	118321667	3329376	59921878	8282281	40693042	38472302
97718996	426405	120724915	1679265	11615267	3339630	32122534	8351136	49901576	38481542
132533091	441294	115840112	1706960	108807372	3375464	44717800	8774730	51858044	40773012
16205154	470956	87015842	1752066	13858925	3459320	62050068	8787616	58970746	42875756
89000648	495978	113438146	1805230	121715494	3486112	94393402	8846885	48110109	46150013
82151770	497777	68085925	1805777	74044814	3516264	63132070	8911140	47861954	47245550
99502992	512632	111726864	1870000	95682625	3520642	68637246	8946102	51979725	48789948
169470683	518580	46365375	1872585	82619406	3523520	88902084	9069273	57803977	50300328
135690076	529142	28769749	1921504	85383396	3537318	50788518	9752358	58105473	53725914
94920180	534240	111575176	1926578	31597911	3629700	45698632	9812340	55189517	53891860
54753126	546040	63770358	1957740	45297487	3801486	50071643	10472604		
61993140	559680	59608958	2061934	30183156	3823824	25820377	10832469		
76356641	575460	5358276	2102625	117651746	3901095	29037917	10932000		
59301659	608220	26271154	2110440	111560710	3930352	36070640	11030968		

Tabela 1: Tabela do teste da aleatoriedade de ImageLocateSubImage

Número de Somas/Subtrações para efetuar o Blur – Algoritmo Rápido				
$\Delta x * \Delta y$	Dimensões			
	300x300	512x512	940x940	1600x1200
3	628801	1832961	6181441	13434401
7	628801	1832961	6181441	13434401
9	628801	1832961	6181441	13434401
11	628801	1832961	6181441	13434401
13	628801	1832961	6181441	13434401
15	628801	1832961	6181441	13434401
17	628801	1832961	6181441	13434401

*Tabela 2: Tabela de dados do Algoritmo Rápido de Blur*

Número de Somas/Subtrações para efetuar o Blur – Algoritmo Lento				
$\Delta x * \Delta y$	Dimensões			
	300x300	512x512	940x940	1600x1200
3	8809488	25780512	87160848	189609888
7	39588272	116512896	395351472	861222272
9	63034200	186044232	632429400	1378600200
11	91701648	271427616	924351888	2016313248
13	125479448	372470472	1270762008	2773827848
15	164257200	488980992	1671303600	3650611200
17	207925272	620768136	2125621272	4646131272

*Tabela 3: Tabela de dados do Algoritmo Lento de Blur*