Eficácia, Eficiência e Escalabilidade em Método de Aprendizado Não Supervisionado de Busca de Imagens

Lucas Pascotti Valem, Daniel Carlos Guimarães Pedronette

[1] Departamento de Estatística, Matemática Aplicada e Computação (DEMAC)
Universidade Estadual Paulista (UNESP), Rio Claro, Brasil

lucasvalem@rc.unesp.br, daniel@rc.unesp.br

Resumo. *Vários métodos de aprendizado não supervisionado têm sido propostos obtendo melhorias significativas na eficácia de sistemas de busca de imagens. No entanto, apesar do considerável ganho de eficácia, esses métodos geralmente requerem altos custos computacionais, não contemplando adequadamente requisitos de eficiência e escalabilidade. Esse trabalho de iniciação científica propôs um método de aprendizado não supervisionado que, apesar de ganhos significativos de eficácia, também considera requisitos de eficiência e escalabilidade. Conceitos de computação paralela e heterogênea, utilizando CPUs e GPUs, foram aplicados no desenvolvimento do trabalho.*

Abstract. *Various unsupervised learning methods have been proposed obtaining significant improvements in the effectiveness of image search systems. However, despite the relevant effectiveness gains, these approaches commonly require high computation efforts, not addressing properly efficiency and scalability requirements. In this scientific initiation work, we present a novel unsupervised learning approach which achieves significant effectiveness gains considering both efficiency and scalability requirements. Parallel and heterogeneous computing, using CPU and GPU devices, were exploited.*

1. Introdução

Atualmente, o grande avanço tecnológico em relação à aquisição, armazenamento e compartilhamento de imagens tem contribuído para o aumento de coleções de imagens em larga escala. Com o rápido crescimento na quantidade de imagens digitais disponíveis, torna-se premente o desenvolvimento de sistemas eficazes e eficientes para indexação e organização do conteúdo visual. Nesse cenário desafiador, uma das abordagens mais promissoras consiste no uso de Sistemas de Recuperação de Imagens Baseados no Conteúdo (*Content-Based Image Retrieval - CBIR*). Os sistemas *CBIR* podem ser genericamente definidos como um conjunto de técnicas cujo objetivo reside em organizar e buscar imagens por meio da análise de seu conteúdo visual [Datta et al. 2008].

Uma aplicação comum dos sistemas *CBIR* está em, dada uma coleação de imagens e uma imagem de consulta, retornar uma lista ordenada de resultados (*ranked list*) contendo as imagens mais similares no topo. A similaridade entre as imagens é avaliada de acordo com propriedades visuais, como forma, cor e textura. Tal similaridade entre as imagens é calculada utilizando-se um descritor de imagens, caracterizado por duas funções: a primeira extrai e codifica as características da imagem em um vetor, e a segunda define uma medida de similaridade usada para comparar dois vetores.

Os Sistemas *CBIR* evoluiram, por décadas, apoiados pelo desenvolvimento de novos descritores, explorando novas características visuais e medidas de distância. No entanto, este modelo se mostrou limitado ao relacionar características de baixo nível a conceitos de alto nível. Mais recentemente, visando resolver essa questão, novas etapas de recuperação não diretamente relacionadas a aspectos baixo nível foram propostas [Liu et al. 2007]. Nesse cenário, foram propostos métodos de aprendizado não supervisionado capazes de calcular uma medida de distância mais eficaz entre as imagens, sem a necessidade de intervenção do usuário [Yang et al. 2009]. O principal objetivo desses métodos é substituir o cálculo de distâncias que considerem apenas pares de imagens por medidas mais globais, capazes de analisar as coleções de forma geral, assim como o relacionamento entre as imagens [Yang et al. 2013].

Diversos métodos de aprendizado não supervisionado propostos recentemente apresentam ganhos significativos na qualidade de busca de imagens. Todavia, em geral, tais métodos são avaliados considerando apenas a eficácia, ignorando requisitos de eficiência e escalabilidade, também imprescindíveis em aplicações reais. Além da qualidade dos resultados recuperados, considerar o tempo gasto para se obter os resultados em coleções cada vez maiores é indispensável.

Este trabalho apresenta um método de aprendizado não supervisionado que considera de forma conjunta aspectos de eficácia, eficiência e escalabilidade. O algoritmo *RL-Recommendation* [Valem et al. 2015] explora as informações contidas nos *ranked lists* através de recomendações não supervisionadas entre as imagens. Enquanto os ganhos de eficácia são comparáveis a outros métodos, o algoritmo requer baixo custo computacional utilizando apenas as primeiras posições dos *ranked lists*. Tal abordagem permite utilizar técnicas de indexação, tornando o método apropriado para coleções de imagens de diferentes tamanhos. Além disso, uma solução paralela para o algoritmo utilizando computação heterogênea (CPUs e GPUs) também foi proposta.

Diversos experimentos foram realizados com objetivo de avaliar a eficácia, eficiência e a escalabilidade do método proposto. Foram considerados cinco *datasets* públicos de diferentes tamanhos e vários descritores de imagens. Os resultados experimentais não só revelam que o método proposto é capaz de produzir significativas melhorias em várias tarefas de recuperação de imagens, mas também que o ganho pode ser obtido em coleções de imagens de diferentes tamanhos e em reduzidos tempos de execução. O algoritmo proposto também foi avaliado em comparação a outros métodos recentemente propostos.

O restante do texto é organizado como se segue: a Seção 2 apresenta a definição do problema e a Seção 3 descreve o algoritmo *RL-Recommendation* proposto. A Seção 4 apresenta os resultados experimentais e a Seção 5 as publicações relacionadas. Por fim, a Seção 6 discute as conclusões.

2. Definição Formal do Problema

Esta seção tem como objetivo apresentar uma definição formal para o modelo de busca de imagens e aprendizado não supervisionado, utilizado ao longo do texto. Seja $\mathcal{C}$=$\{img_1, img_2, \ldots, img_n\}$ uma coleção de imagens, onde n é o tamanho da coleção $\mathcal{C}$. Seja $\rho(i, j)$ uma função que define a distância entre duas imagens de acordo com seus correspondentes vetores de características. A distância $\rho(i, j)$ entre todas as imagens img_i, $img_j \in \mathcal{C}$

podem ser calculadas para se obter uma matriz de distância A, tal que $A_{ij} = \rho(i, j)$. A matriz de distância A é usada como entrada para vários algoritmos de aprendizado não supervisionado, mas frequentemente causa dificuldades de escalabilidade para grande coleções de imagens, levando a uma complexidade de armazenamento e tempo de pelo menos $O(n^2)$.

Uma representação alternativa para resultados de recuperação é baseada nos *ranked lists*. Baseada na função de distância ρ, um *ranked list* τ_q pode ser calculado em resposta a uma imagem de consulta img_q. Os *ranked lists* podem conter informações de uma coleção inteira, mas especialmente em suas primeiras posições espera-se localizar as imagens mais relevantes relacionadas a imagem de consulta. Portanto, uma estratégia adequada para acelerar o processo de busca consiste em considerar um subconjunto das L imagens mais similares, onde $L \ll n$ é o número de imagens nas primeiras posições do *ranked list*. Essa é uma estratégia útil especialmente para grandes coleções de imagens, onde n é muito grande, e portanto τ_q é muito custoso de se calcular.

O *ranked list* τ_q=$(img_1, img_2, \ldots, img_L)$ pode ser formalmente definido como a permutação de uma coleção de imagens $\mathcal{C}_s \subset \mathcal{C}$, a qual contem as imagens mais similares a uma dada imagem de consulta img_q, tal que $|\mathcal{C}_s| = L$. Uma permutação τ_q é uma bijeção do conjunto $\mathcal{C}_s$ no conjunto $[L] = \{1, 2, \ldots, L\}$. Para uma permutação τ_q, $\tau_q(i)$ é interpretado como a posição da imagem img_i no *ranked list* τ_q. Pode-se dizer que, se img_i é classificado antes da img_j no *ranked list* da img_q, isto é, $\tau_q(i) < \tau_q(j)$, então $\rho(q, i) \leq \rho(q, j)$.

Tomando toda imagem $img_i \in \mathcal{C}$ como uma imagem de consulta img_q, pode-se obter um conjunto $\mathcal{R} = \{\tau_1, \tau_2, \ldots, \tau_n\}$ de *ranked lists* para cada imagem da coleção $\mathcal{C}$. O objetivo deste trabalho é propor um algoritmo para redefinir o conjunto de *ranked lists* $\mathcal{R}$ produzindo um conjunto mais efetivo $\mathcal{R}_e$. Portanto, o algoritmo de aprendizado não supervisionado pode ser definido como uma função f_u, tal que: $\mathcal{R}_e = f_u(\mathcal{R})$.

O objetivo da função f_u é explorar o conjunto $\mathcal{R}$ realizando recomendações não supervisionadas baseadas nas informações contidas nas posições iniciais dos *ranked lists*.

3. Algoritmo *RL-Recommendation*

O algoritmo *RL-Recommendation* [Valem et al. 2015] é baseado em recomendações não supervisionadas realizadas entre as imagens. Métodos de recomendação, originalmente criados para seleção automática de itens com base em preferências pessoais, são simulados por um algoritmo de maneira não supervisionada. As recomendações são realizadas baseadas nas informações contidas em *ranked lists*, nos quais as imagens nas primeiras posições são recomendadas entre si. Neste cenário, a recomendação significa que a distância entre duas imagens deve diminuir, e portanto, devem ser movidos para as primeiras posições nos *ranked lists* correspondentes.

O método apresentado é inspirado pelo algoritmo *Pairwise Recommendation* [Pedronette and Torres 2012]. No entanto, a proposta difere em aspectos importantes. Enquanto o *Pairwise Recommendation* requer a distância entre todas as imagens de um *dataset* como dado de entrada, o algoritmo *RL-Recommendation* requer só as primeiras posições dos *ranked lists*, os quais podem ser obtidos utilizando de estruturas de indexação. A partir dos *ranked lists*, uma matriz de distâncias esparsa é calculada. *RL-Recommendation* dispensa o uso de *clustering* e requer um número pequeno de iterações

para sua convergência. Como resultado, o algoritmo apresenta custo computacional muito menor.

O algoritmo pode ser descrito considerando quatro etapas principais:

1. **Cálculo de Matriz de Distância Esparsa:** a entrada do algoritmo consiste em um conjunto de *ranked lists* $\mathcal{R}$, o qual assegura propriedades de escalabilidade ao algoritmo. No entanto, medidas de distância são necessárias para se realizar recomendações não supervisionadas. As distâncias entre as imagens formariam uma matriz quadrada completa, mas como o algoritmo considera apenas as imagens nas primeiras posições dos *ranked lists*, tem-se uma matriz esparsa A.
2. **Cálculo da Coesão:** a coesão [Pedronette and Torres 2012] é uma medida que provê uma estimativa não supervisionada da eficácia de um determinado *ranked list*. Sendo assim, pode-se aplicar o princípio de que um *ranked list* de alta coesão possui mais autoridade para realizar recomendações. A medida também é utilizada como critério de convergência: as recomendações são realizadas enquanto a coesão dos *ranked lists* continuar crescendo.
3. **Recomendações Não Supervisionadas:** as primeiras posições dos *ranked lists* constituem a região que contém a informação com maior acurácia provida pelo descritor de imagem. Esta informação é explorada para criar um perfil para as imagens nas k primeiras posições, onde se realizará as recomendações não supervisionadas. Se duas imagens estão contidas nesse perfil, isso indica uma similaridade, produzindo uma recomendação que reduz a distância entre elas.
4. **Ordenação dos *Ranked Lists*:** as recomendações alteram as distâncias entre as imagens. Portanto, os *ranked lists* precisam ser atualizados para refletir as novas posições. Um procedimento de ordenação é realizado para atualizar os *ranked lists* de acordo com as novas distâncias calculadas.

As etapas de 2 a 4 são repetidas enquanto a média da coesão dos *ranked lists* continua a aumentar acima de um dado limiar. As próximas seções descrevem as principais contribuições do algoritmo e apresentam uma solução de paralelização. Mais detalhes podem ser encontrados em [Valem et al. 2015].

3.1. Cálculo de Matriz de Distância Esparsa

Embora a entrada do algoritmo seja definida como conjunto de *ranked lists* $\mathcal{R}$, as recomendações não supervisionadas requerem medidas de distância. Dessa forma, a distância é estimada com base nas informações fornecidas pelos *ranked lists*. Apenas as distâncias entre as imagens nas primeiras L posições dos *ranked lists* foram consideradas, gerando uma matriz de distância em sua maior parte esparsa A (com aproximadamente $n \times L$ valores utilizados), o que aumenta a escalabilidade do algoritmo.

A distância entre duas imagens é calculada baseada na soma recíproca referente ao seus *ranked lists*. Formalmente, dada duas imagens, img_q e img_i, a distância $\rho(q, i)$ é definida como $\rho(q, i) = \tau_q(i) + \tau_i(q)$. De acordo com esta formulação, a matriz de distância A (tal que $A_{qi} = \rho(q, i)$) pode ser facilmente calculada processando todos os *ranked lists* pela soma das recíprocas posições. No entanto, as referências não são simétricas e os *ranked lists* não contêm todas as imagens (só as primeiras L posições). Portanto, pode-se observar situações onde a img_i está no *ranked list* da img_q mas o inverso não ocorre ($img_i \in \tau_q$ mas $img_q \notin \tau_i$).

Uma solução foi proposta com o objetivo de manter a complexidade em $O(n \times L)$. Inicialmente, todos os pares de imagem que apresentam referências nos *ranked lists* têm suas distâncias definidas como um valor máximo igual a $2 \times L$. Em uma segunda etapa, cada referência contida em um *ranked list* produz um decremento de L e um incremento no valor de posição ($\tau_q(i)$) correta. Dessa forma, pares de imagens que se referem reciprocamente nas primeiras L posições, tem os valores iniciais substituídos pelas suas respectivas posições. Em pares de imagens que apresentam referências não recíprocas, parte da distância continua com o valor máximo L. Mais detalhes sobre o algoritmo proposto podem ser encontrados em [Valem et al. 2015].

3.2. Paralelização utilizando OpenCL

Esta seção discute o *design* paralelo do algoritmo *RL-Recommendation*. Foi adotado o padrão OpenCL, aberto e multiplataforma, que define uma *API* para paralelização e computação heterogênea. O código OpenCL é executado em um *device* podendo ser uma CPU, GPU ou algum outro tipo de acelerador. O OpenCL suporta modelos distintos de programação como o *data-parallel*, o *task-parallel* e outros modelos híbridos.

A Figura 1 ilustra o design do *RL-Recommendation* Paralelizado usando seis *kernels*. Um *kernel* pode ser entendido como uma etapa da paralelização, cada instância de um *kernel* é executada em uma unidade chamada *work-item*. O mesmo código é executado em paralelo por diferentes *work-items* com dados distintos.

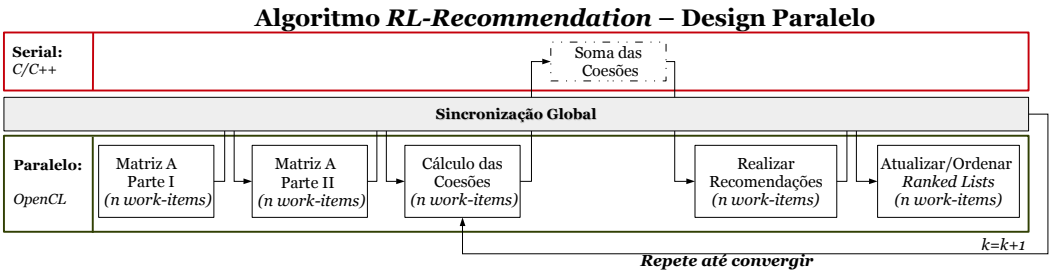

Figura 1. Design do *RL-Recommendation* Paralelo [Valem et al. 2015].

Os dois primeiros *kernels* realizam o preenchimento da matriz de distância, como discutido na Seção 3.1. São utilizados dois *kernels* para garantir sincronização (todos os *work-items* da "*Parte I*" devem terminar antes que se inicie a "*Parte II*"). O terceiro *kernel* calcula as medidas de coesão para todos os *ranked lists*, de forma que cada *work-item* processe a coesão para um *ranked list*. Uma média da coesão é calculada serialmente e utilizada para verificar o critério de convergência.

As recomendações não supervisionadas são realizadas pelo quarto *kernel*. Também utiliza-se n *work-items*, cada um processando um *ranked list* distinto. Embora as recomendações processadas sejam independentes, diferentes *ranked lists* podem atualizar posições concorrentes da matriz de distância A provocando perda de algumas atualizações. Uma possível solução é assegurar acesso exclusivo, mas o custo associado com a sincronização granulada no OpenCL é significativa. Portanto, como sugerido por outros trabalhos [Pedronette et al. 2012], permite-se atualizações diretas a matriz A devido ao pequeno impacto na eficácia do algoritmo (como demonstrado pelos intervalos de confiança na Seção 4).

O último *kernel* executa um procedimento de ordenação para atualizar os *ranked lists*. Como a maior parte das mudanças ocorrem apenas no começo dos *ranked lists* [Pedronette et al. 2012], usa-se o algoritmo de *Insertion Sort*, cuja complexidade tende a ser linear quando a entrada está quase ordenada.

Foram considerados também dois modelos de transferência de memória definidos pelo padrão OpenCL: *Write Buffer* e *Map Buffer*. O modelo *Write Buffer* requer que as transferências de memória sejam explicitamente transferidas para a memória do dispositivo em questão. Por outro lado, no caso do *Map Buffer* o modelo consiste apenas na realização da cópia do ponteiro dos dados na memória. Ambas estratégias foram avaliadas nos nossos experimentos, conforme resultados apresentados na Seção 4.

4. Avaliação Experimental

Esta seção apresenta um resumo dos experimentos realizados para avaliações de eficiência, eficácia e escalabilidade do método proposto. Para uma descrição completa, consulte [Valem et al. 2015].

4.1. *Datasets*, Descritores e *Baselines*

Vários experimentos foram realizados considerando 5 *datasets* distintos com tamanho variando entre 280 e 72.000 imagens. Também foram utilizados 18 descritores locais e globais, considerando características como contorno, cor e textura. A Tabela 1 apresenta um resumo dos *datasets* e descritores utilizados. Na análise de eficácia, todas as imagens de cada *dataset* são consideradas como imagens de consulta. O MAP (*Mean Average Precision*) foi usado como medida de eficácia para a maior parte dos *datasets*, exceto para a coleção N-S. Para o *dataset* MPEG-7, também considera-se o *Recall@40*. No caso do *dataset* ALOI, usou-se a estrutura de indexação BP-Tree [Almeida et al. 2010] para processar os *ranked lists*.

Tabela 1. *Datasets* e descritores usados na avaliação experimental.

Soccer [van de Weijer and Schmid 2006]: 280 cenas, compostas por imagens de 7 times de futebol com 40 imagens por classe. Avaliação de eficácia utilizando MAP (%).
Descritores: *Border/Interior Pixel Classification (BIC) [Stehling et al. 2002], Auto Color Correlograms (ACC) [Huang et al. 1997], and Global Color Histogram (GCH) [Swain and Ballard 1991]*;
MPEG-7 [Latecki et al. 2000]: 1.400 imagens com diferentes formas divididas em 70 classes. Frequentemente utilizada para avaliação de métodos de aprendizado não supervisionado. Avaliação de eficácia utilizando MAP (%) e Recall@40.
Descritores: *Segment Saliences (SS) [Torres and Falcão 2007], Beam Angle Statistics (BAS) [Arica and Vural 2003], Inner Distance Shape Context (IDSC) [Ling and Jacobs 2007], Contour Features Descriptor (CFD) [Pedronette and Torres 2010], Aspect Shape Context (ASC) [Ling et al. 2010], and Articulation-Invariant Representation (AIR) [Gopalan et al. 2010].*
Brodatz [Brodatz 1966]: 1.776 imagens, composta por 111 texturas diferentes divididas em 16 classes. Avaliação de eficácia utilizando MAP (%).
Descritores: *Local Binary Patterns (LBP) [Ojala et al. 2002], Color Co-Occurrence Matrix (CCOM) [Kovalev and Volmer 1998], Local Activity Spectrum (LAS) [Tao and Dickinson 2000]*; MAP (%)
N-S [Nistér and Stewénius 2006]: 10.200 imagens, composta de 2.550 objetos capturados de 4 diferentes maneiras (ponto de vista, distância e condições de iluminação). Avaliação de eficácia utilizando N-S score.
Descritores: *ACC [Huang et al. 1997], BIC [Stehling et al. 2002], Color and Edge Directivity Descriptor (CEED) [Chatzichristofis and Boutalis 2008a], Fuzzy Color and Texture Histogram (FCTH) [Chatzichristofis and Boutalis 2008b], Joint Composite Descriptor (JCD) [Zagoris et al. 2010], Scale-Invariant Feature Transform (SIFT) [Lowe 1999].*
ALOI [Geusebroek et al. 2005]: 72.000 imagens de 1000 classes de objetos com diferentes pontos de vista, oclusão e condições de iluminação. Avaliação de eficácia utilizando MAP (%).
Descritores: *ACC [Huang et al. 1997], BIC [Stehling et al. 2002], GCH [Swain and Ballard 1991], Color Coherence Vectors (CCV) [Pass et al. 1996], Local Color Histograms (LCH) [Lu et al. 1994].*

Os experimentos para avaliação de eficiência consideraram a média de 10 execuções e intervalos de confiança de 95%. O hardware onde os experimentos fo-

ram realizados é composto de uma *CPU Intel Xeon CPU E3-1240* e uma *GPU AMD Radeon HD 7900 Series*. O ambiente de software é dado pelo sistema operacional *Linux 3.11.0-15 - Ubuntu 12.04* com *OpenCL 1.2 AMD-APP*. Os algoritmos *Parwise-Recommendation* [Pedronette and Torres 2012] e *RL-Sim* [Pedronette and Torres 2013, Pedronette et al. 2013] foram considerados como *baselines* em vários experimentos.

4.2. Avaliação de Eficácia

Esta seção apresenta a avaliação de eficácia do algoritmo proposto. A Tabela 2 apresenta os resultados de MAP para três *datasets*. Foram reportados os valores de MAP das implementações seriais e paralelas GPU do algoritmo *RL-Recommendation*. Note que os intervalos de confiança são muito pequenos, indicando uma pequena variação entre diferentes execuções. O ganho relativo é calculado baseado na execução serial. Ganhos muito significativos são observados para a maior partes dos descritores, variando de +5.92% a +29.22%. Para propósitos de comparação, também reportou-se o MAP do algoritmo *Parwise-Recommendation*. Os resultados de eficácia do *RL-Recommendation* são superiores para a maioria dos descritores.

Tabela 2. Avaliação de eficácia do algoritmo *RL-Recommendation*.

Descritor	*Dataset*	MAP Original	*Pairwise Recom.*	*RL-Recom.* Serial	*RL-Recom.* Paralelo (GPU)	Ganho
SS [Torres and Falcão 2007]	MPEG-7	37.67%	39.90%	48.68%	48.64% ± 0.0062	+29.22%
BAS [Arica and Vural 2003]	MPEG-7	71.52%	77.65%	79.58%	79.57% ± 0.0047	+11.27%
IDSC [Ling and Jacobs 2007]	MPEG-7	81.70%	86.83%	88.80%	88.78% ± 0.0067	+11.86%
CFD [Pedronette and Torres 2010]	MPEG-7	80.71%	91.38%	91.39%	91.37% ± 0.0055	+13.23%
ASC [Ling et al. 2010]	MPEG-7	85.28%	91.80%	91.34%	91.32% ± 0.0050	+7.11%
AIR [Gopalan et al. 2010]	MPEG-7	89.39%	95.50%	96.12%	96.12% ± 0.0071	+7.53%
GCH [Swain and Ballard 1991]	Soccer	32.24%	32.35%	34.38%	34.44% ± 0.0340	+6.64%
ACC [Huang et al. 1997]	Soccer	37.23%	40.31%	41.23%	41.20% ± 0.0239	+10.74%
BIC [Stehling et al. 2002]	Soccer	39.26%	42.64%	45.15%	45.17% ± 0.0693	+15.00%
LBP [Ojala et al. 2002]	Brodatz	48.40%	51.92%	51.26%	51.24% ± 0.0047	+5.91%
CCOM [Kovalev and Volmer 1998]	Brodatz	57.57%	66.46%	64.34%	64.32% ± 0.0059	+11.76%
LAS [Tao and Dickinson 2000]	Brodatz	75.15%	80.73%	79.71%	79.71% ± 0.0031	+6.07%

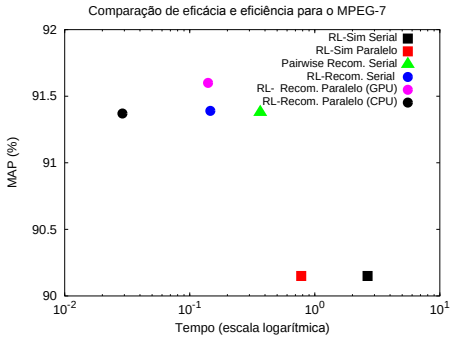

Figura 2. Eficácia e Eficiência.

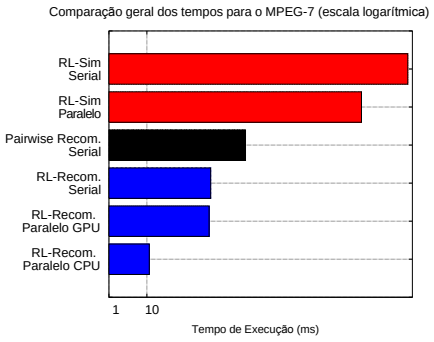

Figura 3. Avaliação de Eficiência.

A Figura 2 apresenta uma análise conjunta de eficácia e eficiência. O tempo de execução e o MAP são responsáveis pela posição dos algoritmos no gráfico. Portanto, um algoritmo ideal, com alta eficácia e baixo tempo de execução, é posicionado próximo ao canto superior esquerdo do gráfico. Note que o algoritmo *RL-Recommendation* ocupa as melhores posições.

4.3. Avaliação de Eficiência

Foram realizados experimentos para avaliar eficiência, considerando implementações serial e OpenCL, diferentes dispositivos (CPU, GPU) e modelos de transferência de

memória. A Figura 3 apresenta uma comparação dos tempos de execução do *RL-Recommendation* (serial e paralelo) em relação ao *Pairwise Recommendation* (serial) e o *RL-Sim* (serial e paralelo) considerando-se o *dataset* MPEG-7. Podemos observar que, apesar do uso da escala logarítmica, o tempo de execução do *RL-Recommendation* (em azul) é significativamente menor que os demais.

Tabela 3. Avaliação de eficiência do algoritmo *RL-Recommendation*.

Algoritmo	Exec.	Dispositivo	Soccer	MPEG-7	Brodatz	N-S
Pairwise Recom.	Serial	CPU	0.1149 ± 0.00018	0.3663 ± 0.00094	0.6672 ± 0.00140	14.802 ± 0.11059
RL-Recommendation	Serial	CPU	0.0607 ± 0.00000	0.1462 ± 0.00021	0.1108 ± 0.00102	0.1868 ± 0.00018
RL-Recommendation	Paralelo	GPU[1]	0.1380 ± 0.00642	0.1401 ± 0.00250	0.1004 ± 0.00412	$\mathbf{0.0582 \pm 0.00633}$
RL-Recommendation	Paralelo	GPU[2]	0.1538 ± 0.01056	0.2438 ± 0.00371	0.2376 ± 0.00326	0.3754 ± 0.00604
RL-Recommendation	Paralelo	CPU[1]	0.0131 ± 0.00100	0.0319 ± 0.00043	0.0299 ± 0.00129	0.1166 ± 0.00085
RL-Recommendation	Paralelo	CPU[2]	$\mathbf{0.0128 \pm 0.00104}$	$\mathbf{0.0290 \pm 0.00075}$	$\mathbf{0.0284 \pm 0.00114}$	0.1149 ± 0.00055

Modelo de Transferência de Memória: [1]*Write Buffer*; [2]*Map Buffer*.

A Tabela 3 apresenta o tempo de execução (médias e intervalos de confiança) para o algoritmo *RL-Recommendation* considerando diferentes *datasets*, *devices* e modelos de transferência de memória. Pode-se observar que os resultados de performance do *RL-Recommendation* são muito superiores aos do algoritmo *Pairwise Recommendation*. Considerando apenas as execuções seriais do *dataset* N-S, o *RL-Recommendation* é $79.3\times$ mais rápido. As implementações paralelas demonstram resultados ainda mais significativos. O tempo de execução para o *dataset* N-S (com 10.200 imagens) é de apenas $0.0582s$ para a execução em paralelo utilizando a GPU.

4.4. Avaliação de Escalabilidade

Foi realizado um experimento visando analisar o comportamento do algoritmo quando se varia o tamanho dos *ranked lists* de entrada (constante L). O *dataset* ALOI e o descritor LCH [Lu et al. 1994] foram considerados neste experimento. Variou-se o valor de L de 70 a 7000, reportando o tempo médio do aprendizado não supervisionado de distâncias por *ranked list*. Foram reportados os mesmos resultados para o algoritmo RL-Sim [Pedronette et al. 2014], como *baseline*. A Figura 4 ilustra os resultados. Pode-se observar uma média de tempo muito baixa para o algoritmo *RL-Recommendation*, mesmo para crescentes valores de L.

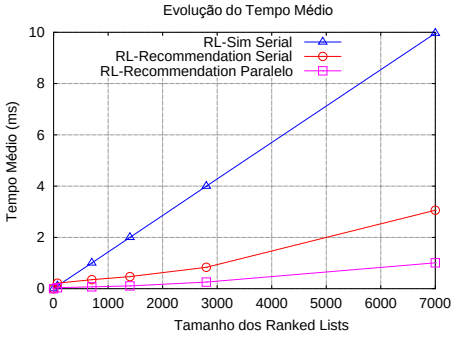

Figura 4. Análise de Escalabilidade.

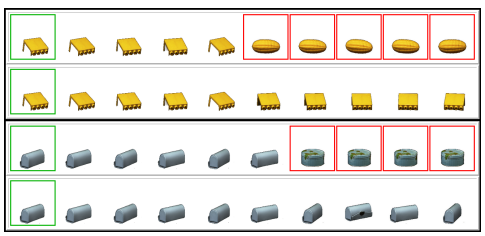

Figura 5. Exemplos de resultados.

4.5. Resultados Visuais

Alguns exemplos visuais de resultados são apresentados na Figura 5. A imagem apresentada em quadro verde é a imagem de consulta. Os *ranked lists* obtidos são apresentados

a direita, com imagens incorretas em quadros vermelhos. São apresentados dois exemplos de consultas, com os respectivos resultados antes e depois da aplicação do algoritmo, considerando o *dataset* ALOI [Geusebroek et al. 2005].

5. Publicações

Os resultados do trabalho de iniciação científica apresentados nesse documento deram origem a um artigo científico aceito para publicação em conferência internacional. O artigo [Valem et al. 2015] foi aceito na categoria *Full Paper* na *ACM International Conference on Multimedia Retrieval (ICMR) 2015* - Qualis A2.

6. Conclusão

Neste trabalho, é apresentada uma proposta de um algoritmo de aprendizado não supervisionado para tarefas de recuperação de imagens. A principal motivação consiste em explorar as informações contidas nos *ranked lists* para aumentar a eficácia de tarefas em sistemas *CBIR*. O método difere dos trabalhos anteriores, pois considera ao mesmo tempo aspectos de eficácia, eficiência e escalabilidade. Também foram explorados conceitos de computação paralela e heterogênea.

Referências

Almeida, J., da S. Torres, R., and Leite, N. J. (2010). BP-tree: An efficient index for similarity search in high-dimensional metric spaces. In *CIKM*, pages 1365–1368.

Arica, N. and Vural, F. T. Y. (2003). BAS: a perceptual shape descriptor based on the beam angle statistics. *Pattern Recognition Letters*, 24(9-10):1627–1639.

Brodatz, P. (1966). *Textures: A Photographic Album for Artists and Designers*. Dover.

Chatzichristofis, S. A. and Boutalis, Y. S. (2008a). CEDD: color and edge directivity descriptor: a compact descriptor for image indexing and retrieval. In *ICVS*, pages 312–322.

Chatzichristofis, S. A. and Boutalis, Y. S. (2008b). Fcth: Fuzzy color and texture histogram - a low level feature for accurate image retrieval. In *WIAMIS*, pages 191–196.

Datta, R., Joshi, D., Li, J., and Wang, J. Z. (2008). Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2):5:1–5:60.

Geusebroek, J.-M., Burghouts, G. J., and Smeulders, A. W. M. (2005). The amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112.

Gopalan, R., Turaga, P., and Chellappa, R. (2010). Articulation-invariant representation of non-planar shapes. In *ECCV*, volume 3, pages 286–299.

Huang, J., Kumar, S. R., Mitra, M., Zhu, W.-J., and Zabih, R. (1997). Image indexing using color correlograms. In *CVPR*, pages 762–768.

Kovalev, V. and Volmer, S. (1998). Color co-occurence descriptors for querying-by-example. In *ICMM*, page 32.

Latecki, L. J., Lakmper, R., and Eckhardt, U. (2000). Shape descriptors for non-rigid shapes with a single closed contour. In *CVPR*, pages 424–429.

Ling, H. and Jacobs, D. W. (2007). Shape classification using the inner-distance. *PAMI*, 29(2):286–299.

Ling, H., Yang, X., and Latecki, L. J. (2010). Balancing deformability and discriminability for shape matching. In *ECCV*, volume 3, pages 411–424.

Liu, Y., Zhang, D., Lu, G., and Ma, W.-Y. (2007). A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1):262 – 282.

Lowe, D. (1999). Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157.

Lu, H., Ooi, B., and Tan, K. (1994). Efficient image retrieval by color contents. In *ADB*, pages 95–108.

Nistér, D. and Stewénius, H. (2006). Scalable recognition with a vocabulary tree. In *CVPR*, volume 2, pages 2161–2168.

Ojala, T., Pietikäinen, M., and Mäenpää, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *PAMI*, 24(7):971–987.

Pass, G., Zabih, R., and Miller, J. (1996). Comparing images using color coherence vectors. In *ACM-MM*, pages 65–73.

Pedronette, D. C. G. and Torres, R. d. S. (2010). Shape retrieval using contour features and distance optmization. In *VISAPP*, volume 1, pages 197 – 202.

Pedronette, D. C. G. and Torres, R. d. S. (2012). Exploiting pairwise recommendation and clustering strategies for image re-ranking. *Information Sciences*, 207:19–34.

Pedronette, D. C. G. and Torres, R. d. S. (2013). Image re-ranking and rank aggregation based on similarity of ranked lists. *Pattern Recognition*, 46(8):2350–2360.

Pedronette, D. C. G., Torres, R. d. S., Borin, E., and Breternitz, M. (2012). Efficient image re-ranking computation on GPUs. In *ISPA*.

Pedronette, D. C. G., Torres, R. d. S., Borin, E., and Breternitz, M. (2013). RL-Sim algorithm acceleration on GPUs. In *SBAC*.

Pedronette, D. C. G. a., Almeida, J., and Torres, R. D. S. (2014). A scalable re-ranking method for content-based image retrieval. *Information Sciences*, 265:91–104.

Stehling, R. O., Nascimento, M. A., and Falcão, A. X. (2002). A compact and efficient image retrieval approach based on border/interior pixel classification. In *CIKM*, pages 102–109.

Swain, M. J. and Ballard, D. H. (1991). Color indexing. *International Journal on Computer Vision*, 7(1):11–32.

Tao, B. and Dickinson, B. W. (2000). Texture recognition and image retrieval using gradient indexing. *JVCIR*, 11(3):327–342.

Torres, R. d. S. and Falcão, A. X. (2007). Contour Salience Descriptors for Effective Image Retrieval and Analysis. *Image and Vision Computing*, 25(1):3–13.

Valem, L. P., Pedronette, D. C. G., Torres, R. d. S., Borin, E., and Almeida, J. (2015). Effective, efficient, and scalable unsupervised distance learning in image retrieval tasks. *ICMR*.

van de Weijer, J. and Schmid, C. (2006). Coloring local feature extraction. In *ECCV*, pages 334–348.

Yang, X., Koknar-Tezel, S., and Latecki, L. J. (2009). Locally constrained diffusion process on locally densified distance spaces with applications to shape retrieval. In *CVPR*, pages 357–364.

Yang, X., Prasad, L., and Latecki, L. (2013). Affinity learning with diffusion on tensor product graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence,*, 35(1):28–38.

Zagoris, K., Chatzichristofis, S., Papamarkos, N., and Boutalis, Y. (2010). Automatic image annotation and retrieval using the joint composite descriptor. In *PCI*, pages 143–147.