K-023

領域分割に基づく類似画像検索の runlength 符号化による高速化

Fast Similarity Calculation using Runlength for Regions-based Image Retrieval

金城 賀真† 小早川 倫広† 星 守† 大森 匡†

1. はじめに

山本らは画像の内容に基づく類似画像検索の手法の 1 つとして，画像を構成する領域の重なりを算出することによる，構図による類似画像検索を提案している [1][2](検索例を図 5 に示す)．領域分割された各領域にラベルを付加した画像 (以下，領域画像とよぶ) に対し，領域の重なり度合いを測定する尺度として，集合間の類似度である **Jaccard** 係数をベースとした新しい類似度 (重み付き Jaccard 係数)

$$Sim(I_i, I_j) = \sum_{R_{ix} \in I_i} \sum_{R_{iy} \in I_j} \frac{|R_{ix} \cap R_{jy}|^2}{|R_{ix}| \times |R_{jy}|} \times \frac{|R_{ix} \cap R_{jy}|}{|R_{ix} \cup R_{iy}|} \quad (1)$$

を提案し，類似画像検索を行っている．ただし，R_{ix}, R_{jy} は領域画像 I_i, I_j を構成する領域であり，$|R|$ は領域 R に属する画素数である．

しかし，重み付き Jaccard 係数に基づく類似度の計算には，画素が画像 I_i，I_j のどの領域に属しているかを 1 対 1 に比較しなければならないため，計算量は画素数に比例する．また現状では，1 つの問い合わせに対して，データベース内全ての領域画像と類似度を計算しているため，データベース内の領域画像数に比例して検索時間が増大していく．このため，重み付き Jaccard 係数に基づく類似度を用いた類似画像検索の問題点として，検索に掛る時間が大きいということが挙げられる．例えば，デスクトップ PC(OS:Linux，CPU:Athlon64 1.8GHz，Memory:1GB) 環境下で，160×106 ピクセルの領域画像 30,000 枚で構築されたデータベース中すべての領域画像と問い合わせ画像との類似度計算に，約 90 秒必要となる．

そのため，重み付き Jaccard 係数に基づいた類似画像検索システムを実用的に運用するためには，検索の高速化が必要となる．

一般的に，検索の高速化には，
(1) 類似度計算時間の削減
(2) 類似度計算回数の削減
の 2 つが考えられる．(1) は類似度計算自体の高速化であり，(2) は索引構造などを用いて検索対象候補を減らし，1 回の問い合わせに行う類似度計算の回数を削減し，検索を高速化する手法である．

本稿では，類似度計算時間の削減に焦点を当てる．基本的なアイデアは，領域画像のデータ構造において，画素単位で領域情報を持つのではなく，領域の連結性に着目し，ある程度画素をまとめて領域情報を持つことで，式 (1) の類似度を高速に計算するというものである．具体的には，領域画像のデータ構造に runlength 符号化を適用することで，類似度計算時間を削減する．

以降，2 節では，runlength 符号化適用方法について述べ，3 節で類似度計算時間，領域画像のデータサイズの比較の結果を示す．4 節では，領域画像に runlength 符号化を適用した場合の runlength 符号の性質の分析，評価を行い，5 節では runlength 符号化した場合の類似度計算時間の簡単なモデルでの比較を述べ，6 節でまとめを述べる．

†電気通信大学大学院情報システム学研究科

2. 類似度計算の高速化手法

式 (1) を計算するためには，$|R_{ix}|$，$|R_{jy}|$，$|R_{ix} \cap R_{jy}|$，$|R_{ix} \cup R_{jy}|$ を算出する必要がある．$|R_{ix}|$，$|R_{jy}|$ の値は領域画像自体から独立に計算可能であり，各領域からあらかじめ取得することができる．また，$|R_{ix} \cup R_{jy}|$ は $|R_{ix}|$，$|R_{jy}|$，$|R_{ix} \cap R_{jy}|$ から計算可能である．そこで，式 (1) を高速に計算するためには，$|R_{ix} \cap R_{jy}|$ を高速に計算することが必須となる．

本節では，runlength 符号化を適用した類似度高速計算手法を提案する．

2.1 runlength 符号化適用方法

領域画像の表現に対する runlength 符号化の適用方法を述べる．領域画像に対し，左から右，上から下にラスタスキャンを実行し，得られるラベルの 1 次元系列に対し，runlength 符号化を適用する．

ただし，一般的な runlength 符号は (ラベル，同じラベルが連続する個数) の組の列で表現されるが，ここでは，計算の簡単化のため，(ラベル,run の最終の位置) の組の列で表現する．例えば，ラベル"1"が付加された領域とラベル"2"が付加された領域からなるラベル系列 $I = \{1,1,1,2,2,1,1,1,1\}$ が与えられたとき，$I' = \{(1,3),(2,5),(1,9)\}$ と記述する．

2.2 類似度の計算手順

領域数 X の画像 I_i，領域数 Y の画像 I_j 間の類似度計算の手順を示す．ただし，I_i を構成している領域を $R_{ix}(x = 1,\ldots,X)$，I_j を構成している領域を $R_{jy}(y = 1,\ldots,Y)$ とする．I_i, I_j の runlength 表現を $I'_i = (L_{i1}, f_{i1}),\ldots,(L_{iN}, f_{iN})$，$I'_j = (L_{j1}, f_{j1}),\ldots,(L_{jM}, f_{jM})$ とする．ただし，N，M は run の個数である．runlength 表現を用いた $|R_{ix} \cap R_{jy}|$ の計算は，次の 4 ステップからなる．

step1 $X \times Y$ の行列 Inter を用意し，行列 Inter の各成分を"0"とする．

step2 I'_i の第 n 番目の run(L_{in}, f_{in}) と I'_j の第 m 番目の run(L_{jm}, f_{jm}) の共通区間の長さ $l = |[f_{in-1}+1, f_{in}] \cap [f_{jm-1}+1, f_{jm}]|$ を求める．

step3 行列 Inter の (L_{in}, L_{jm}) 成分に共通区間の長さ l を加える．

step4 step2 と step3 を共通区間が存在しなくなるまで実行し，終了する．

Procedure Intersection(I'_i, I'_j)
$I'_i = (L_{i1}, f_{i1}), \ldots, (L_{iN}, f_{iN})$
$I'_j = (L_{j1}, f_{j1}), \ldots, (L_{jM}, f_{jM})$

出力
Inter:$X \times Y$ の行列 (Inter[x][y]=$|R_{ix} \cap R_{jy}|$)

手順
$n = 1$; $m = 1$;
while ($n! = N \cap m! = M$)
  I'_i の第 n 番目の run(L_{in}, f_{in}) と I'_j の第 m 番目の run(L_{jm}, f_{jm}) の共通区間の長さ $l = |[f_{in-1} + 1, f_{in}] \cap [f_{jm-1} + 1, f_{jm}]|$ を求める.
  Inter[L_{in}][L_{jm}]+ = l;
  if ($f_{in} \leq f_{jm}$) $n++$; **else** $m++$;
end while

図 1: 領域の積集合を計算するための手続き

実際の手順を，図 1 に示す．$|R_{ix} \cap R_{jy}|$ を (x, y) 成分とする行列 Inter より得られた $|R_{ix} \cap R_{jy}|$ と $|R_{ix}|$, $|R_{jy}|$ より $|R_{ix} \cup R_{jy}|$ を算出し，式 (1) より類似度を計算できる．

2.3 計算の手間

runlength 符号化を用いた場合の領域画像 I_i，I_j 間の類似度の計算量を考える．I_i，I_j 間の類似度計算において，$|R_{ix} \cap R_{jy}|$ の計算がもっとも手間のかかる計算であることから，$|R_{ix} \cap R_{jy}|$ の計算について考える．

図 1 から，行列 Inter の (L_{in}, L_{jm}) 成分に共通区間の長さ l を加える操作は，I'_i の run の個数を N，I'_j の run の個数を M とすると，$(N + M)$ 回となる．すなわち計算量は $O(N + M)$ となる．つまり，run の個数が小さくなると計算時間は小さくなり，run の個数が大きくなると，計算時間も大きくなることを示している．一般的に $N + M$ は画素数に比べて十分小さく，step2 の負荷が大きくなければ，この手法により類似度計算の高速化が可能であることを示している．

3. 実験

runlength 符号化を適用した場合としない場合について，類似度計算時間，データサイズについて比較実験を行った．データベースとして，160 × 106 ピクセルの領域画像 30,000 枚を用意した．

3.1 類似度計算時間の比較

runlength 符号化を適用した場合としない場合について，類似度計算時間の比較のための実験を行った．

[実験手順]
実験にはデスクトップ PC を用いる．データベース内の画像を I_i(i=1, ..., 30000)) としたとき，以下の手順で実験を行った．

1. $i = 1, ..., 30000$ に対して，I_i とデータベース内の全ての領域画像との類似度計算時間の総和を計測

表 1: 1 回の問い合わせに要する類似度計算時間の総和の比較 (30,000 枚)(単位:sec)

	平均	分散	最大	最小
runlength を非適用 (p)	93.06	0.04	93.57	92.64
runlength を適用 (r)	16.96	22.16	39.57	12.34
p/r	5.49	0.0018	2.36	7.57

2. 1 回の問い合わせに必要な類似度計算時間の総和の平均値，分散値，最大値，最小値を算出

実験結果を表 1 に示す．

[考察]
表 1 より，runlength 符号化を適用した場合は適用しない場合と比較して類似度計算時間が，平均して約 1/5 に短縮されたことが分かる．

また，保持する run の個数が最大 (1450) である領域画像が問い合わせのときに類似度計算時間は最大となり，保持する run の個数が最小 (27) である領域画像が問い合わせのときに類似度計算時間は最小となった．このことより，runlength 符号化を適用した場合の類似度計算時間は run の個数に依存することが確認された．

また，類似度計算時間が最大のときでも runlengh 符号化を適用しない場合の約 1/2 の短縮になるので，runlength 符号化は有効な手段であると言える．

3.2 データサイズの比較

runlength 符号化を適用した場合と適用しない場合の 1 枚当たりの領域画像のデータサイズの比較を行う．runlength 符号化を適用した場合，領域のラベルを 1byte，run の最終の位置を 2byte で記述し，計 3byte でそれぞれの run を記述した．runlength 符号化を適用しない場合は各画素に，属する領域のラベルを 1byte で割り当てた．データベース内の領域画像に対し，runlength 符号化を適用した場合としない場合の 1 枚当たりのデータサイズは表 2 のようになった．データサイズは平均して約 1/14 に減少した．また，最悪の場合でも約 1/4 に収まる．

runlength 符号化を適用して生成された領域画像のデータを索引として考えると，データサイズが小さくなるに従い，類似画像検索システム実装時において，メモリ上で保持できる索引の数が増加する．メモリ上の索引が増加すると，ディスクアクセスの回数が減少するので，検索時間が削減できる．また，ディスク上の索引の読み込みに掛かる時間もデータサイズに比例して小さくなる．そのため，データサイズが小さいということは，I/O 時間が削減できるので，高速な検索システム実装の重要な点であると言える．データサイズの観点から考えても runlength 符号化は有効な手法だと言える．

4. runlength 符号の分析

領域画像に runlength 符号化を適用した場合の runlength 符号の性質を見るため，run の切れ目の位置，長さ，個数についての頻度分布を調べた．

表 2: 1 枚当たりの領域画像のデータサイズの比較 (単位:byte)

	平均	偏差	最大	最小
runlength を非適用	16960	0	16960	16960
runlength を適用	1219	596	4350	78

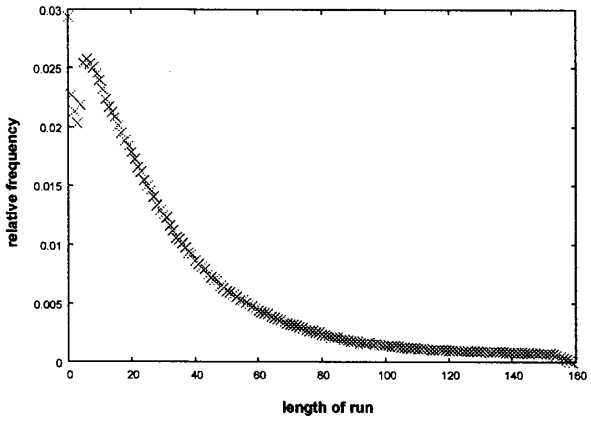

図 2: run の長さの相対頻度分布

4.1 run の切れ目の位置の分布

データベース内全ての領域画像から全ての run の切れ目 (画像上での領域分割点) の位置を取り出したときの run の切れ目の横軸の座標の相対頻度分布を図 3 に示す. 図 3 から，画像の両端 (横軸の座標が 0 及び 160 付近) 以外では，run の切れ目が同程度の頻度で発生していることがわかる. ただし，画像の両端では，相対頻度の値は小さい. これは，一般的に中央にオブジェクトを配置して写真を撮ることが多く，領域分割をした場合に画像の中央付近に分割線が発生しやすいなどの理由が考えられる.

このことから，画像の両端以外では，run の切れ目が同確率で独立に発生するという推測を立てることができる.

4.2 run の長さの分布

データベース内全ての領域画像から全ての run を取り出すしたときの run の長さの相対頻度分布を図 2 に示す. 平均値は 43.6 であった.

図 2 から run の長さの頻度は幾何分布に従うという推測ができる.

4.3 理論的分析

切れ目の位置，run の長さの分布について，理論的に議論する. ある x 座標上で run の切れ目が発生する確率を考えると，図 3 の頻度がほぼ一様になる x 座標上では平均して 0.024 であった.

ここで，run の切れ目が確率 0.024 で独立で発生するという仮定を立てる.

ある run の長さが k だったとき，$k-1$ 回連続 run の切れ目が発生せず，k 回目で run の切れ目が発生したと

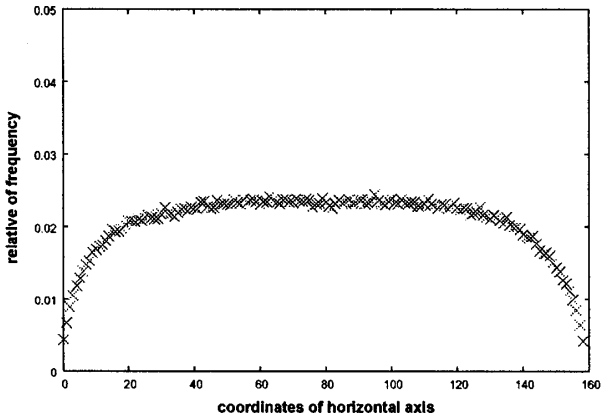

図 3: run の切れ目の位置の相対頻度分布

捉えると，run の長さの確率変数 $Lenth$ は幾何分布

$$P(Lenth = k) = (1-p)^{k-1}p$$

に従うと考えることができる. 仮定より，run の切れ目が発生する確率 p を 0.024 だとすると，run の長さの期待値 $E(Lenth)$ は

$$E(Lenth) = 1/p = 1/0.024 = 41.7$$

となる. この値は実際の run の長さの平均値 43.6 と近い値となっている.

また，領域画像において，1 行ごとの run の個数の相対頻度分布は図 4 のようになり，平均値は 4.83 であった. 1 行ごとの run の個数をある区間の中で run の切れ目が発生する回数+1 だと捉えると，1 行ごとの run の個数の確率変数 Num はポアソン分布を用いて，

$$P(Num = k+1) = e^{-\lambda}\frac{\lambda^(k)}{(k)!}$$

に従うと考えることができる. ポアソン分布において，λ の値は確率変数の期待値 $E(Num)$ に等しい. また，$E(Num)$ は区間の長さ n と run の切れ目が発生する確率 p の積に 1 だけ加えた数に等しい. ここでは 106×160 ピクセルの領域画像の 1 行を 1 区間としているので，$n = 160$ である. よって，

$$\lambda = np + 1 = 160 \times 0.024 + 1 = 4.81$$

となる.

この値は，実際の 1 行の run の個数の平均値 4.83 に非常に近い値となっている. 分散の値は差が大きいが，これは，図 3 における，領域画像の両端付近では run の切れ目の頻度が小さくなることが影響していると考えられる. しかし，第 1 次のモーメントである λ の値が実際の平均値とほぼ一致する. したがって，行ごとの run の個数の頻度分布がポアソン分布に従うということがある程度成り立つ.

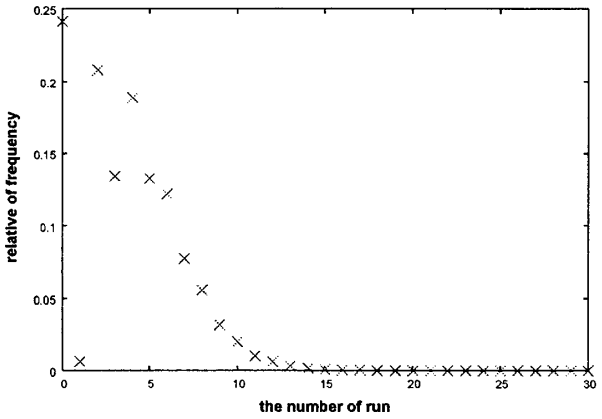

図 4: 1 行ごとの run の個数の相対頻度分布

5. コストモデル

類似度計算時間について，runlength 符号化によってどの程度類似度計算時間が削減できるか簡単なモデルで議論する．

類似度の計算量は，$|R_{ix} \cap R_{jy}|$ の計算に必要なコストに依存し，他の部分はほとんど無視できる．また，$|R_{ix} \cap R_{jy}|$ を求めたあとの計算手順は runlength 符号化を適用した場合としない場合で同じである．そこで，どれだけ $|R_{ix} \cap R_{jy}|$ の計算時間が削減できるかを議論する．runlength 符号化を適用しない場合とした場合それぞれの $|R_{ix} \cap R_{jy}|$ の計算コストモデルを式 (2),(3) とする．

$$F_p = \alpha_p \times N_p \tag{2}$$
$$F_r = \alpha_r \times N_r \tag{3}$$

ただし，α_p, α_r は 1 回のループに要する手間であり，N_p, N_r はループ回数である．

ここで，F_p/F_r を考える．N_p は画素数であり，N_r は最悪で I_i, I_j の run の個数の和が上限である．run の個数は領域画像によって異なるので，ここでは，データベース内の領域画像 1 枚あたりの run の個数の平均値 (=389) を使用する．よって，本稿で用いたのデータセットでは，

$$N_p/N_r \sim (16960)/(389 \times 2) = 21.8$$

となり，

$$F_p/F_r \sim 21.8 \times \alpha_p/\alpha_r$$

となる．

一方，α_p/α_r は，様々な要素が関係するので正確に値を算出することが難しい．ここで，簡単な指標として，runlength 符号化を適用しない場合とした場合とで 1 回のループにどれぐらいの時間がかかるかをシミュレーションし，その実測値の比を α_p/α_r として利用する．シミュレーションとして，ラベル L_1(値は $1, \ldots, N$ のどれか)，L_2(値は $1, \ldots, M$ のどれか) をランダムに生成し，2 次元配列の (L_1, L_2) 成分に 1 加える操作を 16,960 回実行するために要する時間 T_p と，ランダムに run を作ったファイルを 2 つ用意し，run の共通区間 l を求め，2 次元配列の (L_1, L_2) 成分に l を加える操作を 16,960 回繰り返し実行するために必要な時間 T_r を測定した．このとき，T_p/T_r は 0.274 であった．よって

$$\alpha_p/\alpha_r \simeq T_p/T_r \simeq 0.274$$

となる．よって F_p/F_r は大まかな値として，

$$F_p/F_r \sim 5.97 \tag{4}$$

となる．この値は類似度計算時間の比 $93.06/16.96 = 5.49$ と近い値となっている．

6. まとめ

本稿では，領域画像の表現に runlength 符号化を適用することで，領域分割に基づく類似画像検索を高速化できることを示した．領域画像 30,000 枚をデータセットとして runlength 符号化を適用する場合としない場合での類似度計算時間，データサイズの比較を行った．類似度計算時間の比較について，runlength 符号化を適用した場合，しない場合と比べて類似度計算時間は約 1/5 になることを示した．また，データサイズは約 1/14 になった．類似度計算時間，データサイズの点より，領域画像の表現に runlength 符号化を適用することの有用性を示した．

また，領域画像を runlength 表現したときの，run の切れ目の位置，長さの分布を理論的に分析した．類似度計算時間について runlength 符号化を適用する場合としない場合で簡単なモデルを立て比較した．モデルを用いた類似度計算の比 5.97 は類似度計算の実測値の比 5.46 と近い値となった．

参考文献

[1] Atsushi Yamamoto, Michihiro Kobayakawa, Mamoru Hoshi, and Tadashi Ohmori. Similarity measures for image retrieval based on segmented regions. *Proc. IWAIT2007*, pp. 529–534, 2007.

[2] 山本敦, 小早川倫広, 星守, 大森匡. 画像の領域分割に基づく類似画像検索. 情報処理学会 研究報告, Vol. 2006, No. 21, pp. 19–26, 2006.

A 領域分割に基づく類似画像検索の例

(a) 問い合わせ画像

(b) 検索結果

図 5: 問い合わせ画像と検索結果