改良四分木符号の論理操作について

6B−3

横川完治

日立ソフトウェアエンジニアリング(株)

1 はじめに

改良四分木符号とは、四分木符号 (quadtree) にブロック符号の概念を導入して複雑な2値画像の処理を効果的にしたものである。この符号が圧縮率や符号化速度について優れている点は既に報告されている [1]。ここでは、2値画像のビット操作の適用を最小限にし、四分木の特徴を生かした処理をなるべく多くするようにして、論理操作の高速化を計ることを述べる。

2 改良四分木符号

2^nx2^n の2値画像 $\{a_{ij}\}$ を考える。画像処理の最小の単位として、ブロックを導入する。ブロックの大きさは 2^{l_0}x2^{l_0} で2値画像は 2^{n-l_0}x2^{n-l_0} 個のブロックからなる。ブロックの色をブロックに属す画素が全て1の時 black、全て0の時 white、それ以外の時 gray であると定義する。四分木の l レベル (l = 0, 1, ..., n - l_0) の quadrant の数は最大 2^lx2^l で、quadrant の大きさは 2^{n-l}x2^{n-l} である。quadrant の色は、quadrant 内のブロックの色が一様なら、その色、一様でないなら、hetero と定義する。各 quadrant q は四分木のレベル l とそのレベルにおける quadrant の左上角の点の座標 x,y と色 c と、特に色が gray または hetero である時は、その quadrant 内の2値画像の部分配列で決まる。

改良四分木符号 {q} の生成アルゴリズムは、まず root から始めて、各 quadrant の内部を調べ、色が black または gray ならその quadrant の情報を出力し、white なら無視し、hetero ならその quadrant を4分割して、各 subquadran について同じ操作を行なう。

各画素 a_{ij} には一意的に整数 k を次の式で対応させることができる [2]。

$$i = i_{n-1}i_{n-2}...i_0$$

$$j = j_{n-1}j_{n-2}...j_0$$

の時、

$$k = i_{n-1}j_{n-1}i_{n-2}j_{n-2}...i_0j_0$$

したがって、各 quadrant についてもレベル l と座標 (x, y) から画素の座標 (i, j) を対応させて k を計算できる。改良四分木符号 {q} はこの整数 k についてソートされているものとする。生成の際、quadrant を調べる順序を適当に取れば、改良四分木符号はこの仮定を満たす。

3 AND 操作

改良四分木符号の AND 操作について述べる。入力は2つの改良四分木符号で、出力は1つの改良四分木符号であり、ここではいずれもファイルである。入力の先頭とは、後に述べるスタックが空でない時は、スタックのトップ、空の時は、ファイルの現在の位置をさす。

まず、各入力の先頭の quadrant を取り出し、その位置関係を調べて、次の3通りの場合に分類する。

場合1： 一致する、

場合2： 一方が他方を含む、

場合3： 重なりがない。

場合1については、2つの quadrant の色の組合せでさらに次の3通りの場合に分かれる。

1. 2つとも black ... 1つの quadrant を出力する、

Logical Operations on Images Using Improved Quadtrees
Kanji YOKOKAWA
Hitachi Software Engineering Co., Ltd.

2. black と gray ... gray の quadrant を出力する、

3. 2つとも gray ... 2つの quadrant の部分配列を AND 操作をして生成された部分配列を持つ quadrant を出力する。

次に2つの入力の先頭の quadrant を取り出す。

場合2については、quadrant の色が2つとも black であるならば、小さい方の quadrant を出力し、この入力の先頭の quadrant を取り出す。gray と black であるならば、大きい方の quadrant を4分割して、スタックに積み、この入力の先頭の quadrant を取り出す。

場合3については、前にある、すなわち整数 k の小さい quadrant を捨て、この入力の先頭の quadrant を取り出す。

各 quadrant の更新が可能なかぎり、以上の処理を繰り返す。

4 OR 操作

次に改良四分木符号の OR 操作について説明する。AND 操作と同様に、各入力の先頭の quadrant を取り出し、その位置関係で3通りに分類する。

場合1については、2つの quadrant の色の組合せで、さらに次の2通りの場合に分かれる。

1. 少なくとも1つが black ... black の quadrant を出力する、

2. 2つとも gray ... 2つの部分配列を OR 操作して生成された部分配列を持つ quadrant を出力する。

次に2つの入力の先頭の quadrant を取り出す。

場合2については、大きい方の quadrant の色が black であるならば、小さい方の quadrant を捨て、この入力の先頭の quadrant を取り出す。gray であるならば、大きい方の quadrant を4分割して、スタックに積み、この入力の先頭の quadrant を取り出す。

場合3については、前にある quadrant を捨て、この入力の先頭の quadrant を取り出す。

各 quadrant の更新が可能な限り、以上の処理を繰り返す。一方の入力が尽きた時、他方の入力の残りを出力する。

5 EOR 操作

次に改良四分木符号の EOR 操作を示す。AND 操作と同様に、各入力の先頭の quadrant を取り出し、その位置関係で3通りに分類する。

場合1については、2つの quadrant の色の組合せでさらに次の3通りに分かれる。

1. 2つとも black ... なにもしない、

2. black と gray ... gray の quadrant の部分配列を NOT 操作を作用してできる部分配列を持つ quadrant を出力する、

3. 2つとも gray ... 2つの部分配列を EOR 操作をして生成された部分配列を持つ quadrant を出力する。

次に2つの入力の先頭の quadrant を取り出す。

場合2については、大きい方の quadrant を4分割して、スタックに積み、この入力の先頭の quadrant を取り出す。

場合3については、前にある quadrant を出力し、この入力の先頭の quadrant を取り出す。

各 quadrant の更新が可能な限り、以上の処理を繰り返す。一方の入力が尽きた時、他方の入力の残りを出力する。

参考文献

[1] 横川完治：複雑な2値画像のための四分木符号の改良、情報処理学会第37回（昭和63年後期）全国大会、p.1560(1988).

[2] Gargantini, I.: An Effective Way to Represent Quadtrees, Comm. ACM, Vol.25, No. 12, pp.905-910(1982).