

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号
特許第4391978号
(P4391978)

(45) 発行日 平成21年12月24日 (2009. 12. 24)

(24) 登録日 平成21年10月16日 (2009. 10. 16)

(51) Int. Cl.

F I

HO 4 N 1/405 (2006. 01)

HO 4 N 1/40

B

請求項の数 49 (全 50 頁)

(21) 出願番号	特願2005-303410 (P2005-303410)	(73) 特許権者	505388322
(22) 出願日	平成17年10月18日 (2005. 10. 18)		ソフトウェア・イメージング・グループ・
(65) 公開番号	特開2006-121697 (P2006-121697A)		リミテッド
(43) 公開日	平成18年5月11日 (2006. 5. 11)		Software Imaging Gr
審査請求日	平成20年10月20日 (2008. 10. 20)		oup Limited
(31) 優先権主張番号	0423105.6		イギリス オーエックス4 2エイチエヌ
(32) 優先日	平成16年10月18日 (2004. 10. 18)		オックスフォード オックスフォード
(33) 優先権主張国	英国 (GB)		ビジネス パーク ガーシントン ロード
早期審査対象出願			9400
		(74) 代理人	100101454
			弁理士 山田 卓二
		(74) 代理人	100081422
			弁理士 田中 光雄
		(74) 代理人	100125874
			弁理士 川端 純市
			最終頁に続く

(54) 【発明の名称】 ビットマスク生成システム

(57) 【特許請求の範囲】

【請求項 1】

画像中の位置を、一定の範囲の影のそれぞれ1つに関連付けるマルチレベル画像データの要素を受ける受取り部と、

各々が前記範囲の影のそれぞれの影に関連付けられる1組の確率バイナリビットマスクアレイを表すデータを記憶するビットマスク記憶部であって、前記バイナリアレイ中のエントリーは、前記関連付けられた影を表すドットおよびギャップのパターンを定義するビットマスク記憶部と、

マルチレベル画像データの要素をハーフトーン画像データに変換するように動作可能な変換ユニットであって、前記変換は、

変換されるマルチレベル画像データの要素のために、前記影に関連付けられた前記ビットマスクアレイを選択し、

前記マルチレベルデータ要素によって識別された前記位置を使用して、前記選択されたビットマスク中の1つまたは複数のエントリーを識別し、および

前記ビットマスクアレイ中の前記識別された1つまたは複数のエントリーを、前記位置に対するハーフトーンデータとして出力することによって行われる変換ユニットと

を備え、

前記ビットマスク記憶部に記憶されたデータによって定義される前記バイナリビットマスクアレイ中のエントリーは、

各エントリーが、前記バイナリアレイ中で隣接している同じタイプの2つ以上のエント

リーからなるクラスタの一部をなし、または前記バイナリアレイ中の同じタイプのエントリーからなるクラスタの一部である同じタイプのエントリーに隣接するように並べられることを前提として、

各ビットマスクアレイ中の同じタイプのエントリーの広がり依存する距離関数を最大化するように分布され、

バイナリビットマスクアレイのエッジにある前記エントリーは、前記バイナリビットマスクアレイの逆側のエッジにある対応するエントリーに隣接していることを特徴とするマルチレベル画像データをハーフトーン画像データに変換する画像処理装置。

【請求項 2】

前記ビットマスク記憶部に記憶されたデータによって定義される前記バイナリビットマスクアレイ中のエントリーは、各エントリーが、前記バイナリアレイ中の同じタイプの他の少なくとも 1 つのエントリーに隣接するように並べられることを特徴とする請求項 1 に記載の画像処理装置。

10

【請求項 3】

前記ビットマスク記憶部に記憶されたデータによって定義される前記バイナリビットマスクアレイ中のエントリーは、各エントリーが、前記バイナリアレイ中の隣接している同じタイプの 3 つ以上のエントリーからなるクラスタの一部であるか、または前記バイナリアレイ中の同じタイプの 3 つ以上のエントリーからなるクラスタの一部である同じタイプのエントリーに隣接するように並べられることを特徴とする請求項 1 に記載の画像処理装置。

20

【請求項 4】

前記エントリーからなるクラスタは、同じタイプの 3 つのエントリーを含み、前記エントリーは、前記ビットマスク記憶部に記憶されたデータによって定義されるアレイ中に並べられ、そうすることによって、前記エントリーの 2 つが同じタイプの第 3 のエントリーに直接隣接しているエントリーを含むようになり、前記エントリーの前記 2 つは、前記同じタイプの前記第 3 のエントリーの前記バイナリアレイ中の位置に垂直方向に直接隣接しており、水平方向に直接隣接している位置にある前記同じタイプのエントリーを含むことを特徴とする請求項 3 に記載の画像処理装置。

【請求項 5】

前記バイナリアレイ中の位置にあるエントリーが、同じタイプのエントリーからなるクラスタの一部である同じタイプのエントリーに斜めに隣接している場合、前記バイナリアレイ中のエントリーは、前記タイプのエントリーからなるクラスタの一部である同じタイプのエントリーに隣接していることを特徴とする請求項 3 または 4 に記載の画像処理装置。

30

【請求項 6】

前記バイナリアレイ中の位置にある前記エントリーが、同じタイプのエントリーからなるクラスタの一部である同じタイプのエントリーに垂直方向に直接隣接しているか、または水平方向に直接隣接している場合、前記バイナリアレイ中のエントリーは、前記タイプのエントリーからなるクラスタの一部である同じタイプのエントリーに隣接していることを特徴とする請求項 3 または 4 に記載の画像処理装置。

40

【請求項 7】

前記ビットマスク記憶部に記憶されたデータによって定義される前記バイナリビットマスクアレイ中のエントリーは、各エントリーが、前記バイナリアレイ中の隣接している同じタイプの 3 つ以上のエントリーからなるクラスタの一部であるように並べられることを特徴とする請求項 3 に記載の画像処理装置。

【請求項 8】

前記範囲の影の中間範囲にある連続する影の少なくともいくつかのペアに関連付けられた前記バイナリビットマスクアレイは、明るい影に対する前記ペアのビットマスクアレイによって定義されるドットの前記パターンのすべてではないがほとんどが、前記ペアの暗い影に対するビットマスクによって定義されるドットの前記パターンのサブセットとして

50

含まれるようなものであることを特徴とする請求項 1 から 7 のいずれかに記載の画像処理装置。

【請求項 9】

前記連続する影に関連付けられたバイナリビットマスクアレイは、明るい影に対するビットマスクアレイによって定義されるドットの前記パターンのほとんどまたはすべてが、連続する次の暗い影に対するビットマスクによって定義されるドットの前記パターンのサブセットとして含まれるようなものであることを特徴とする請求項 1 ないし 8 のいずれかに記載の画像処理装置。

【請求項 10】

前記連続する影に関連付けられたバイナリビットマスクアレイのペアの間で異なる、対応するエントリーの数、データが前記ビットマスク記憶部に格納される前記バイナリアレイのすべてに対するしきい値より小さいことを特徴とする請求項 1 ないし 9 のいずれかに記載の画像処理装置。

10

【請求項 11】

前記バイナリビットマスクアレイの各々は、エントリーの $n \times m$ のアレイを含み、前記変換ユニットは、座標 x 、 y に関連付けられた位置について選択されたビットマスク中のエントリーとして、前記選択されたビットマスク中の、 n を法とする x 、 m を法とする y の位置にある前記エントリーを選択するように動作可能であることを特徴とする請求項 1 ないし 10 のいずれかに記載の画像処理装置。

【請求項 12】

20

前記ビットマスク記憶部は、1 組の確率バイナリビットマスクアレイを表すデータを格納し、マルチレベル画像の同じライン中の異なる影のためのマルチレベルデータの要素を処理するビットマスクデータは、連続する記憶場所に格納されることを特徴とする請求項 1 ないし 10 のいずれかに記載の画像処理装置。

【請求項 13】

前記ビットマスク記憶部は、複数の組のデータを含むデータを格納し、データの各組は、前記影の範囲にある影の数に対応するいくつかのバイナリ数を含み、前記データの組の各々は、マルチレベル画像のラインを処理するデータを含み、前記変換ユニットは、一群の数から、処理されるマルチレベル画像データのラインに対して、処理されるマルチレベル画像データの要素によって表される前記影に基づいて 1 つのバイナリ数を選択するように動作可能であることを特徴とする請求項 12 に記載の画像処理装置。

30

【請求項 14】

前記変換ユニットは、処理されるマルチレベル画像データのライン中のマルチレベルデータの要素の数をカウントするカウンタをさらに備え、前記変換ユニットは、マルチレベルデータの要素に対するハーフトーンデータとして、前記カウンタの現在の値を使用して選択される前記マルチレベルピクセルに対する前記マルチレベル値に基づいて選択されるバイナリ数のエントリーを出力するように動作可能であることを特徴とする請求項 12 または 13 に記載の画像処理装置。

【請求項 15】

画像中の位置を一定の範囲の影のそれぞれ 1 つに関連付けるマルチレベル画像データの要素を受け取るステップと、

40

前記範囲の影のそれぞれの影に関連付けられる 1 組の確率バイナリビットマスクアレイを表すデータを記憶するステップであって、前記バイナリアレイ中のエントリーは、前記関連付けられた影を表すドットおよびギャップのパターンを定義するステップと、

マルチレベル画像データの要素をハーフトーン画像データに変換するステップであって、前記変換は、

変換されるマルチレベル画像データの要素のために、前記影に関連付けられた前記ビットマスクアレイを選択し、

前記マルチレベルデータ要素によって識別された前記位置を使用して、前記選択されたビットマスク中の 1 つまたは複数のエントリーを識別し、および

50

前記ビットマスクアレイ中の前記識別された1つまたは複数のエントリーを、前記位置に対するハーフトーンデータとして出力することによって行われステップと

を備え、

前記記憶されたデータによって定義される前記バイナリビットマスクアレイ中のエントリーは、

各エントリーが、前記バイナリアレイ中で隣接している同じタイプの2つ以上のエントリーからなるクラスタの一部をなし、または前記バイナリアレイ中の同じタイプのエントリーからなるクラスタの一部である同じタイプのエントリーに隣接するように並べられることを前提として、

各ビットマスクアレイ中の同じタイプのエントリーの広がり依存する距離関数を最大化するように分布され、

バイナリビットマスクアレイのエッジにある前記エントリーは、前記バイナリビットマスクアレイの逆側のエッジにある対応するエントリーに隣接しているとみなされることを特徴とするマルチレベル画像データをハーフトーン画像データに変換する画像処理方法。

【請求項16】

前記1組の確率バイナリビットマスクアレイを表すデータを記憶するステップは、ビットマスクアレイを定義するデータを記憶するステップを含み、前記定義されたバイナリビットマスクアレイ中の前記エントリーは、各エントリーが、前記バイナリアレイ中の同じタイプの他の少なくとも1つのエントリーに隣接するように並べられることを特徴とする請求項15に記載の方法。

【請求項17】

前記1組の確率バイナリビットマスクアレイを表すデータを記憶するステップは、ビットマスクアレイを定義するデータを記憶するステップを含み、エントリーは、各エントリーが、前記バイナリアレイ中の隣接している同じタイプの3つ以上のエントリーからなるクラスタの一部であるか、または前記バイナリアレイ中の同じタイプの3つ以上のエントリーからなるクラスタの一部である同じタイプのエントリーに隣接するように並べられることを特徴とする請求項15に記載の方法。

【請求項18】

前記1組の確率バイナリビットマスクアレイを表すデータを記憶するステップは、ビットマスクアレイを定義するデータを記憶するステップを含み、同じタイプのエントリーからなるクラスタは、同じタイプの第3のエントリーに直接隣接しているエントリーを含む2つのエントリーを含み、前記エントリーの前記2つは、前記同じタイプの前記第3のエントリーの前記バイナリアレイ中の位置に垂直方向に直接隣接しており、水平方向に直接隣接している位置にある前記同じタイプのエントリーを含むことを特徴とする請求項17に記載の方法。

【請求項19】

前記バイナリアレイ中の位置にあるエントリーが、同じタイプのエントリーからなるクラスタの一部である同じタイプのエントリーに斜めに隣接している場合、前記バイナリアレイ中のエントリーは、前記タイプのエントリーからなるクラスタの一部である同じタイプのエントリーに隣接していることを特徴とする請求項17または18に記載の方法。

【請求項20】

前記バイナリアレイ中の位置にあるエントリーが、同じタイプのエントリーからなるクラスタの一部である同じタイプのエントリーに垂直方向に直接隣接しているか、または水平方向に直接隣接している場合、前記バイナリアレイ中のエントリーは、前記タイプのエントリーからなるクラスタの一部である同じタイプのエントリーに隣接していることを特徴とする請求項17または18に記載の方法。

【請求項21】

前記1組の確率バイナリビットマスクアレイを表すデータを記憶するステップは、ビットマスクアレイを定義するデータを記憶するステップを含み、各々が、前記バイナリアレイ中の隣接している同じタイプの3つ以上のエントリーからなるクラスタの一部であるよ

うにエントリーを並べることを特徴とする請求項 17 に記載の方法。

【請求項 22】

前記影の範囲の中間範囲にある連続する影の少なくともいくつかのペアに関連付けられた前記バイナリビットマスクアレイは、明るい影に対する前記ペアのビットマスクアレイによって定義されるドットの前記パターンのすべてではないがほとんどが、前記ペアの暗い影に対するビットマスクによって定義されるドットの前記パターンのサブセットとして含まれるようなものであることを特徴とする請求項 15 ないし 21 のいずれかに記載の方法。

【請求項 23】

前記連続する影に関連付けられたバイナリビットマスクアレイは、明るい影に対するビットマスクアレイによって定義されるドットの前記パターンのほとんどまたはすべてが、連続する次の暗い影に対するビットマスクによって定義されるドットの前記パターンのサブセットとして含まれるようなものであることを特徴とする請求項 15 ないし 22 のいずれかに記載の方法。

【請求項 24】

前記連続する影に関連付けられたバイナリビットマスクアレイのペアの間で異なる、対応するエントリーの数は、前記ビットマスク記憶部にデータが格納される前記バイナリアレイのすべてに対するしきい値より小さいことを特徴とする請求項 15 ないし 23 のいずれかに記載の方法。

【請求項 25】

前記ビットマスクアレイの各々は、エントリーの $n \times m$ のアレイを含み、前記座標 x 、 y に関連付けられた位置に対する選択されたビットマスク中のエントリーの識別は、前記選択されたビットマスク中の、 n を法とする x 、 m を法とする y の位置にあるエントリーを識別するステップを含むことを特徴とする請求項 15 ないし 24 のいずれかに記載の方法。

【請求項 26】

前記 1 組の確率バイナリビットマスクアレイを表すデータを記憶するステップは、マルチレベル画像の同じライン中の異なる影のマルチレベルデータの要素を処理するビットマスクデータを、連続する記憶場所に格納するステップを含むことを特徴とする請求項 15 ないし 25 のいずれかに記載の方法。

【請求項 27】

処理されるマルチレベル画像データのライン中のマルチレベルデータの要素の数をカウントするステップと、マルチレベルデータの要素に対するハーフトーンデータとして、前記カウンタの現在の値を使用して選択されるバイナリビットマスクアレイのエントリーを出力するステップとをさらに備えたことを特徴とする請求項 26 に記載の方法。

【請求項 28】

ランレングスデータの複数の要素を受け、

前記ランレングスデータの各要素に、第 1 のタイプのいくつかのエントリーが含まれ、その後、前記ランレングスデータ要素の各々に対して別のタイプのエントリーが続くデータのバイナリアレイを生成し、かつ

前記バイナリアレイの一連のバイナリ数のグループに対して、ビットマスクアレイを表すデータを生成するために数の各一部に対して排他的論理和演算を実施することによって、1 組の確率バイナリビットマスクを表すデータを生成するステップをさらに備えたことを特徴とする請求項 15 ないし 27 のいずれかに記載の方法。

【請求項 29】

請求項 15 ないし 28 のいずれかに記載の方法に従ってマルチレベル画像データを処理するステップと、

前記出力ハーフトーンデータを使用して、プリンタに画像を印刷させるステップとを備えたことを特徴とする印刷方法。

【請求項 30】

請求項 1 ないし 1 4 のいずれかに記載の画像処理装置と、
出力ハーフトーン画像データを受け、前記受けたハーフトーン画像データに対応する画像を記録するように動作可能なプリンタと
を備えたことを特徴とする印刷システム。

【請求項 3 1】

プログラム可能なコンピュータを、請求項 1 ないし 1 4 のいずれかに記載の画像処理装置として構成させることを特徴とするプリンタドライバ。

【請求項 3 2】

第 1 および第 2 のタイプのエントリーからなるバイナリアレイを表すデータを記憶するステップであって、前記バイナリアレイ中のエントリーは、各エントリーが、前記バイナリアレイ中の隣接している同じタイプの 2 つ以上のエントリーからなるクラスタの一部であるか、または前記バイナリアレイ中の同じタイプのエントリーからなるクラスタの一部である同じタイプのエントリーに隣接するように並べられることを前提として、各ビットマスクアレイ中の同じタイプのエントリーの広がり依存する距離関数を最大化するように分布され、

前記エントリーのうちバイナリビットマスクアレイのエッジにあるエントリーは、前記バイナリビットマスクアレイの逆側のエッジにある対応するエントリーに隣接しているステップと、

第 1 のタイプの個別のエントリーを変更して前記第 2 のタイプのエントリーにすることにより前記距離関数が減じられる量を識別するステップと、

第 1 のタイプの個別のエントリーを変更して前記第 2 のタイプのエントリーにすることから生じる、前記距離関数中の減少のサイズに基づいて、第 1 のタイプの 1 組のエントリーを変更対象として検討すると決定するステップと、

前記決定された 1 組の中にあるエントリーを連続して選択し処理して、前記選択されたエントリーが前記第 2 のタイプのエントリーとなるように変更すると、前記記憶されたアレイにエントリーのアレイを定義させることになるかどうかを決定するステップであって、少なくともいくつかのエントリーは、同じタイプの 2 つ以上のエントリーからなるクラスタの一部ではなくなるか、または同じタイプの 2 つ以上のエントリーからなるクラスタにもはや隣接していないエントリーであるステップと、

前記選択されたエントリーの変更が、前記記憶されたアレイにエントリーのアレイを定義させるようなものではない場合、このように変更することによって、前記記憶されたアレイを更新するステップであって、少なくともいくつかのエントリーは、同じタイプの 2 つ以上のエントリーからなるクラスタの一部ではなくなるか、または同じタイプの 2 つ以上のエントリーからなるクラスタにもはや隣接していないエントリーであるステップと、

前記選択されたエントリーを含む、前記第 1 のタイプの隣接している複数のエントリーの代替変更を決定するステップであって、少なくともいくつかのエントリーは、同じタイプの 2 つ以上のエントリーからなるクラスタの一部ではなくなるか、または同じタイプの 2 つ以上のエントリーからなるクラスタにもはや隣接していないエントリーであるエントリーのアレイを、前記単一のエントリーの変更が、前記記憶されたアレイに定義させる場合、前記記憶されたアレイにエントリーのアレイを定義させるようなものではなく、少なくともいくつかのエントリーは、同じタイプの 2 つ以上のエントリーからなるクラスタの一部ではなくなるか、または同じタイプの 2 つ以上のエントリーからなるクラスタにもはや隣接していないエントリーであるステップと、

前記第 1 のタイプのエントリーのうち、前記選択された組のいずれかの個々のエントリーを変更すると、少なくともいくつかのエントリーは、同じタイプの 2 つ以上のエントリーからなるクラスタの一部ではなくなるか、または同じタイプの 2 つ以上のエントリーからなるクラスタにもはや隣接していないエントリーであるエントリーのアレイを、前記記憶されたアレイに定義させることになると決定される場合、前記選択された組の 1 つに関連付けられた、前記第 1 のタイプのエントリーの、決定された代替変更を使用して、前記記憶されたアレイを更新するステップと

10

20

30

40

50

を備えたことを特徴とするビットマスクアレイ生成方法。

【請求項 3 3】

前記隣接している同じタイプの 2 つ以上のエントリーからなる各クラスタは、前記バイナリアレイ中の 1 対の隣接している同じタイプのエントリーを含むことを特徴とする請求項 3 2 に記載の方法。

【請求項 3 4】

前記隣接している同じタイプの 2 つ以上のエントリーからなる各クラスタは、前記バイナリアレイ中の隣接している同じタイプの 3 つ以上のエントリーを含むことを特徴とする請求項 3 2 に記載の方法。

【請求項 3 5】

前記同じタイプの 3 つ以上の隣接しているエントリーからなる各クラスタは、同じタイプの第 3 のエントリーに直接隣接しているエントリーを含む 2 つのエントリーを含み、前記 2 つのエントリーは、前記同じタイプの前記第 3 のエントリーの前記バイナリアレイ中の位置に垂直方向に直接隣接しており、水平方向に直接隣接している位置にある前記同じタイプのエントリーを含むことを特徴とする請求項 3 4 に記載の方法。

【請求項 3 6】

前記バイナリアレイ中の位置にあるエントリーが、同じタイプのエントリーからなるクラスタの一部である同じタイプのエントリーに斜めに隣接している場合、前記バイナリアレイ中のエントリーは、前記タイプのエントリーからなるクラスタの一部である同じタイプのエントリーに隣接していると決定されることを特徴とする請求項 3 2、3 4、または 3 5 に記載の方法。

【請求項 3 7】

前記バイナリアレイ中の位置にあるエントリーが、同じタイプのエントリーからなるクラスタの一部である同じタイプのエントリーに垂直方向に直接隣接しているか、または水平方向に直接隣接している場合、前記バイナリアレイ中のエントリーは、前記タイプのエントリーからなるクラスタの一部である同じタイプのエントリーに隣接していると決定されることを特徴とする請求項 3 2、3 4、または 3 5 に記載の方法。

【請求項 3 8】

前記第 1 のタイプの 1 組のエントリーを変更の検討対象になると決定するステップは、前記エントリーが前記バイナリアレイ中の前記第 2 のタイプのエントリーから区別される限度を示す関数に基づいて、前記第 1 のタイプの 1 組のエントリーを選択するステップを含むことを特徴とする請求項 3 2 ないし 3 7 のいずれかに記載の方法。

【請求項 3 9】

前記バイナリアレイを更新するステップは、前記第 1 のタイプのエントリーが前記第 2 のタイプのエントリーとなるような最小数の変更を伴う前記バイナリアレイの変更を使用して、前記バイナリアレイを更新するステップを含み、前記変更が前記バイナリアレイ中の前記第 1 のタイプのエントリーの最大拡散を生じさせると決定することを特徴とする請求項 3 8 に記載の方法。

【請求項 4 0】

前記第 1 のタイプの 1 組のエントリーに対して決定された前記代替変更の各々を、アレイのエリアが変更される限度を示す値に関連付けるステップをさらに備え、前記バイナリアレイを更新するステップは、変更されている前記バイナリアレイの最小エリアを示す値に関連付けられた前記バイナリアレイ中の 1 つのエントリーの最大拡散を生じる変更を選択するステップを含むことを特徴とする請求項 3 9 に記載の方法。

【請求項 4 1】

前記バイナリアレイが更新された後で、

処理される前記第 1 のタイプの前記エントリーを、前記第 2 のタイプのエントリーとするように設定することによって、前記バイナリアレイを変更し、各エントリーが、前記バイナリアレイ中の隣接している同じタイプの 2 つ以上のエントリーからなるクラスタの一部であるか、または前記バイナリアレイ中の同じタイプのエントリーからなるクラスタ

10

20

30

40

50

の一部である同じタイプのエントリーに隣接するようにエントリーが並べられているアレ
イを前記バイナリアレイが含むまで、さらに隣接している前記第1のタイプのエントリー
を変更し、

前記バイナリアレイが変更在先立って前記第1のタイプの同じ数のエントリーを含むま
で、各エントリーが、前記バイナリアレイ中の隣接している同じタイプの2つ以上のエン
トリーからなるクラスタの一部であるか、または前記バイナリアレイ中の同じタイプのエン
トリーからなるクラスタの一部である同じタイプのエントリーに隣接するようにエン
トリーが並べられるアレイを前記バイナリアレイが確実に含むようにさせるように、前記第
2のタイプの前記バイナリアレイ中のエントリーを変更し、

前記変更されるアレイが、前記第1のタイプの前記エントリーが、変更在先立って前記
バイナリアレイ中で、前記第1のタイプの前記エントリーよりも互いに区別されるアレイ
を含むかどうかを決定し、

前記第1のタイプの前記エントリーをすべて処理した後、前記バイナリアレイ中のエン
トリーの拡散における最大限の改良に関連付けられた前記決定した変更を使用して、前記
バイナリアレイを変更することによって、前記第1のタイプの前記エントリーの各々を連
続して処理するステップをさらに備えたことを特徴とする請求項32ないし40のいずれ
かに記載の方法。

【請求項42】

前記ビットマスクを更新するステップは、元々記憶されたアレイ中の前記第2のタイプ
のエントリーによって表される前記変更されるビットマスク中の前記第1のタイプのエン
トリーの数、あらかじめ設定されたしきい値を超えないようなものであることを特徴と
する請求項32ないし41のいずれかに記載の方法。

【請求項43】

前記更新されたビットマスクアレイを定義するデータを出力するステップをさらに備え
たことを特徴とする請求項32ないし42のいずれかに記載のビットマスクを生成する方
法。

【請求項44】

プログラム可能なコンピュータに、請求項15ないし29、及び、32ないし43のう
ちのいずれかに記載の方法を実施させるコンピュータプログラム。

【請求項45】

請求項44に記載のコンピュータプログラムを記憶することを特徴とする記録媒体。

【請求項46】

請求項44に記載のコンピュータプログラムを記憶することを特徴とするコンピュータ
ディスク。

【請求項47】

請求項44に記載のコンピュータプログラムを記憶することを特徴とする磁気ディスク

。

【請求項48】

請求項44に記載のコンピュータプログラムを記憶することを特徴とする光学ディスク

。

【請求項49】

請求項44に記載のコンピュータプログラムを記憶することを特徴とする光磁気ディス
ク。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、画像処理に関する。より具体的には、本発明は、マルチレベルの画像をハー
フトーン画像に変換することに関するものである。さらにそのような画像を、レーザープ
リントを使って印刷することに関するものである。

【背景技術】

【0002】

画像において色の濃さ (shades of color) を表現することは、プリンタにとって長い間の懸案であった。ブラウン管およびLCD等の表示装置は、ほとんどの場合、色純度およびグレースケールが変化する画像を生成することができるが、ほとんどのレーザープリンタは、ページ上にトナーを吸着させるか否かを行うことができるだけである。したがって、印刷された画像において中間の濃さを表すためには、マルチレベルの画像を、印刷領域ならびに非印刷領域により影が現されるものに変換する必要がある。

【0003】

マルチレベルの画像を、印刷領域ならびに非印刷領域の組み合わせにより影が表される画像に変換することは、ハーフトーニングとして知られている。ハーフトーニングに対しては、さまざまな取り組みがなされている。

10

【0004】

既知の方法の一つに誤差拡散(error diffusion) がある。誤差拡散の基本的な概念は、ある階調の値 (例えば、黒を表す0と白を表す255範囲の値) を有する画素が印刷された場合に、当該階調値を、しきい値と比較するものである。階調値がしきい値を超えない場合、画素が印刷される。階調値がしきい値を超える場合、画素は印刷されない。次に、元の画像の階調値と、印刷画像において実際に現れた階調値 (0か255のいずれか) の差であるエラー値が、演算される。このエラー比率が複数の近接する複数の画素の階調値に加えられ、あるいは、それから減じられる。

20

【0005】

誤差拡散を用いるハーフトーニングは、高い品質の画像を作り出すが、誤差拡散は、多くの欠点を有している。第一に、誤差値が近接する画素に加えられ、あるいは、それから減じられるにつれ、誤差拡散は、画素ごとに多くの演算を必要とする。比較的演算能力が求められ、したがって、誤差拡散を用いるハーフトーニングは、比較的速度が遅い。

【0006】

また、誤差拡散を用いるハーフトーニングの場合、特定の問題を生じさせる、25%、33%、および50%等の、グレースケールが存在する。例えば、50%のグレースケールのための典型的なパターンは、チェッカーボード・パターンである。しかし、誤差拡散は、チェッカーボード・パターンのドットの代わりに、ドットによる行又は列を生成しかねない。ドットによる行又は列を生成させることから生じる、かかる産物は、最終的な出力画像において識別可能である。

30

【0007】

誤差拡散の前記欠点を克服するため、さまざまな代替の方法が提案されている。ある提案は、しきい値のアレイあるいはデイズマトリクスを使用することである。かかるシステムにおいては、処理される画像にわたってモザイク模様になる、固定されたしきい値のアレイが生成される。いずれの画素座標においても、画像のグレースケール値は、しきい値アレイ中の対応する値と比べられる。画像のグレースケール値が、しきい値アレイのそれよりも小さい場合、画素が印刷される。画像のグレースケール値が、しきい値アレイの対応する値よりも大きい場合、画素は印刷されない。特定の方法によりしきい値アレイ中のしきい値を配置することにより、達成可能な誤差拡散と同様の結果を生じさせ、適切なしきい値アレイを生成するシステムの例が提案されている (例えば特許文献1または2参照)。

40

【0008】

しきい値アレイを用いるハーフトーニングについての代替システムは、ビットマスクを用いるシステムである。しきい値アレイは、例えば0から255の範囲のしきい値のアレイから構成されるが、ビットマスクを用いるシステムにおいては、それぞれのアレイが0と1の配置により構成される256のビットマスクアレイが記憶される。各ビットマスクにおける0と1の配置は、ビットマスクに関連するグレースケールを表すドット配列を表している。したがって、例えば、明るい色を表す、あるグレースケール用のビットマスクは、印刷がされていないことを表す0を含むアレイからほぼ構成されている。これとは逆に、暗い色用のビットマスクは、ほぼ1から構成されている。

50

【0009】

マルチレベル画像が、ビットマスクを用いて変換される場合、まず、記憶されたビットマスクの1つを選択するため、印刷されるイメージ画素のグレースケール値が用いられる。次に、ビットマスクにおける入力1つを識別するため、印刷される当該画素のx、y座標が用いられる。これは、グレースケールのレベルによって決定される特定のビットマスク用の1と0の比率を有する1か0のいずれかであって、ビットマスクが表現しようとするものである。識別されたビットマスク入力が1と等しくトナーが塗工されず、識別されたビットマスク入力が0と等しい場合、トナーがページ上に塗工される。

【0010】

ビットマスクを用いた印刷は、しきい値アレイを用いるシステムを越える顕著な効果を有する。表現される各グレースケールのために、ビットマスクが記憶されるにつれ、各グレースケールのために印刷されるドット配置が最適化される。したがって、グレースケールの特定の影を表すトナーが、外見上好ましくなるよう分布される。すなわち、独立したトナーのセットではなく、色の濃さとして認識されるよう、ドットの各配置を、入念に演算することができ。従来、このことは、配置用の空間周波数を決定するため、候補となるドット配置を処理することにより達成されていた。好ましい外観を実現するため、低い空間周波数ではなく、高い空間周波数を有する配置が選択される。これにより、色の濃さではなく、ドットのパターンという印象を与えるドットの過度な凝集を確実に減少させる（例えば、特許文献3参照）。

【0011】

【特許文献1】英国特許2352579号

【特許文献2】米国特許5726772号

【特許文献3】米国特許4910501号

【発明の開示】

【発明が解決しようとする課題】

【0012】

しかし、ビットマスクシステムは、3つの不利益を甘受しなければならなかった。2の近似する色の濃さが隣同士に表される場合、2つの濃さの間の境界は、影が混ざり合ったように見えることが好ましい。しかし、画像の2つの部分が隣接して印刷されると、当該2つのレベルのドット配置を最適化しても、認識可能な境界を生じさせてしまう。これは、一の配置の端部におけるドットの拡散が、他の影用の配置におけるドット位置に近い位置にあるからである。したがって、配置の異なるものを隣接して印刷すると、前記2の濃さの間の境界にドットの凝集を生じさせてしまう。このような問題は、輪郭削り (contouring) として知られている。

【0013】

ビットマスクアレイが有する二番目の問題は、一つのしきい値アレイを記憶するのに必要とされる記憶容量は、ビットマスクのセットを記憶するのに要求される容量よりかなり少ないことである。したがって、例えば、0から255の範囲のしきい値用の32×32のしきい値アレイの場合、210の8ビット数が記憶のため必要となる。これに対し、256の32×32ビットマスクを表すデータと近似するデータを記憶するため、32倍に匹敵するデータを記憶しなければならない。

【0014】

より具体的には、3つ目の問題はレーザープリンタを使って画像を印刷する際に発生する。一般的に、レーザープリンタでトナーのドット1つ分を配置または描画 (rendering) する作業の結果は、非常に貧弱なものとなる。というのは、印刷媒体上の極くわずかな部分にトナーを沈殿させようとしても、多くの場合そのスポットにトナーが付着しないからである。同様に、トナーの沈殿がもつこの精度の限界ゆえに発生した別々の印刷抜け部分をカバーしようとしても、トナーは抜けた部分だけでなく領域全体に付着してしまうという類似問題も存在する。ビットマスク生成は、マスク全体に1と0を完全に行き渡らせるための最適化を画像を散在した個別トナードットと非印刷領域によって表現する

ので、レーザープリンタによる画像の印刷結果が非常に貧弱になってしまうのである。

【0015】

したがって、ビットマスクを基礎とした印刷システムは、各レベルで印刷されるドットの配列を最適化し、かつ上述の問題緩和する能力を求められる。

【課題を解決するための手段】

【0016】

本発明のある側面は、出力ハーフトーン画像に表れる孤立したドットの発生を低減させるビットマスクのセットを生成可能とする、ビットマスク生成システムを提供する。

【0017】

本発明の本側面は以下の形で実施される。すなわち、ビットマスク生成システムは (a) 結果として出来上がるビットマスクがドットの織りなす模様としてではなく、濃淡をもったグレーとして見えるように、0と1を出来る限り均等にビットマスク全体に広げ、(b) かつ、0と1のエントリーをビットマスクにおいてグループ化することによって、レーザープリンタによって貧弱に描画される部分、すなわち、孤立したドットが作る模様や非印刷領域からなる色の濃淡を描画しなければならないという事態を回避すべきである。すなわちビットマスク生成システムは (a) と (b) 間のバランスを配慮するものでなくてはならない。

【0018】

本側面において、この目的を達成すべく、ビットマスク生成システムは、複数の1から成るクラスタを構成要素とする淡いグレーに関連づけられたビットマスクを生成する。そもそもビットマスク生成は、レーザープリンタのために行うのであるが、レーザープリンタが高信頼度をもって描画を行うことのできるように、上述の各クラスタの大きさが十分確保される。ビットマスク生成システムが、ビットマスクにおける中間的なグレーレベル用の処理として、(マスクに新たなクラスタを丸々1個追加するのではなく) 1のエントリーからなる既存のクラスタを拡大することによって、エントリーの広がり改善することが可能であると判断された場合、グレーの連続した濃淡を実現すべく、このアプローチが用いられる。とはいえ、この処理の結果0エントリーからなる極度に小さなクラスタが発生しないということが条件であるが。最後に、グレーの中でも最も濃いものに関しては、ビットマスク生成システムは、0エントリーを構成要素とする諸クラスタの総体からなるビットマスクアレイを決定し、それらのクラスタが決してレーザープリンタによって印刷されないようにする。本発明のさらなる側面は、隣り合うグレーレベル間に現れる輪郭を軽減するビットマスクが生成されるという点にある。本発明のさらなる別側面は以下の形で実施される。すなわち、各レベル用に1と0の広がり最优化されるにもかかわらず、ビットマスクの生成と最適化がなされる際のビットマスク上のエントリー数(この数は、連続したグレーレベル用のビットマスク間で異なる)は、現在のしきい値よりも少ない。

【0019】

本発明の他の側面は、圧縮形式により記憶されるビットマスクを生成可能とする、ビットマスク生成システムを提供する。

【0020】

本発明の本側面は以下の形で実施される。すなわち、1つのグレーレベル用のビットマスクの多くの部分は、次の連続レベル用のビットマスクの部分的形成のためにコピーされる。次に、グレーレベルを連続化させるために、ビットマスクのある領域を次のビットマスクにコピーするという作業が、圧縮されたビットマスク表象生成のために行われる。

【0021】

本発明の別の側面ならびに実施形態は、添付する図面に示された特定の実施形態を参照することにより明らかになる。

【発明を実施するための最良の形態】

【0022】

ビットマスクを用いた印刷システムの概要

まず、本発明に基づくビットマスクを用いた印刷の概略を、図1Aならびに図1Bを参照

10

20

30

40

50

して説明する。

【0023】

図1Aは、画像の印刷に伴うステップを示している。印刷される元の画像100の一部が示される。この例において、かかる部分100は、似てはいるが同じでないグレーの影 (shade of grey) を有する2の隣接する領域101、102を含んでいる。最初に、元の画像100は、アレイにおける各画素が、元の画像における画素の濃さを示す値を有するマルチレベル画素200のアレイとして、コンピュータのメモリ内に記憶される。これにより、例示した画像100の場合、領域101が影225に対応し、領域102が影224に対応すると、図1Aに示されたマルチレベル画素データ200が記憶される。

【0024】

10

画像が印刷される際、アレイ内の各2値の画素が0又は1の値を有する2値の画素300を生成するため、マルチレベル画素200が用いられる。マルチレベル画素200の2値画素300への変換は値1を有する2値画素に変換される特定の値を有するマルチレベル画素の割合が、徐々に、明るいグレーの濃さを表す画素について減少させるようなものである。2値の画素300のアレイが生成されると、次に、当該2値画素300は、印刷領域と非印刷領域のパターンを有する出力像400を生成するため、値1を有する2値画素300における各画素を、レーザープリンタがトナーの塗工で印刷するよう動作させるため、用いられる。

【0025】

2値画素の値を設定するため、ビットマスクアレイのセットが記憶される。図1Bは、それぞれが各グレーレベル用であってマルチレベル画素を示すことができる、256ビットのマスクアレイ350 1 - 255から350 - 0を記憶する、メモリ310を示している。図1Bにおいては階調255、225、224ならびに0用ビットマスクアレイの一部の詳細が示されている。

20

【0026】

図1Bから見て取れるように、黒色であること示すレベル0と関連するビットマスクアレイ350 - 0は、全体が1で占められたアレイにより構成される。これとは逆に、白色であること示すレベル255と関連する2値アレイ350 - 255は、全体が0で占められたアレイにより構成される。連続した暗いグレー濃度用のアレイのための1の入力数が増加した場合、レベル224および225等の中間のグレー値のための中間のビットマスクアレイは、1の入力と0の入力が混在するビットマスクアレイ350 - 224ならびに350 - 225を含む。

【0027】

30

2値の画素値が設定されると、メモリ310に記憶されたビットマスクアレイ350 - 255から350 - 0の一つを選択するため、当該2値画素に対応するマルチレベル画素の値が用いられる。次に、マルチレベル画素の座標が、選択されたビットマスクから各入力を選択するため用いられる。次に、選択された入力値である1又は0は、その2値画素用の値として記憶される。

【0028】

図1Aの2値画素のアレイを、図1Bに示すレベル224及び225のビットマスクアレイと比較することにより、この方法でビットマスクアレイを用いることの効果が、ビットマスクアレイの一部を、生成された2値の画素300のアレイに複製することであることが判る。これにより、225の値を有するマルチレベル画素に対応する2値画素300の最初の3行は、グレーレベル225用のビットマスクアレイ350 - 25の最初の3行の複製に対応する。同様に、244の値を有するマルチレベル画素に対応する2値の画素300の次の3行は、グレーレベル224用のビットマスクアレイ350-224における数の2番目の3行の入力の複製に対応する。

40

【0029】

外見上好ましい画像を生成するため、各グレーレベル用のビットマスクアレイにおける1ならびに0の配置は、印刷された画像が、ドットの独立したパターンではなく、グレーの影であると認識されるようトナーを拡散させること、が重要である。印刷された画像の2の隣接する領域が、同じようなグレーの影で構成されている場合、境界が一つのグレーレベルから次のレベルへと融合することも好ましい。しかし、各レベル用のドットの拡散を最適化することは、画像中に2の異なる階調が互いに隣接する場合、輪郭削り (contourin

50

g) として知られる問題を生じさせる。ドットの拡散を表すためそれ自身が最適化された異なるアレイから入力を選択することは、境界に、ドットの凝集、あるいは、ギャップを生じさせてしまうからである。

【0030】

本発明によると、このような問題を緩和するビットマスク350-0から350-255のセットが提供される。このことは、特定のグレーの影を表すためのドットの拡散を生じさせるため、各ビットマスクアレイは、最適化されているが、かかる最適化のプロセスはあるグレーレベル用のアレイのほとんどの入力が、かかるアレイにおける隣接する階調と同じである、ビットマスクのセットを有することにより達成される。このことは、レベル225用のビットマスクアレイ350-225における入力のほとんどが、レベル224用のビットマスクアレイ350-224における対応する入力と等しいレベル224ならびに225用のアレイの類似性により、図1Bに示されている。

10

【0031】

また、孤立したドットやホールを最終的な画像において描画しなくてはならない状況の発生回数を減らすために（このことに関する詳細は後述するが）、ビットマスクの生成時に、1と0の各エントリは各ビットマップにおいてグループ化されクラスタ化される。こうして、異なる濃度を印刷しなければならぬ領域が意図された通り、濃度を違えて印刷する確実性（信頼度）がアップする。

【0032】

各グレーレベル用のビットマスクの最適化を行うことにより、ビットマスクが、ドットの集合ではなく、影であると認識されるドットのパターンを生成することを、確実とする。しかし、グレーレベルの台のビットマスクの多くが、次のグレーレベル用のビットマスクに対応するので、印刷された画像中のドットの配置であって、隣接する階調の領域間の境界に沿ったものは、ドットが外見上好ましい拡散を達成することもできる。また、一のレベルにおけるビットマスクアレイのほとんどの部分が他のものの当該分と等しいことを確保することにより、ビットマスクアレイのセットは、後に詳細を説明する圧縮に非常に適したものとなる。

20

【0033】

ビットマスクアレイプリンタドライバ、ならびに印刷画像を生成するシステム

ここで、ビットマスクアレイ、本発明に基づき前記ビットマスクアレイを組み込むプリンタドライバ、を生成するシステムの詳細について図2を参照して説明する。

30

【0034】

よく知られているように、プリンタドライバは、プリンタの動作を制御するソフトウェアプログラムである。したがって、各プリンタのメーカーは、自身の特定のプリンタに適したプリンタドライバを要求する。この目的を達するため、各プリンタメーカーは、特定のプリンタにおいて利用可能なプリンタの機能を選択可能であり、適切なプリンタドライバを生成可能であるように、各プリンタドライバ制作会社によりプリンタドライバ作成キットが作られる。

【0035】

図2を参照すると、プリンタドライバ制作会社により用いられるビットマスク生成コンピュータ1が設けられている。かかるビットマスク生成コンピュータ1は、プリンタドライバに組み込まれるビットマスクアレイデータを生成するようプログラムされている。次に、プリンタメーカーによって用いられるプリンタドライバ生成コンピュータ2が設けられている。かかるプリンタドライバ生成コンピュータ2はビットマスクアレイ生成コンピュータ1により生成されたビットマスクアレイデータを組み込むプリンタドライバを生成するためのプリンタドライバ生成キットを有するコンピュータを備えており、マルチレベル画像データを、その後、印刷可能なハーフトーン画像データに変換するため、プリンタドライバが、ビットマスク生成コンピュータ1によって前に生成されたデータを用いる場合、最終的に生成されたプリンタドライバは、ホストコンピュータ2のメモリおよびデジタル複製部4内でロードされる。

40

50

【0036】

後に詳細を説明するようにビットマスクアレイ生成コンピュータ1によって生成されたビットマスクアレイは、外見上好ましい配列で分散される各グレーレベル用に生成されたドットのパターンを生じさせるようなものである。また、かかる生成は、異なる階調が、隣接して印刷される場合、隣接するビットマスクアレイのほとんどの部分が同じであるので、異なるレベル用のビットマスクアレイにより表されるドットパターンが、許容できる程度の量の輪郭削りを保持するようなものである。本生成方法はまた、生成されたハーフトーン画像において孤立したドットやホールが発生する回数を減らすものである。具体的には、本生成方法は、孤立したドットやホールが高信頼度をもって印刷することのできない状態にある画像を、レーザープリンタによる印刷に適したものにする。

10

【0037】

本実施形態において、ビットマスク生成コンピュータ1は、そのメモリ内に、印刷される階調の範囲を示すドットの位置を表すビットマスクアレイのセットを生成する生成モジュール、および、生成されたデータを圧縮するための圧縮モジュール9、を備えている。生成モジュール8により、ビットマスクのセットが生成されると、それらは、圧縮されたビットマスクデータを生成する圧縮モジュール9に送られる。次に、圧縮されたビットマスクデータは、その後、プリンタドライバ生成コンピュータ2に送られるCDROM10上に記録される。

【0038】

当該プリンタドライバ生成コンピュータ2は、CDROM10上に記録された圧縮データを読み取る。また、プリンタドライバ生成コンピュータ2のメモリ内には、テキストドライバ11、画像ドライバ13のセットおよびドライバエンジン14のセットが設けられる。

20

【0039】

前記テキストドライバは、テキストデータに対応する画像を印刷するため、テキストデータを処理し、テキストデータをプリンタ命令に変換するための、従来のプリンタドライバ用テキストドライバを備えている同様に画像ドライバ13は、画像データを処理し、画像データをプリンタの命令に変換する、画像処理モジュールを備えている。ドライバエンジン14は文書をプリンタの命令に変換するため、テキストドライバおよびプリンタドライバを調整する、関数のライブラリを備えている。

【0040】

使用に際し、プリンタドライバ生成コンピュータ2は、CDROM10から読み取った、圧縮済みのビットマスクデータを、生成されるプリンタドライバ内に含まれる選択された画像ドライバ13内に組み込む。次に、選択された画像ドライバ13および選択されたテキストドライバ11ならびにドライバエンジン14を表すデータは、CDROM20、21上に、プリンタドライバとして記憶される。CDROM20、21上に記憶されたプリンタドライバは、次に、ホストコンピュータ3およびデジタル複製部4のメモリ内でロードされる。

30

【0041】

ホストコンピュータ43の場合、プリンタドライバ生成コンピュータ2により記録されたCDROM20から読み取られたデータは、ホストコンピュータ3のメモリ内にプリンタドライバとして記憶される。ホストコンピュータ3のメモリ内には、ワードプロセッシングプログラム等の文書生成プログラム27を含む他のプログラムも記憶される。文書生成部27により生成された文書ファイルが印刷されると、ビットマスク生成コンピュータ1によって前に生成された圧縮ビットマスクを伴うプリンタドライバ25が呼び出される。次に、プリンタドライバ25は、圧縮されたビットマスクデータを解凍し、後に画像29を印刷するホストコンピュータ3に接続されたレーザープリンタ28に送られる、ハーフトーン画像データを生成するため、解凍されたビットマスクデータを用いる。

40

【0042】

プリンタドライバ生成部9により生成されたデジタル複製部4用のプリンタドライバの場合、その上に、生成されたプリンタドライバを表すデータが記憶されたCDROM21から当該データが読み出され、デジタル複製部4のメモリ内にプリンタドライバ30として記憶され

50

る。このようなデジタル複製部は、スキャナ31およびレーザープリンタ32を備えている。画像が複製されると、デジタル複製部のスキャナ31は、まず、画像を走査する。次に、ビットマスク生成コンピュータ1により生成された圧縮済みビットマスクデータを含む、プリンタドライバであって、走査された画像を処理し、デジタル複製部のレーザープリンタ32に印刷済みの画像36を出力させるものが呼び出される。

【0043】

ビットマスクデータ生成の概略

ハーフトーン出力画像29、36を生成するため用いることができるビットマスクのセットを生じさせるビットマスク生成コンピュータ1の生成モジュール8によりビットマスクを生成することにより、画像を表すトナーが、好ましい態様により配置され、輪郭削りが制御されること、また、印刷された小さな孤立領域と印刷されなかった小さな孤立領域の描画が減少することの詳細について、図3から図10を参照しつつ、ここで、説明する。

【0044】

図3は、図1のビットマスク生成コンピュータ1の生成モジュール8のブロック図である。

【0045】

この実施形態において、生成モジュール8は、ビットマスクのセットを表すデータの生成を調整するマスク生成モジュール40；後に詳細を述べる重み付け関数を表すデータを記憶するよう構成される重み付けマスク記憶部(weight mask store) 42；1および-1の間の範囲の浮動小数点数であって、かかる数は、アレイ内にランダムに配置され、1から-1間の範囲内でランダムに拡散する当該数が記憶されたアレイを含む乱数テーブル、44；現在生成されているビットマスクを表す0ならびに1のアレイを記憶するための1対の記憶部である現在のマスク記憶部46と稼動マスク記憶部47；生成されているビットマスクに関連する浮動小数点数のアレイ用の記憶部である現在の重み付けマップ記憶部 (weight map store) 48と稼動重み付けマップ記憶部49；生成中の現在ビットマスクの1対のアレイにおける座標を識別するデータ記憶部である位置外れリスト(out of position list) 50およびニュードットリスト52；ならびに、生成モジュール8により前もって生成されたグレースケールの複数のレベル用のビットマスクを表すデータを記憶するビットマスク記憶部54；を備えている。

【0046】

本実施形態において、マスク生成モジュール40は、各ビットマスクが32×32の0と1のアレイを備えた、256ビットのマスクのセットを生成するよう構成される。したがって、現在の稼動ビットマスク記憶部46と47はそれぞれ、32×32の2値アレイ、および、:浮動小数点数による32×32のアレイを有する現在の、また、稼動している重み付けマップ記憶部48と49ならびに乱数テーブル44、を記憶するよう構成される。まず、現在の、また稼動している重み付けマップ記憶部46と47、さらに現在の、また、稼動している重み付けマップ記憶部48と49における入力は、すべて0に設定される。ランダムに配置されたランダム浮動小数点数は、乱数テーブル44に前もって記憶されている。

【0047】

マスク生成モジュール40により生成された各ビットマスクは、アレイと関連する階調を示す、ドットの配置を表している。ある階調の平面的な画像を、ハーフトーン画像に変換するためビットマスクが用いられている場合、その領域を表す無地の色を表す、結果としてのトナーのパターンは、生成されたアレイにおける1の配置に対応する。外見上好ましい画像を生成するため、無地色の領域を表す画像中のドットは、均等に分散され、過度に集中しないことが好ましい。このような理由から、マスク生成モジュール40は、生成されたビットマスクアレイの1の位置が当該アレイにわたって拡散する、ビットマスクを生成するよう配置される。

【0048】

とはいえ、レーザープリンタ用のビットマスクの場合は、ビットマップが、出力画像で孤立した小さなトナー領域または小さな非印刷領域が発生するのを制限する性質をもつものであることもまた好ましい。というのは、レーザープリンタは、確実にこのような孤立

した小さなトナー領域や小さな非印刷領域を描画することができないからである。よって、本発明に基づけば、ビットマスクセットが、列を構成する0エントリーの過剰な凝固を避けようとするため、その結果、要件間に競合が起きてしまうのだが、本ビットマスク生成モジュールは、それらの要件間のバランスをとり、かつ、その一方でそれと同時に、孤立した1または0エントリーが含まれないようにすることによって、出力画像にある孤立した小さなトナー領域と孤立した小さな非印刷領域の数を減らそうとする。

【0049】

画像の中にドットの行（ライン）が生成される際、生成物を回避するために、生成されたパターンは何らかの方法でランダム化されることが望ましい。詳細は後述するが、これは、生成されたビットマスクアレイに存する1の位置をランダム数テーブル44にあるランダムな値に依存させることで実現させる。

10

【0050】

以上望ましいとされた結果を生むには、マスク生成モジュール40は、ビットマスクを生成することで、各グレースケールレベルを取得すべく、まずランダム数字テーブル44と重み付けマップ記憶部48に格納されているデータを使用する。これは修正のための候補となり得るものを1セット分確定するためである。詳細は後述するが、これらの候補は、生成されたビットマスクにもともとあった1との距離よりもずっと大きくなるように選択される。

【0051】

次に、修正のための各候補に関して、候補エントリーの修正が、孤立した1エントリーまたは孤立した0エントリーを発生させるかどうか、その結果それらがビットマスクに現れるかどうかの判断が下される。もし、孤立したエントリーを生成しない候補が発見されたら、現在のビットマスクの記憶部46のその位置に格納されているアレイに1がひとつ入力される。現在の重み付けマップ記憶部48の内部に格納されているデータは、次に、重み付けマスク記憶部42に格納されている重み付けファンクションデータを用いてアップデートされる。

20

【0052】

もし、候補となるエントリーがすべて0エントリーを、生成されるビットマスクに発現させるようであれば、マスク生成モジュール40は、修正の対象となる各候補のために、そのような結果の発生を回避するようなエントリークラスタを見つけ出す。そして、発見されたクラスタの一つが、現在のマスク記憶部46にあるビットマスクと現在の重み付けにある重み付けマップ記憶部48をアップデートするため使用される。

30

【0053】

このようにして、孤立した1のグループや0のグループがビットマスクに決して現れないようにし、かつ、同時に新たなビットマスク修正が広がるようにするための修正が、ビットマスクアレイ全体に広がるようにすべく、各グレーレベルへの修正がなされる。

【0054】

必要な数分の0が1に変換された時、マスク生成モジュール40は、作成されているレベルのために、ビットマスク上でスムージングオペレーション（平滑化動作）を行う。その際、現在の作業対象となっている重み付けマップ記憶部48と49における重み付けマップ、重み付けマスク記憶部42における重み付けマスク、乱数表44、アウトオブポジション50、新ドットリスト52が使用される。

40

【0055】

この平滑化動作は、複数の1がアレイを用いて好ましいグレースケールの画像を生成するような方法により分布されるよう、現在のビットマスクにおける1の分布を最適化するとともに、直前のビットマスク用のビットマスク内に現れる1の大部分が、現在のビットマスクにも確実に現れるようにする。その一方で直前のビットマスクのためにビットマスクに現れる0の大半もまた、必ず、現在のビットマスクに現れるようにし、1や0エントリーが孤立することするという現象を回避する。

【0056】

50

特定のグレーレベルについてかかる最適化が実行された後、現在のマスク記憶部46における現在のマスク記憶部4中の現在のビットマスクの複製が生成され、ビットマスク記憶部54内に記憶される。ビットマスク生成モジュール40は、次に、前のレベル用のビットマスクを用い、次のレベル用に新しいビットマスクを生成する。あるレベルから次のレベルへとデータを複製することにより、近接するグレーレベル用のドットの拡散は、同じとなり、したがって、輪郭削りを確実に減少させる。これにより、ビットマスク生成モジュール40はビットマスク記憶部54内に、ビットマスクのセットを表す 32×32 の2値アレイであって256セットを生成し、記憶させる。

【0057】

256のビットマスクセットが完全に生成され、記憶されると、次に、圧縮モジュール9が呼び出される。かかる圧縮モジュール9は、元のビットマスクデータの約10分の1のサイズの圧縮データを生成するため、記憶されたビットマスクを処理する。この圧縮済みデータは、プリンタドライバ生成コンピュータ2により生成されたプリンタドライバ25、30内に組み込むためにCDROM10上に記録される。

【0058】

ビットマスク生成コンピュータによる処理

ここで、ビットマスクデータを生成するビットマスク生成コンピュータ1の全体的な処理を、ビットマスク生成コンピュータ1の処理のフロー図である図4を参照して詳細に説明する。

【0059】

(i) ビットマスクの生成

まず、ビットマスク生成コンピュータ1は、マスク生成モジュール40を呼び出す。かかるマスク生成モジュール40が、初めて呼び出されると、当該マスク生成モジュール40は、重み付けマスクデータを重み付けマスク記憶部42に記憶させる(S4-1)。かかる重み付けマスクデータは、生成されるビットマスク内で1の拡散を可能にする関数を表す。この目的を達するため、マスク生成モジュール40は、ビットマスクアレイにおける各位置のため、他の複数の1に対する当該位置の相対的な近似性(relative closeness)を表す値が演算可能となるよう、データそ記憶する。具体的には、本実施形態において、以下の距離関数が用いられる。

【0060】

【数1】

$$D(x,y) = \frac{1}{\sqrt{\hat{x}^2 + \hat{y}^2}} \text{ for } |\hat{x}^2 + \hat{y}^2| < 60$$

$$= 0 \text{ for } |\hat{x}^2 + \hat{y}^2| \geq 60$$

$$= 100 \text{ for } \hat{x} = \hat{y} = 0$$

【0061】

ここで、 ∂x および ∂y は、アレイ内の2点におけるx座標とy座標とのそれぞれの差分から決定される。

その際以下の等式が用いられる。

【0062】

【数2】

$$\hat{x} = [|x_1 - x_2| - 1] * \lambda_x + 1$$

$$\hat{y} = [|y_1 - y_2| - 1] * \lambda_y + 1$$

【0063】

x_1, y_1, x_2, y_2 が、アレイにおける2点のための座標であり、 λ_x と λ_y が、画像における実際の距離（とはいえこれは出力される画像におけるピクセルの最小寸法のことだが）をスケールリングするための要素である。よって、プリンタ用のビットマスクにおいて、トナーの出力領域の寸法がxとyで等しい場合、 λ_x と λ_y は双方とも、1と等しくなり、等式

は単に二つの座標間の距離を表すものとなる。反対に、x方向のピクセルサイズがy方向のピクセルサイズの半分となるような形でビットマスクが使われている場合、 $\lambda x = 1/2$ と $\lambda y = 1$ であるので、等式は以下のようになる。

【0064】

【数3】

$$\delta x = \frac{1}{2} |x_1 - x_2| + \frac{1}{2}$$

$$\delta y = |y_1 - y_2|$$

10

【0065】

対になった様々な整数値の距離関数のために計算された値は、重み付けマスク記憶部42の内部に蓄えられる。

【0066】

図5は、 λx 、 λy が双方とも1であることを示す距離関数を使って計算を行い、データアレイを重み付け記憶部42内部に蓄えた例を示している。これらの数値計算や、数値の重み付けマスク記憶部42への保存を示すことにより、後説明がなされる通り、0エントリーがビットマスクにどのように広がっているかを示す重み付けマップの生成は、非常に俊敏に決定される。

【0067】

20

距離関数データが、重み付けマスク記憶部42内部に保存された後、マスク生成モジュール40は、現在のビットマスクにおいて必要とされている0の数が、1に変換されたかどうかを調べ、確定する(S4-2)。必要な数は、通常のテクニックを用いて確定される。すなわち、ビットマスクのワンセット分に含まれる1エントリーの数を増やし、結果として出てくる印刷出力が、徐々に濃度が減少する濃淡色として表現されるようにする。

【0068】

もし、必要な追加0数が1に変換されていなかったら、マスク生成モジュールは現在のビットマスク記憶部46に保存されているアレイ内の0エントリーを数多く選択するという作業(S4-3)を開始する。

【0069】

30

図6のフロー図（マスク生成モジュール40の処理フロー図）にさらなる詳細を示すように、マスク生成モジュール40は、まず、数配列の最小値を特定するため、重み付け現在のマップ記憶部48に記憶された重み付けマップを検索する(S6-1)。

【0070】

本実施形態において、重み付けマップは、まず、0による 32×32 のアレイを有しているので、ここでの最小値は0に等しい。

【0071】

次に、マスク生成モジュール40は、しきい値パーセンテージより小さく、重み付けマップにおける入力の最小値を越える値に関連づけられている、現在の重み付けマップ記憶部48にある重み付けマップにおける位置の座標を特定する(S6-2)。本実施形態において、前記しきい値は、特定された最小値の2%に設定されている。

40

【0072】

これらの座標は、修正の候補リストとして、関連重み付けマップ数値の昇べき順に並び替えられ保存される。2つ以上のエントリーが同じ重み付けマップ値に関連付けられている場合、それらのエントリーは、乱数表44に保存されている確認済み座標のために、数値の昇べき順に並べられる。

【0073】

並び替えの済んだ修正用候補リストが一旦生成されると、マスク生成モジュールは(S6-3)、リストの中にある最初の座標セットを選択し、それらの座標によって確定された0エントリーを変えることが、現在のマスク記憶部46に保存されているアレイ内の既存0エン

50

トリークラスタを拡張する作用をもつかどうかを調べる。

【0074】

本実施形態において、より具体的には、マスク生成モジュール40は、まず現在のマスク記憶部46にあるアレイのエントリーをテストし、処理された座標によって確認されたエントリーを変更することが、アレイ内の1のエントリーからなる新クラスタの生成につながるかどうかを判断する。

【0075】

図7は、本発明の実施形態に対応する実行可能な12のクラスタパターンを示している。本実施形態においては、マスク生成モジュール40は、まず、処理された座標によって確定された0エントリーを変えることが、以下の説明に符号するタイプの1エントリーのクラスタ生成を導くかどうかをチェックする。(i) 網掛けが施された正方形は、考慮の対象となっているエントリーを表している。よって、(i)の形状において、座標(x,y)が変更のための考慮の対象となっている0エントリーである場合、マスク生成モジュール40はまた、(x, y-1) と (x+1,y)に関連付けられているエントリーが双方とも1という値と関連付けられているという事態にあるかどうかを確定しようとする。もし、隣り合ったエントリーがチェックを行うために存在していないという状態を作るために、選択された0エントリーがアレイのエッジ部分に置かれている場合、マスク生成モジュール40は、逆側のエッジにある確定位置に存するエントリーをその代わりにチェックする。

【0076】

処理された座標によって確定された0エントリーを変えることにより、1エントリーのクラスタが図7で示された形状で生成されるか否かが検証された後、(i) もし、生成されていなかったということならば、図7に示されている次のパターンがテストされる。このような仕方では、(x,y)における0エントリーの場合は、マスク生成モジュールは、(x+1,y)と(x, y+1)がともに0エントリーか否かを確定しようとするのである。このテストは、考慮の対象となっている0エントリーの変更が、1エントリーのクラスタ生成を導いていると確認されるか、または、その反対に、0エントリーを変えても1エントリーのクラスタ生成は導かれなことが確認されるまで、繰り返される。

【0077】

処理された座標によって確定された0エントリーを変更することが、図7で示したような新しいタイプの1エントリークラスタを導かないことが判明したら、マスク生成モジュール40は、処理された座標によって確定された0エントリーを変更することが、既存の1エントリークラスタの拡張をもたらすかどうかの判断を行う。

【0078】

本実施形態において、上述の判断は以下の方法でなされる。すなわち、座標(x,y)を持つ変更の候補となる0エントリーの場合、マスク生成モジュール40は、(x+1,y) (x1,y) (x, y+1) (x, y-1)のいずれかの座標が、現在のマスク記憶部46に格納されているアレイに含まれるひとつの1エントリーと関連付けられてかどうかをチェックする。また、上述の方法で、これらの確定された1エントリーが図7で示されたタイプの1エントリークラスタの一部を形成するか否かをチェックする。そしてやはり、隣り合ったエントリーがチェックを行うために存在していないという状態を作るために、確定されたエントリーが一つでもアレイのエッジ部分に置かれている場合、マスク生成モジュール40は、逆側のエッジにある確定位置に存するエントリーをその代わりにチェックする。

【0079】

もし、マスク生成モジュール40(S6-4)が、処理された座標の候補によって確定された0エントリーに変更を加えることが、現在のマスク記憶部46にあるアレイに存する0エントリークラスタ生成を引き起こすことにつながらないと判断した場合、あるいは、その反対に、アレイ内にある既存の1エントリークラスタを拡張することにつながらないと判断した場合、マスク生成モジュール40は(S6-5)、生成された候補リストにある候補座標のすべてが処理されたか否かの判断を下す。すべてが処理されていないのであれば、候補リストにある次のエントリーが選択され(S6-6)、その座標が既存の1エントリークラスタに隣接

10

20

30

40

50

しているかどうかを見るためのテストが行われる。(S6-4)

【0080】

修正のための候補がテストされた時、その修正のための候補が1エンタリーの新クラスタ生成を導くか、または、候補が既存の1エンタリークラスタに隣接していたら(S6-4)、マスク生成モジュール40は(S6-7)、処理された座標候補と関連付けられた現在のマスク記憶部46にある0エンタリーをアップデートするという作業、そして、現在の重み付けマップ記憶部48内に格納されているデータをアップデートするという作業を始める。

【0081】

具体的には、マスク生成モジュール40は、処理されている座標によって確定された現在のマスク記憶部46にある0エンタリーを修正し、そして、現在の重み付けマップ記憶部48にある重み付けマップに内の各エンタリーを順に取り上げ、アップデートされるグリッド上の点の相対的な位置のために、xの距離とyの距離と1に変えられたビットマスク内にある0の位置を決定する。その際、以下の等式が使われる。

【0082】

【数4】

$$x_{dist} = \min[abs(X_1 - X_2), w - abs(X_1 - X_2)]$$

$$y_{dist} = \min[abs(Y_1 - Y_2), h - abs(Y_1 - Y_2)]$$

【0083】

ここで、X1、Y1は、1に変更されていた、現在のビットマスク46内の0の座標でありX2、Y2は、更新されようとする重み付けマップ48におけるアレイ内の位置の座標であり、wおよびhは、本実施形態では、アレイの幅と高さに対応する、そのいずれもが32に等しい値である。

【0084】

算出されたxの値を自乗し、同様に算出されたyの値を自乗し、両者を加算した値が、本実施形態においては49に設定されているところの閾値に満たなければ、マスク生成モジュール40は、当該xおよびyの距離を用い、重み付けマップ記憶部42内で、ある値を探し、以下の式を用いて、現在の重み付けマップ記憶部48における重み付けマップ内の識別された位置のための入力を更新する。

【0085】

【数5】

$$W_{new} = W_{old} + weight\ mask\ value * [1 + f(level) * r(x, y)]$$

【0086】

ここで、W_{new}は、重み付けマップ記憶部48内のアレイの配置に記憶された新たな値であり、W_{old}は、重み付けマップ記憶部48内のアレイに記憶された前の値であり、Weightmask valueは、重み付けマスク記憶部42を用いて取り込まれた値でありf(level)は、これに対しビットマスクが生成されるグレーレベルに基づいて変更される0と1間に存する関数であり、r(x, y)は、更新されている重み付けマップ48内の位置に対する、乱数テーブル44内の対応する場所に記憶された乱数の値である。

【0087】

図8は、本発明の本実施形態において用いられる関数f(レベル)を示すグラフである。本実施形態においては、最初に生成された64のビットマスク、ならびに、生成されているビットマスクのセット内の128番目のビットマスクの最大値を1に増加させる関数f(レベル)を有するビットマスクのセット内の少なくとも64のビットマスク用に、f(レベル)用の0と等しい値が用いられる。本実施形態において、関数f(レベル)は変形し、シフトした正規曲線であり、値が70から100において急に増加し、値が130から160で急に下落する形式である。

【0088】

乱数テーブル44内の1から-1の乱数により決定される重み付け値の生成は、重み付けマ

ップ48内に記憶された重み付け値にノイズ成分を取り込むことになる。充たされた0の4分の1から充たされた0の4分の3の間(between quarter full of 0's and three quarter full of 0's)でビットマスクが生成されている場合、距離関数のみに基づいて決定される1と0の拡散によって、一連のラインやコラム等の規則正しく並べられた形式のアレイとなるビットマスクを生じさせる。かかる規則正しいドット配列は、出力画像内の副産物のように見えるという傾向がある。この方法でノイズ成分を取り込むことにより、前記の規則正しく並べられたアレイの生成は、阻止される。乱数テーブル44内に記憶された値を用いることでこれを達成しようとする、0から1へ変更されるのに適切である位置の識別が繰り返し行われる。

【0089】

10

図9Aと9Bは数列の抽出物において変更された0エンタリーを示す一例であり、F(level)をゼロに設定して、図5で示された重み付けマスクを使いつつ重み付けマップをアップデートした結果発生するビットマスクの一部と、対応する重み付けマップエンタリーの関連数値増加を表現している。

【0090】

図9Aと9Bの図を用いた図解で分かるように、上で詳細を説明した方法で重み付けマップをアップデートすることは、変更の対象となった0の近くに存する座標用の重み付けマップエンタリーを増やすことに他ならない。重み付けマップエンタリーの数値を増やすと、その結果、現在のマスク記憶部46にあるビットマップ内の1エンタリーを示す座標の近くにある座標と関連するエンタリーが、より高い重み付けマップ値に関連付けられることになる。このようにして、最も低い重み付けマップ値に関連付けられた0エンタリーを、変更の候補として選ぶことによって、マスク生成モジュール40は、1エンタリーがビットマスク全体に広がる形で、ビットマスクを生成する。同時に、新1エンタリークラスタを生成するものに限って変更を実施するか、または、既存の1エンタリーのクラスタを拡張するものに限って変更を実施することにより、マスクモジュール生成40は、生成されたモジュールが孤立した小さなトナー領域によって表現されるはずのハーフトーンイメージ領域の出現を確実に削減する。

20

【0091】

重み付けマップがアップデートされた後(S6-7)、マスク生成モジュールは、現在のビットマスクが、ビットマスクのマスクの変更の原因になったかどうかの判断を下す。これは、孤立した0エンタリーがあるために、必ずしも確実に印刷されるとは言えない孤立した「ホール」の存在を確認するためである。さらに具体的に言うと、マスク生成モジュール40は、現在のマスク記憶部46に格納されているアレイにおいてアップデートされたばかりのエンタリーに隣接する座標用各8ビットマスクエンタリーをチェックする。このように、座標(x,y)に関連付けられている値を修正する場合は、以下の各座標が考慮の対象となる。

30

【0092】

$(x-1,y-1), (x-1,y), (x-1,y+1), (x,y-1),$
 $(x,y+1), (x+1,y-1), (x+1,y), (x+1,y+1)$

アレイのエッジにある変更されたエンタリーの場合は、やはり、既存のエンタリーを確定しない座標の代わりに、対応するポジションにあり、かつ反対側のエッジにあるエンタリーが考慮の対象となる。これらの座標に関連付けられたビットマスク内のエンタリーのいずれかが0とイコールである場合は必ず、マスク生成モジュール40は、チェックを行い、これらの0の値が、より大きな隣接した実現可能0値グループの一部であることを確認する。

40

【0093】

0エンタリークラスタに対する変更が、現在のマスク記憶部46におけるアレイ内の1エンタリークラスタを生成または拡張するか否かを判断すべく、候補である0エンタリーがテストされるのと類似した方法で、マスク生成40は隣接した各座標を順番にテストする。

$(x-1,y-1), (x-1,y), (x-1,y+1), (x,y-1), (x,y+1),$

50

$(x+1, y-1), (x+1, y), (x+1, y+1)$

【0094】

まず、一つの隣接座標が、現在のマスク記憶部46にあるマスク内の0エンタリーを発見できるかどうかを見る。もし、発見されないのであれば、この座標に対するさらなるテストは不要となる。

【0095】

もし、隣接座標が0エンタリーを発見したならば、「発見された0エンタリーが、より大きい0エンタリークラスタの一部であるか否か？」という観点から隣接座標がテストされる。この手順は上述の(S6-4)と同じである。すなわち、図7の(i)で示されている考慮の対象となる0エンタリー部分をあらわす網掛け正方形は、0エンタリーのパターンを表しているが、0エンタリーがこのパターンの一部かどうかを、まずテストするのである。もし、一部でないのであれば、図7で示されている次のパターンが、考慮の対象となっている0エンタリーが0エンタリークラスタの一部を形成していると確認されるまで、または、その反対に、0エンタリーは、図7で示されたクラスタタイプの一部ではないと確認されるまで、テストが続行する。

【0096】

最後に、もし、0エンタリーが図7で示されているどの形状の0エンタリークラスタの一部を形成していないと確認されたならば、マスク生成モジュール40は、テスト対象となっている0エンタリーに隣接している座標と関連している0エンタリーをテストし、これらの座標が、0エンタリークラスタの一部を形成する0エンタリーを発見できるか否かを調べる。このようにして、マスク変更モジュール40は、変更されたエンタリーに隣接する0エンタリー（1エンタリーになるように変更が加えられた0エンタリー）が、0エンタリークラスタの一部なのか、逆に、0エンタリークラスタに隣接するのかを確認する。

【0097】

変更されたばかりの0エンタリーに隣接する座標をすべてテストした後、もし、これらのエンタリーがすでに1の値であること、またはその逆に、0の値であること（これらの値はすべてより大きな0値クラスタの一部を形成する）が確認されたならば、候補となった変更は、孤立した0値のクラスタや極度に小さい0値のクラスタ生成を引き起こさないことになる。実際そうだったとしよう。そうすると、マスク生成モジュール40は、変更のためのエンタリー候補探しを止める。そして、この場合、マスク生成モジュール40は、格納されたビットマスクと、現在のマスク46と現在の重み付けマップ48記憶部はアップデートしてあるので(S4-2)、生成されたレベルでの要求変更数に到達したか、それ以上を成し遂げたかをチェックする。

【0098】

ところが、もし(S6-8)、候補となっているエンタリーを0から1へ変更することによって、小さな0エンタリーの孤立グループが発生する判断されたとした場合、マスク生成モジュール40は、リセットを行う(S6-9)。すなわち、現在の記憶部46にあるエンタリー候補を0にリセットし、現在の重み付けマップ記憶部48にある重み付け値をアップデートし、重み付けマップ値の前回の変更をリバースさせるのである。

【0099】

具体的には、新たに0が現在のマスク記憶部46にある現在のマスクに追加される時、重み付けマスク生成モジュール40が、重み付けマップ記憶部48にある重み付けマップの諸エンタリーの諸数値をアップデートするのと類似した方法で、マスク生成モジュール40は、以下の等式を用いて重み付けマップにおけるエンタリーをアップデートすることによって、現在選択されている1を0から1に変更するという効果を、重み付けマップから取り除く。

【0100】

【数6】

$$W_{new} = W_{old} - \text{weight mask value} * [1 + f(\text{level}) * r(x, y)]$$

10

20

30

40

50

【0101】

ここで、 W_{new} 、 W_{old} 、重み付けマスク値、 $f(level)$ 、 $r(x,y)$ は、前述の方法で数値計算がなされる。

【0102】

マスク生成モジュール40は、次に(S6-5)、変更候補リストから最終変更候補を選出するに至ったかどうかを確認し、もしも、まだそれに至っていないのであれば、候補が変更に適しているかどうかに関する決断を下す(S6-4) 前に、次の変更候補選出を行う(S6-6)。

【0103】

このようにして、マスク生成モジュール40は、0エントリーが1つだけ関係する候補者の適当な修正が確認されるか、または、提案されたエントリー候補を変更することにより、ビットマスクにある0エントリーの小さなグループが孤立した0エントリーが生成されるまで、変更候補と確認されたものをそれぞれ順番に処理していく。

10

【0104】

(S6-5)もしも、候補リスト全体が処理され、各候補を処理した結果、孤立した1エントリーまたは0エントリーの小さなグループが生成されたと確認されたならば、マスク生成モジュール40は、図6Bを参考にしながら以下説明する通り、現在のビットマスクにおける複数の0エントリーの変更に関係する代替変更選択へと進む。

【0105】

まず、最初に(S6-10)、マスク生成モジュール40は、「現在のマスク記憶部46にある現在のマスク」のコピーと「現在の重み付けマップ記憶部48にある現在の重み付けのマップのコピー」を作成し、それぞれ対応するコピーを作業対象のマスク記憶部47と作業対象の重み付けマップ記憶部49に置く。

20

【0106】

次にマスク生成モジュールは、最初の修正候補を選択し、そのエントリーによって生成されたアレイの中に孤立した1エントリーをひとつ生成ができたことを理由に、その修正候補が前回却下されたか否かを判断する(S6-11)。

【0107】

その判断が「否」であり、かつ、候補となっている0エントリーの修正によって、0エントリーの孤立したグループが生成されたという理由で、修正候補が却下されたのであれば、マスク生成モジュール40は(S6-12)、1に等しい0エントリーを設定することによって、「作業対象マスク記憶部47に保存されているビットマスクで現在処理されている座標候補」によって確定された0エントリーを修正する。そして、修正されたエントリーの座標は、チェックのために、座標リストに追加されることとなる。既に説明した通り、作業対象重み付けマップ記憶部49にある重み付けマップは、ステップ(S6-7)で記したのと同じ等式を使って、作業対象マスク記憶部47にあるビットマスクの修正の説明を行うべく、アップデートされる。

30

【0108】

もし、マスク生成モジュール40が、「エントリーの修正が、ビットマスクにおいて孤立した0エントリーの生成を引き起こした」と判断したならば、マスク生成モジュール40は、(S6-13)、「修正候補を含んだ0エントリークラスタ」を修正することにより、作業対象マスク記憶部47に格納されているビットマスクを修正する。

40

【0109】

より具体的には、マスク生成モジュール40は、0エントリーを1に設定することにより、処理される候補座標に関連付けられている0エントリーを最初に修正する。そして、既述の方法で、作業対象重み付けマップ記憶部49にある重み付けマップは、ステップ(S6-79)に関連して既に説明した等式を使って、作業対象ビットマスク記憶部47にあるビットマスクの修正に関する説明を行うべく、アップデートされる。

【0110】

それから、ビットマスク生成モジュール40は図7に描かれている潜在的クラスタ形状をそれぞれ順に吟味し、作業対象重み付けマップ記憶部49にある重み付け数値の合計が最小

50

となるものクラスタ形状を確定する。

【0111】

例えば、修正候補エントリーの座標が(x,y)で、図7でクラスタ(i)として描かれている形状のクラスタを生成するための重み付け数値が吟味されるところ。この場合、マスク生成モジュール40は、座標(x,y-1)と(x+1,y)に関連付けられた作業対象重み付けマップ記憶部49にある重み付け数値エントリーの合計値を確認する。また、アレイのエッジ部にある候補エントリーに関してであるが、これはやはり、アレイの逆のエッジ部にある適当な座標が使われる。

【0112】

最軽量値と関連付けられているクラスタ形状が確定した時、座標のために確定したクラスタ形状の他のメンバーを構成する0エントリーは、作業対象マスク記憶部47に格納されているビットマスクにおいて「1に設定する」という変更がなされる。ステップ(S6-7)において既述の通り、作業対象重み付けマップ記憶部49にある重み付けマップは、最軽量値のクラスタメンバー2つの座標を使って、アップデートされる。最後に、現在吟味の対象となっている候補座標と、クラスタメンバーの2つの座標が、チェックのために、エントリーリストに追加される。

【0113】

この段階で、生成モジュール8は、作業対象マスク記憶部47において、現在のマスク記憶部46にある対応するエントリーと異なるエントリーのすべてを含むリストを格納しており、また、作業対象マスク記憶部47にある0エントリーは、マスク内に現れる孤立した1エントリーが存在しないものとなっていることになる。最後に、作業対象重み付けマップ49における諸数値は、距離関数（「作業対象マスク記憶部47に格納されているアレイ内ポイント」が、「作業対象マスク記憶部47に格納されているビットマスクアレイにおいて発生する1エントリー」から、どれだけ離れているかを示す関数）を定義する。

【0114】

マスク生成モジュール40は、次のステップとして、(S6-14)座標リストにある最初の座標を選択し、今しがた修正されたばかりのエントリーはどれかを確認し、そのエントリーを変更することが、作業対象マスク記憶部47に格納されているビットマスクにある孤立した0エントリーを生成することにつながるかどうかを判断する(S6-15)。これは、より具体的には、あるエントリーが座標(x,y)をもつべく変更された時、マスク生成モジュール40は、作業対象マスク記憶部47にあるエントリーの中から、変更の対象となったエントリーに隣接するエントリーを8つ順番にそれぞれテストし、ステップ(S6-8)で説明された通りの方法で、隣り合ったエントリーが0エントリーの孤立したクラスタの一部となっていないかどうかを判断する。そして、アレイのエッジ部にあるエントリーに関しては、アレイの逆のエッジにある対応座標がチェックされることとなる。

【0115】

「隣接する座標は、孤立した0クラスタの一部である」と確認されたら、マスク生成モジュール40は、0エントリーを1エントリーに設定することによって、作業対象のマスク記憶部47にある座標のために、0エントリーを変更する(S16-16)。それからマスク生成モジュール40は、ステップ(S6-7)との関連で既述の方法を用いて、作業対象重み付けマップ記憶部49に格納されている重み付けマップ内の重み付け数値をアップデートする。次に、マスク生成モジュール40は、作業対象マスク記憶部47にある新たに変更されたエントリーの座標を、チェックのために座標リストに追加する。

【0116】

必要に応じて、変更されたエントリーに隣接するすべてのエントリーがチェック/アップデートされた後、マスク生成モジュールは、変更によって、ビットマスクに孤立した0エントリーを発生したか否かを調べるために、(S6-17)変更されたエントリーリストにあるすべての変更エントリーがチェックされたかどうかを確認する。もし、この確認の結果が「否」であったならば、マスク生成モジュール40は(S6-15)、変更対象エントリーリストにある次の座標セットを選択し(S6-18)、(S6-15)その変更対象エントリーに隣接するエ

10

20

30

40

50

ントリーが孤立した0グループの一部を形成するか否かを確認する。

【0117】

このようにして、マスク生成モジュール40は、1エントリーと0エントリーの配列において孤立した0エントリーが存在しない状態を完成させるまで、ビットマスク記憶部47にあるアレイと作業対象重み付けマップ記憶部49にある重み付けマップを変更し続ける。

【0118】

それからマスク生成モジュール40は、(S6-19)現在処理されている変更候補を巻き込みながら、提案された変更のスコア計算を行う。この計算は、「変更対象エントリーの座標用の現在の重み付けマップ記憶部48にある重みマップエントリーの合計」と「変更エントリーを確定するのと同一座標用の作業対象重み付けマップ記憶部49から引き出した重み付けマップ数値の合計」間の差異を確認することによって実施される。それからこのデータは、変更提案対象エントリーの座標を示すリストと一緒に格納されることになる。

【0119】

それからマスク生成モジュール40は(S6-20)、処理が提案された変更候補の中の最後の候補に達したか否かを確認する。もし、この確認結果が「否」であったら、マスク生成モジュールは、現在のマスク記憶部46と現在の重み付けマップ記憶部48からそれぞれデータをコピーすることによって、作業対象マスク記憶部47と作業対象マスク記憶部重み付けマップ記憶部49にあるデータをリセットする。それからマスク生成モジュール40は、(S6-21)代替ビットマスクを生成すべく提案された変更候補を使う前に(S6-11)、換言すれば、提案された候補が1に設定され、マスク内に孤立した1エントリーグループや0エントリーグループが存在しなくなるようにビットマスクが変更される前に、変更対象となる可能性のあるエントリー候補リストを選択し、却下の理由が「その候補が、1エントリークラスタの一部を形成しなかったこと」であるか否かを確認する(S6-12-S-20)。

【0120】

最後の変更候補が処理されたと確認されたら(S6-20)、生成モジュール8は、生成された変更候補リストにある各候補用に以下のリストを生成する。すなわち、孤立した1エントリー/0エントリークラスタや提案された各変更に関連づけられた重み付けスコアの変更を一切含まない代替ビットマスク生成の0エントリー候補リストである。

【0121】

それからマスク生成モジュール40は(S6-22)、使用する変更として、「重み付けスコアの最小変化に関連付けられている現在のビットマスクにもたらす変更数が最少となるような修正」を選択する。しかしそのような条件に合う修正が2つ存在する可能性がある。すなわち、ビットマスクに対する変更数も同じで、それぞれの場合の重み付けスコアも同じである場合である。この場合、乱数表44における最小値と関連付けられた変更候補によって生成される変更が、実施の対象として選択される。

【0122】

実施すべき変更が決定したら、マスク生成モジュール40は、「選択された修正を示す座標リストが指定する現在のマスク記憶部46に存する0エントリー」をそれぞれ修正し、座標を一つずつ使いながら現在の重み付けマップ48にある値をアップデートする。その際、ステップ(S6-7)に関連して既述の等式が用いられる。

【0123】

こうしてマスク生成モジュール40は、現在のマスク記憶部46に格納されているビットマスクが、「重み付け価値の最小増加と関連付けられているような1と0の代替列」となると同時に、現在のマスク記憶部46にある1や0の列に孤立した1エントリーが含まれず、また、出力画像に信頼性の低い描画を導いてしまうような孤立した0エントリーが含まれないようにする。

【0124】

図4に戻ると、マスク生成モジュール40は(S4-2)、少なくとも要求されている数の0エントリーが、現在のビットマスク記憶部46に格納されているビットマスクにおいて変更されたかどうかを確認する。この確認結果が「否」であれば、マスク生成モジュール40は(S4-

3)は、図6Aと図6Bに関連して説明したような方法で、もう一度(S4-2)、現在のレベルにとって十分なビットマスク変更がなされたか否かをチェックする前に、0エンタリーを1つ、またはそれ以上変更する。

【0125】

ビットマスクアレイにおいて要求されている数の0が変更されたのであれば(S4-2)、マスク生成モジュールは、新たに計算されたビットマスク(S4-4)内で0のポジションを最適化させるという作業へと進む。これは、図10A-Cに関して後述するが、可能であれば、1エンタリーの広がり改善するためである。

【0126】

(ii) ビットマスクの変更

ここまで説明してきた通り、「現在のマスク記憶部46における1への変換対象となる0の選出」は、あるビットマスクアレイ内にある既存の1の配列に依存する機能に基礎を持っている。このことは、最初に1に変換された0は、変換の結果として、それ以降ビットマスクに発生する1の位置や後続のレベルに対して強い影響を及ぼすことを意味している。本実施形態における初期の変更の影響を減らすために、いったんあるレベル用の初期のビットマスクが生成されたら、初期ビットマスクアレイが、孤立した1エンタリーや0エンタリーのグループを生成することなく、アレイにおける1エンタリーの広がり改善できるか否かを確認すべく、平滑化オペレーション(S4-4)を行う。平滑化オペレーションに係る処理が以下、図10 A-Cを用いて説明される。

【0127】

「前レベルのためのビットマスクアレイにある0エンタリー」に対応する、「現在のレベルのためのアレイにおける1エンタリー座標」が新ドットリスト52で示されているが、あるレベルのためのビットマスク上で行われる平滑化オペレーションが行われる時、マスク生成モジュール40はまず(S10-1)、この新ドットリスト52にデータを格納する。マスク生成モジュール40は、次に、「現在のビットマスク記憶部46と現在の重み付けマップ記憶部49」にある「ビットマスクと重み付けマップ」を、作業対象マスク記憶部47と作業対象重み付けマップ49にそれぞれ格納し、最初の1が見つかるまで、作業対象マスク記憶部47に格納されているビットマスクアレイをスキャンする。

【0128】

次にマスク生成モジュール40は(S10-2)、アレイにおいて選択された1が、新ドットリスト52に示されている座標に対応しているか否かを確認する。もし、この確認結果が「否」であったならば、選択された1は、前レベルのためのビットマスクにおいては、0によって表現されているということである。もしも、アレイにおいて選択されている1の座標が、新ドットリスト52内に格納されていないとしたら、それは、アレイにある1もまた、前レベルのためのアレイにある1として表現されているということを示している。

【0129】

マスク生成モジュール40が、選択された1は、前のレベルのビットマスク内の1によっても表されると判断すると(S10-2)、次に、マスク生成モジュール40は、位置外れリスト50内に座標がいくつ含まれているかを判断し(S10-3)、これとしきい値を比較する。

【0130】

本実施形態において、最初の16ビットのマスクについて輪郭削りの影響を減少させるため、あるレベル用のアレイ内に1が存在する場合、次のレベル内にも存在させるようにする。アレイ内に1がほとんどなく、したがって、トナードットが印刷されていない場合には、1の配置を変更し、これにより、輪郭削りを識別できるよう印刷されるドット配置を増加させることができる。これとは逆に、ミッドレンジの影用のビットマスクアレイの場合のように、アレイ内により多くの1が見られ、より多くのトナードットが印刷される場合(ミッドレンジの濃淡を表すビットマスクアレイの場合等がこれに当てはまる)、これらのドットの比較的少ない数の位置を変更したことによる影響は、輪郭削りを識別できる程度に増加しないが、中間階調を表すドットのアレイの外観を向上させる。

【0131】

このように、本実施形態において、変更可能な前のレベル用のアレイ内の1の数を識別するために用いられるしきい値は、ビットマスクで表されるように、グレーレベルに基づいて変化する。具体的には、本実施形態において、最初の16レベルについては、前のアレイ内に表されていない1を変更することができ、第4まですべてのレベルについては、前のアレイからの1を移動させることができる。

【0132】

このように、例えば、マスク生成モジュール40が、最初の16レベルの1のビットマスクを処理しており、当該マスク生成モジュール40が、アレイから選択された1が前のアレイ内にも示されていることを識別する場合、マスク生成モジュール40は、前記アレイ内の当該1の位置を変更しようとししない。これとは逆に、生成されるビットマスクのレベルが、16より大きく、その位置が変更されている、前のアレイにおける1を表す現在のアレイにおける1の4つの位置より小さい場合、マスク生成モジュール40は、アレイにある孤立した1エントリーまたは0エントリーの生成を防ぎつつ、処理の対象となっている1エントリーの位置を変え、これによって、アレイにある1エントリーの広がり改善できるかどうかを判断する。(S10-4, S10-19)。

10

【0133】

それからマスク生成モジュール40は、作業対象のビットマスク記憶部47のビットマスク内にある選択された1エントリーを0に変えるという修正作業を行う。それから、マスクモジュール40は、前記アレイから選択された現在の1が存在することの影響を取り除くため、作業対象の重み付けマップ記憶部49内の重み付けマップを更新する(S10-5)。その際と

20

【0134】

次に、マスク生成モジュール40は(S10-5)、選択された1エントリーの変更が、「選択されたエントリーに隣接するエントリーが作業対象マスク記憶部47に格納されているビットマスクアレイにある実行可能な1エントリークラスタの一部ではなくなってしまったという結果」をもたらしていないかどうかを確認する。

【0135】

この確認は、具体的にはマスク生成モジュール40の行う以下のような作業によってなされる。すなわち、マスク生成モジュール40は、「アップデートされたばかりのエントリーに隣接する座標用作業対象マスク記憶部47」にある8つのエントリーをそれぞれテストする。アレイのエッジにあるエントリーに関しては既に述べたが、それに対応したアレイの逆エッジにあるエントリーがチェックの対象となる。これらの座標に関連付けられたビットマスクのエントリーがひとつでも「イコール1」であれば、マスク生成モジュール40は、引き続いて、これらの1エントリーが、隣接した1エントリーの大きなグループの一部であることを確認するためのチェックを行う。これは、これらの1エントリーが、図7で示されたクラスタの形状のどれかに対応しているエントリークラスタの一部を形成しているか否か、図7の形状のどれかに対応する1エントリークラスタの一部を形成するエントリーに隣接する1エントリーであるか否かを判断する。ステップ(S6-4)に関連して孤立1エントリー用テストについては既述であり、このテストは、そこで述べられたのと同様の方法で行われる。

30

40

【0136】

もし、隣接したエントリーの中で、実行可能な1エントリークラスタ内にはもはや存在しないと判断されたものがあつたならば、それらの座標は、「修正対象となった1エントリーを示す一連の座標」に追加され、次いで、その列に加わる次の座標が、上述の(S10-4 S10-5)で示された方法で処理されるべく、選択される。

【0137】

修正対象となったエントリーを示す座標の序列を処理する際に(S10-5)、処理の対象となった座標に隣接するエントリーがどれひとつとして、「0エントリーからなる極度に小さなクラスタの一部を形成する0エントリー」でなかったとしよう。その場合、マスク生成モジュール40は、「チェックのための座標用序列」の中でも最後に位置する座標が処理

50

されているか否かを(S10-8)をチェックする。もし、このチェックの結果が「否」であれば、次の座標セットが選択され(S10-9)、「作業対象重み付けマップ記憶部にある重み付けマップ47」と「作業対象重み付けマップ記憶部にある重み付けマップ」が選択された座標を用いてアップデートされる(S10-4)。ただ、このアップデートは、選択された座標に隣接するポイントが、作業対象マスク記憶部42におけるマスクにおいて、孤立した0値となっていないかどうかを確認するためのテスト(S10-5)が行われる前に実施される。

【0138】

このようにして、マスク生成モジュール40は、作業ビットマスク記憶部49のビットマスクに孤立した1エントリーが一切存在しなくなるまで、隣接した既存の1エントリーを0エントリーに設定することによって、作業対象マスク記憶部49にあるビットマスクを修正する。この修正作業は、処理の対象として現在は、最初に選択されるものとなっている1エントリーも含む。同時に、重み付けマップが、「作業対象マスク記憶部47に格納されているアップデートされたアレイにある1エントリーから、アレイ内のポイントがどれだけ離れているかを示す関数」を見つけることができるように、マスク生成モジュール40はまた、作業対象重みマップ記憶部49に格納されている重み付けマップをアップデートする。

【0139】

極度に小さい1エントリークラスタがひとつも存在しないようにするために、作業対象ビットマスク記憶部にあるマスクに対する修正作業を完了したら、マスク生成モジュール40は次に(S10-10)、順番に新たに修正された1エントリーをテスト対象として、「1エントリーを0エントリーに置き換える作業によって、孤立した0エントリーが発生しなかったか否か」を調べる。

【0140】

より具体的には、マスク生成モジュール40は、変更された0エントリーの座標を順にそれぞれ取り上げる。変更された各0エントリーの各座標のために、マスク生成モジュール40はまず、「0に設定された変更対象エントリー」が「ステップ(S6-8)に関連して既に説明したテスト方法と同じ様式で、図7に図示されたタイプの作業対象マスク記憶部49に格納されているビットマスクにある0エントリークラスタの一部を形成しているか否か」のチェックを行う。アレイのエッジにあるエントリーに関しては、やはり、アレイの逆側のエッジにあるエントリーがテストされる。もし、テスト対象となっている0エントリーが、「図7にある形状のひとつに対応した0エントリークラスタ」の一部を形成していないのであれば、テスト対象である0エントリーに隣接した0エントリーはチェックを受ける。ここでいうチェックとは、「既述の方法で0エントリークラスタの一部を形成するかどうかを確認する」ということである。

【0141】

提案された変更が、図7で示された1クラスタの一部を形成しないような「孤立した0エントリー」を発生させると判断されたら(S10-10)、あるいは、提案された変更が、そのようなクラスタの一部分を形成する0エントリーに隣接しない「孤立した0エントリー」を発生させると判断されたら、現在の変更候補の使用は破棄される。しかし、逆に、(S10-10)マスク生成モジュール40が、変更エントリー候補をすべてテストした際に、全変更エントリーがより大きな0エントリークラスタの一部を形成していることが判明しならば、これは、作業対象マスク記憶部47に格納されているビットマスクには、孤立した1エントリーや0エントリーが一つもないことの立証となる。それから、作業対象マスク記憶部47に格納されているアレイは、「代替1, 0配列」が見つかるか否か、すなわち、ビットマスクを通して1の広がり改善されているかどうかを検証すべく、使用される。

【0142】

より具体的には、ビットマスク生成モジュール40は(S10-11)は、取っ掛かりとして、作業対象マスク記憶部にあるビットマスクの1つまたはそれ以上のエントリーに変更をかけることにより、ビットマスク記憶部47に格納されているビットマスクを用い、1つまたはそれ以上のエントリーをアレイの中に再挿入する。その際、図6Aと図6Bに関して説明をした既述の方法で、孤立した1や0のエントリーの生成を避ける一方で、1の広がりを最大化

するアルゴリズムが用いられる。

【0143】

マスク生成モジュール40は、次に(S10-12)、作業対象マスク記憶部にあるアレイの1エントリー数が、現在のマスク記憶部46に格納されているアレイの1エントリー数に等しいか、または、上回っているかを確認する。もし確認の結果が「否」である場合、マスク生成モジュール40は、(S10-11)、作業対象マスク記憶部47に格納されているマスクにある1エントリーの総数が、現在のマスク記憶部46に格納されているマスクにある1エントリーの数とイコールなのか、それとも上回るのかを再チェックする前に、作業対象マスク記憶部47に存在するアレイにある1つまたはそれ以上の0エントリーを修正する。

【0144】

作業対象マスク記憶部47に格納されているマスクにある1エントリーの総数が、現在のマスク記憶部46に格納されているマスクにある1エントリーの数とイコールかまたはそれを上回るのかが判明したら、マスク生成モジュール40は(S10-13)、両アレイにある1エントリーの総数が等しいために、「修正候補を用いて、1エントリーの広がり改善しようという試み」は放棄されたのか否かをチェックする。

【0145】

しかしながら、もし、1エントリーの再挿入によって(S10-3)、現在のマスク記憶部46にあるマスクと同じ数の1エントリーをもっている作業対象マスク記憶部47内マスクが生成されていたと判断されたら、マスク生成モジュール40は、(S10-4) いまや作業対象ビットマスク記憶部49に格納されているアレイにある0によって表現されている「前レベルにおけるビットマスクの1のエントリー」の数を調査し確認する。もしこの数が、ステップ(S10-3)の関連で既述の通り、アウトオブポジション扱いを許可された1エントリーの閾値を超えるのであれば、現在選択されている変更対象候補を用いて行おうとしていた変更は破棄される。

【0146】

もし、閾値を超えていないのであれば、この時点でマスク生成モジュール40は、双方のアレイが同じ数の1エントリーを含むところの現在のマスク記憶部46に格納されている1エントリーの配列と0エントリーの配列を、作業対象マスク記憶部47に格納するはずである。双方の数列に孤立した1や0が含まれないことも実証され、また、前レベル用ビットマスクのための1エントリー数（ここでいう1エントリーは現在のマスク記憶部46と作業対象マスク記憶部47に格納されているビットマスクには含まれていないもの）は、閾値よりも低くなっているはずである。最後に、現在の重み付けマップ記憶部48、49、作業対象重み付けマップ記憶部48、49においては、アレイ内の位置、すなわち、現在のマスク記憶部46、47と作業対象マスク記憶部46、46それぞれのアレイに存する1エントリーの相対的なポイント間距離を決める機能が反映されている。

【0147】

次に、マスク生成モジュール40は、作業対象マスク記憶部47に格納されている代替ビットマスク配列が、現在のビットマップ記憶部46に格納されているビットマスクの改良版となっているかどうかを判断する。この判断は具体的には以下のような計算に基づく(S10-15)。すなわち、作業対象マスク記憶部47に含まれるビットマスクに1エントリーが存しており、それらの1エントリーは、「現在のマスク記憶部46に存するビットマスク」を表現している。さて、この「現在のマスク記憶部46に存するビットマスク」には0エントリーが存しているが、それらの0エントリーを対象として、現在の/作業対象ビットマスク記憶部48、49において、それぞれ重み付け数値が定められている。これらの重み付け数値の差を見ることが、具体的な計算の一側面である。一方で、作業対象マスク記憶部47に0エントリーが存しており、それらの0エントリーは、「現在のマスク記憶部46に存するビットマスク」を表現している。さて、この「現在のマスク記憶部46に存するビットマスク」には1エントリーが存しているが、それらの1エントリーを対象として、現在の/作業対象ビットマスク記憶部48、49において、それぞれ重み付け数値が定められている。これらの重み付け数値の合計を見ることが、具体的な計算のもう一つの側面である。

10

20

30

40

50

【0148】

もし、(S10-16)作業対象重み付けマップ記憶部47、48/現在の重み付けマップ記憶部47、48に格納されている重み付けマップを対象としてこのような方法で計算を行った結果、算出された値が負であれば、このことは、「作業対象マスク記憶部47にあるビットマスクによって表されている配列中の1エントリーの広がり、現在のマスク記憶部46における配列よりも広がりがない」ということを示している。

【0149】

しかし、算出された値が正であったら(S10-16)、マスク生成モジュール40は(S10-17)、この算出された「配列のために関連付けられた重み付け数値上の差異」と共に、現在のマスク記憶部46にある座標の値とは異なる「作業対象マスク記憶部47にあるエントリーの座標」を格納する。

10

【0150】

「変更対象エントリーの座標」と「改良の程度を示す重み付け値の差異」が格納された後、または以下の理由から、修正候補が破棄された後、(理由1) 提案された変更が孤立した0エントリーを生成させる可能性が高い(S10-9) (理由2) 提案された1エントリーの代替配列が1エントリーの総数を増加させることになる(S10-13) (理由3) 変更された配列において、1エントリーの数が増加する(上述の1エントリーの極度の増加に関する説明において言及されている1エントリーは、前レベル(S10-14)用としてはビットマスク上の0エントリーとして表現される)、または提案された変更はビットマスク全体にわたって1エントリーの広がりにより改良をもたらしていないと判断された後(S10-16)、マスク生成モジュール40は(S10-18)、現在の重み付けマップ記憶部48と現在のマスク記憶部46とにあるエントリーを、それぞれ、作業対象重み付け記憶部47と作業対象重み付けマスク記憶部49にコピーすることによって、作業対象重み付けマップ記憶部47と作業対象重み付けマップ記憶部49の内容をリセットする。

20

【0151】

さらに、マスク生成モジュール40は(S10-19)、現在のマスク記憶部46に格納されているビットマスク配列のすべての1エントリーが変更の対象として吟味されたかどうかの確認を行う。もしも、この確認の結果が「否」であったら、現在のマスク記憶部46にあるビットマスク配列の次の1エントリーが選択され(S10-20)、マスク生成モジュール40は、上述の方法で、選択されたエントリー候補を巻き込みながら、現在のマスク記憶部にある1エントリーの広がり改善されたか否かを確認しようと試みる(S10-2-10-19)。

30

【0152】

最終的には、マスク生成モジュールは、現在のマスク記憶部に格納されているビットマスクにある最後の1エントリーに到達したかどうかを判断する(S10-19)。この時点で(S10-21)、マスク生成モジュール40は、ビットマスクにある1エントリーの広がり改善を加えることができるすべての1エントリーを対象として、変更によってどの程度の改善をもたらされたのかを示す重みスコア値とともに、改善をもたらすためには変更されねばならない現在のビットマスクにあるエントリーデータを格納しているはずである。

【0153】

もし、現在のビットマスク記憶部46にあるビットマスクアレイにある1エントリーすべてを処理したのに、改良されたデータが一切格納されていなかったら、このことは、マスク生成モジュール40は、現在のマスク記憶部46にあるビットマスクの1エントリー広がり、改善をもたらすことができなかったことを意味する。

40

【0154】

しかしながら、もしデータが格納されたのであれば、広がった1エントリーの最大の改善と関連付けられている変更は、座標によって確定されたいずれかの1エントリーを0エントリーに設定することによって、現在のマスク記憶部46にあるアレイをアップデートすべく(S10-22)、用いられる。それから、現在の重み付けマップ記憶部48にある重み付けマップは、ビットマスクに対する変更を反映するべく修正されるが、それは、「変更されたエントリーの座標を順番にとりあげ、これらの座標とある関数(ここでいうある関数とはS6

50

-7とS6-9のステップに関連して既述のものである)を用いながら、重み付けマップ値をアップデートすることによって」である。ここで当てはまる関数がS6-7なのか、S6-9であるのかは、エントリーが0から1に変えられたのか、1から0に変えられたのかによる。

【0155】

次に、マスク生成モジュール40は、ある1つの0エントリーが、直前レベルのビットマスクにある1つの1エントリーに対応しているところの現在のビットマスク記憶部46のビットマスクにあるエントリー座標を確定することによって、アウトオブポジションリスト50をアップデートする。

【0156】

(S10-23)それから、マスク生成モジュール40は、この時点まで生成されたビットマスクの処理にかかった時間を確認する。本実施形態において、変更の無限ループ発生を防ぐために、あるレベル用のビットマスク平滑化にかけてよい総時間が閾値によって制約されている。この閾値は、使用されているビットマスク生成コンピュータ1の処理能力処理スピードに基づきユーザーが設定する。もし、平滑化作業のタイムリミットに達していないのであれば、マスク生成モジュール40は、1の全体的な配置を改善すべく、アレイにある1の配列を変更しようとする(S10-02, S10-22)前に、ビットマスクアレイにある最初の1エントリーを選択する(S10-1)。

【0157】

もし現在のレベルを平滑化する作業のタイムリミットに達しているのであれば、あるいは、前記アレイにおける1の配列について全く改良が認められない場合、マスク生成モジュール40によるかかるレベル用のビットマスクにおける位置の最適化処理は、終了する。

【0158】

生成されたビットマスクの処理の結果、アレイ内で拡散した1を有するビットマスクが生成される。平滑化処理を行うので、新たな1が特定のレベル用のアレイに追加される順番は、関係ないが、すべての新しい1がアレイに追加された後に、適切な位置の選択は再評価される。

【0159】

アレイの処理は、ビットマスクアレイを使用することにより生じる輪郭削りを許容可能なレベルにすることもできる。隣接するグレーレベル用の出力パターンを比較すると、第1レベルにおける限られた数のドットしか、第2レベルにおいて表示されない。具体的には最初の16階調の場合、連続する各レベルにおいて、追加の4ドットが現れる。次の階調においては、連続する各レベルにおいて追加の4つのドットが現れ、最大4つの位置まで、レベル間で追加のドットを変更してもよい。ドットのほとんどが、連続するレベルにおいて同じ位置に現れるので、連続するグレーレベル用の出力におけるドットの配置は各出力におけるドットは、それら自身の間で均等に拡散し、大量の共通ドットを有することから、外観的に好ましいものでなければならない。

【0160】

最後にアレイ処理は、「レーザープリンタが安定的に印刷しない可能性のある極度に小さい1エントリーのグループや極度に小さい0エントリーのグループを含まないビットマスクが生成されていること」の確認も行う。

【0161】

図4を参照すると、マスク生成モジュール40による平滑化動作が完了した後、次に、マスク生成モジュール40は、(S4-5)ビットマスク記憶部54のビットマスクの複製を記憶する。マスク生成モジュール40は、次に、256の異なるグレーレベル用の256ビットのマスクが、この時点でビットマスク記憶部54に記憶されたどうかを判断する。記憶がなされていない場合、マスク生成モジュール40は、次のグレーレベル用に新しいビットマスクを演算する前に(S4-2からS4-5)位置はずれリスト50およびニュードットリスト52をクリアする(S4-7)。

【0162】

したがって、マスク生成モジュール40は、それぞれがビットマスク記憶部54内に記憶さ

10

20

30

40

50

れた256のマスクのセットを生成する。連続する各ビットマスクは、前のレベルのビットマスクおよび重み付けマップを用いて生成されるので、各レベルのビットマスクを生成するには、前のレベル1のビットマスクが考慮される。生成を行うためビットマスクの完全なセットが決定されると(S4-6)、次に、記憶されたビットマスクは、ここで、図11から13を参照して詳細が説明されるように圧縮のため圧縮モジュール9に送られる。

【0163】

(iii) 生成されたビットマスクの圧縮

図11は、本実施形態の圧縮モジュール9のブロック図である。本実施形態において、圧縮モジュール9は、生成モジュール3により用いられたビットマスク記憶部54と同じであるビットマスク記憶部54、ラインマスク記憶部56、排他的論理和プロセッサ58、排他的論理和ラインマスク記憶部60、ランレングス・エンコーダー62ならびに圧縮データ記憶部64、を備えている。

10

【0164】

本実施形態において、0と1で構成される、256の 32×32 のアレイ形式のビットマスクのセットがビットマスク記憶部54に記憶されている場合、圧縮モジュール9は、まず、ラインマスク記憶部56に記憶されたラインマスクのセットを生成して(S4-8)、最初に生成されたビットマスク、ラインマスク記憶部56に記憶されたラインマスクのセットを処理する。次に、圧縮モジュール9は、排他的論理和ラインマスク記憶部60に記憶される排他的論理和ラインマスクを生成するため、排他的論理和プロセッサ59を用い、ラインマスク記憶部56内の生成されたラインマスクを処理する。

20

【0165】

最後に、排他的論理和ラインマスク記憶部60内に記憶されたデータは圧縮データプリンタドライバ内に組み込むことができるよう、最初に圧縮データ記憶部64に記憶され、次に、CDROM10上に記録される(S4-10)圧縮データを生成するため(S4-9)、従来のランレングス・エンコーダー62を用いて処理される。

【0166】

ここで、圧縮モジュール9の処理について、図12および図13を参照しつつ詳述する。

【0167】

図12は、ビットマスク記憶部54、ラインマスク記憶部56、排他的論理和ラインマスク記憶部60、ならびに、圧縮データ記憶部64内に記憶されたデータを示している。

30

【0168】

本実施形態において、ビットマスク記憶部54内に記憶されたデータは、最初に、256の 32×32 ビットマスクアレイのセットを有している。これらのアレイのそれぞれは、第1レベルのビットがすべて0の入力から構成され、256番目のレベルのビットマスク50はすべて1から構成され、中間レベルは、これらのアレイにおいて、1の数が増加している 32×32 の2値アレイを有する。

【0169】

後で詳述するように、マルチレベル画像を、ハーフトーン画像に変換するためにビットマスクが用いられた場合、マルチレベル画像データは、ライン毎に処理される。したがって、圧縮モジュールは、まず、それぞれの独立したライン状データを処理するため、関連するビットマスク記憶部54内の異なるビットマスクからすべてのデータを集める。

40

【0170】

すなわち、圧縮モジュール9は、256番目のレベルのビットマスク用のビットマスクから、最初の32ビットのデータを、最初のラインマスク用の最初の32ビットデータとして複製する。圧縮モジュール9は、次に、255番目のレベルのビットマスク用の最初の32ビットデータを特定する。これは、ビットマスク記憶部における二番目の32ビットデータとして記憶される。この動作は、第1レベルのビットマスクからの最初の32ビットデータが、ラインマスク記憶部56における32ビットデータの256番目のグループとして記憶されるまで繰り返される。

【0171】

50

次に、圧縮モジュール9は、ビットマスク記憶部54内の256レベルのビットマスクから、32ビットデータの二番目のグループを複製する。このように、ラインマスク記憶部56は、それぞれが、256の32ビット数を含む32セットのデータであって、ラインマスクデータのm番目のセットにおける32ビット数のm番目が、ビットマスク記憶部54から(256-n)番目のビットマスクの番目のビットマスクのm番目の32ビット数に対応するものをそれ自身の中に記憶させる。

【0172】

かかる方法によりデータを再配置することは、そこで、各ビットマスクにおける0に対する1の比率が各レベルにおいて増加するビットマスク記憶部54内の256のビットマスクとは逆に、ラインマスク記憶部54におけるデータのセットは、そこで、セットにおける32ビットデータの各グループ0の数が、最初の32ビットゼロから徐々に増加し、32ビットの256番目のグループにおいて32の入力すべてが0となる、32セットのデータを有することを意味する。

【0173】

図13Aは、ラインマスク記憶部56内に記憶されたデータのある部分の例である。図13Aから見られるように、ラインマスク記憶部56内に記憶されたラインマスクにおいて、アレイにおける1の入力は、かかる再配置の結果、異なるグレーレベル用の異なるビットマスク間に同じ値1が複製される、コラムを構成する。特定のレベルに対して新しい1が追加されるとともに、連続するレベルを通じて続き、次に、1が平滑化プロセスにおいて変更されなかった場合、新しい1のコラムが始まる。特定の1の入力が、異なるビットマスクを生成することにより行われる、平滑化プロセスに移ると、1のコラムは終了する。

【0174】

ビットマスク記憶部54からのすべてのデータが、ラインマスク記憶部56内1に複製されると、圧縮モジュール9は、排他的論理和ラインマスク記憶部60内に記憶される排他的論理和ラインマスクデータを生成するため、ラインマスク記憶部56内のデータを用いるよう、排他的論理和プロセッサ58を呼び出す。

【0175】

具体的には、排他的論理和プロセッサ58は、まず、ラインマスク記憶部56から、表された最初の32ビット数を取り出し、その数について、32個の1を有する32ビット数を用いて排他的論理和動作を実行する。すなわち、排他的論理和プロセッサ58は、32個の1から構成される二番目の32ビット数を有する前記32ビットの32ビット数それぞれについて、以下の真理値表に基づき、ビットに関する(bit wise)排他的論理和動作を実行する。

【0176】

【表1】

Input A	Input B	Output
0	0	0
1	0	1
0	1	1
1	1	0

【0177】

次に、かかる排他的論理和の結果は、排他的論理和ラインマスク記憶部内に最初の32ビットデータとして記憶される。排他的論理和プロセッサ58は、ラインマスク記憶部56から次の32ビットデータを選択し、この32ビット数、ならびに、最初のビットマスクから最初の32ビットデータのセットについて排他的論理和動作を実行する。かかる排他的論理和動作の結果は、排他的論理和ラインマスク記憶部60内に二番目の32ビットデータとして記憶される。かかる動作は、二番目および三番目の32ビットデータの間続けられ、その後、最初のラインビットマスクから次の32ビット数の各ペアまでの間続けられる。同じ処理が、ラインマスク記憶部56内のラインビットマスクの次の各セットからのデータについて行

10

20

30

40

50

われる。

【0178】

排他的論理和プロセッサ58によるかかる処理の結果を、図13Bに示す。図13Aと図13Bを比較すると、排他的論理和プロセッサによる処理は、0、ならびに、1であって、それが、ラインマスク記憶部56に記憶されたデータ中の1のコラムがどこで始まり、あるいは終了するが示すものが、まばらに配された(sparsely filled)アレイを生成するものである。

【0179】

次に、圧縮モジュール9は、ランレングス・エンコーダー62を呼び出させる。かかるランレングス・エンコーダー62は、排他的論理和ラインマスク記憶部60内に記憶されたデータを、当該排他的論理和ラインマスク記憶部60内に記憶されたそれぞれの1を分離する0の数
10
を決定することにより、従来の方法で処理する。したがって、例えば、図13Cに示すように、図13Bにおける数のアレイは、図13Cに示す数のセットに変換される。かかるデータは次に、圧縮データ記憶部64に記憶される。

【0180】

ビットマスク形式から、ラインマスク記憶部56に記憶されたラインマスク形式へのデータの再配置は、ビットマスクを表すサイズを減少させない。しかし、排他的論理和動作の実行し、その後、ランレングス・エンコーディングを用いる動作の組み合わせは、記憶するデータの全体量を約10分の1(a factor of 10)に減少させる。これはほとんどの場合、あるレベルに存在する1は、次のレベルにおいて1によって表され、同様に、あるレベルに
20
存する0は、次のレベルにおいて0によって表されるからである。したがって、排他的論理和プロセッサによる処理結果は、まばらに配されたアレイとなり、これにより、ランレングス・エンコーディングによって大幅に圧縮されたアレイとなる。次に、圧縮データは、それが、プリンタドライバ25、30内に組み込まれるよう、プリンタドライバ生成コンピュータ2に送られるCDROM10上に記録される。

【0181】

プリンタドライバにおける圧縮済みビットマスクの使用

ここで、プリンタドライバ25、30において、本発明に基づき生成された圧縮データを使用することについて、図14から図20を参照しつつ説明する。

【0182】

図14は、本発明の本実施形態に基づくホストコンピュータ3のブロック図である。ホスト
30
コンピュータ3のメインメモリ内に記憶されているのは、ワードプロセッシングプログラム等の文書生成プログラム27およびプリンタドライバ25である。さらに、クライアントコンピュータは、データを処理するマイクロプロセッサ80、マイクロプロセッサによりアクセス可能なキャッシュメモリ82、ならびに、プロセッサ80により最近用いられたデータを記憶するページメモリ84をも備えている。

【0183】

本実施形態においてごプリンタドライバ25は、当該プリンタドライバに組み込まれたプリンタドライバ生成コンピュータ2により選択されたテキストドライバ11であるテキスト
40
プロセッサ86、ならびに、プリンタドライバ生成コンピュータ2により画像ドライバ13から選択された画像ドライバである画像プロセッサ88を備えている。

【0184】

また、画像プロセッサ88は、ビットマスク生成コンピュータ1によりCDROM10に記録され、プリンタドライバ生成コンピュータ2により画像データプロセッサ88に組み込まれたデータである圧縮ビットマスクデータ92を含んでいる。かかる画像プロセッサ88は、圧縮データ92からラインビットマスクのセットを生成する解凍モジュール94、解凍ラインマスク記憶部95、文書作成部27から受けた画像データを記憶する生画像データ記憶部96、ならびに、画像プロセッサ88により生成されたハーフトーンデータを記憶するハーフトーン画像データ記憶部98を備えている。さらに、プリンタドライバ25は、それにホストコンピュータ3が接続されているプリンタ28(図14には図示せず)用に現在選択されている構成を記憶する設定データ記憶部99を含んでいる。
50

【0185】

図15を参照すると、ハーフトーン画像を生成するためのホストコンピュータ3の動作がここで示されている。

【0186】

まず、ホストコンピュータ3は、印刷される文書を生成する(S15-1)ため、文書作成プログラム27を用いるかかる文書は、ワードプロセッシング、画像、あるいは、テキストおよび画像データの両方を含む、文書の形式により構成することもできる。

【0187】

文書ファイルの生成後、ホストコンピュータ3は、ユーザーが生成された文書を印刷する旨表明したかどうかを判断する(S15-2)。そうである場合、ホストコンピュータ3は、生成された文書ファイルをハーフトーン画像データ98に変換するため、プリンタドライバ25を呼び出す。

10

【0188】

具体的には、プリンタドライバ25は、印刷画像29を出力するため用いられるプリンタ28用に現在選択されている設定を識別するため、まず、設定データ99をチェックする(S15-3)。したがって、例えば、設定データ99は、用いられる特定の印刷密度を表してもよい。設定データ99を1チェックした後、プリンタドライバ25は、生データをハーフトーン画像に変換するために用いられる、圧縮ビットマスクデータ92を選択する。

【0189】

具体的には、特定のプリンタ設定用のビットマスクのセット対応する圧縮ビットマスクデータ92が、プリンタドライバ25により識別される。かかるデータは、ビットマスク生成コンピュータの圧縮モジュール9により前もって生成されたランレングスに符号化されたデータを含んでいる。

20

【0190】

ランレングスに符号化されたデータの一部の例を、図16Aに示す。次に、排他的論理和ビットマスクデータを生成するため、ランレングスに符号化されたデータを最初に用いたプリンタドライバ25の解凍モジュール94が、(S15-4)が呼び出される。

【0191】

これは、各ラインが、圧縮ラインマスク記憶部95に記憶された32の入力を含む、0と1のアレイを生成する解凍部94によって達成される。ランレングスに符号化されたデータの要素を順次取り出すことにより、解凍部は、生成されたアレイに、ランレングス符号化データの第1の数に対応する一連のゼロを組み込む。かかる数のゼロがアレイ内に組み込まれた後、かかるアレイの次の入力は、1に設定される。次に、ランレングス符号化データの次の要素が、処理される。ランレングス符号化データの次の要素に対応する複数の入力はアレイ内で再び0に設定され、その後、別の入力が1に設定される。

30

【0192】

図16Bは、図16Aに示すランレングス符号化データを用いて生成された32X32のアレイの一部の例である。

【0193】

圧縮ラインマスク記憶部95に記憶された生成データは、すべて0のデータから構成される32ビットの2値数を有する解凍データにおける最後の32ビットについて、排他的論理和動作を実行することにより、解凍部94によって処理される。かかる排他的論理和動作の結果は、記憶され、次に、かかる数、ならびに、アレイ内の32ビットの次のセットについて行われた排他的論理和動作の結果が記憶される。この動作は、32ビットの最初のセットが256の32ビット数の各グループに達するまで、連続する32ビットの間繰り返される。

40

【0194】

図16Bのアレイに対する、かかる処理の結果を、図16Cに示す。処理の効果は、ビットマスクが図12に示されたよう元々作られていた場合に、解凍ラインマスク記憶部95内の解凍データを、ビットマスク生成コンピュータ1の圧縮モジュール9のラインマスク記憶部56内に前から存在していたものとして、32のラインビットマスクのセットに変換することであ

50

る。すなわち、本実施形態において、解凍ラインマスク記憶部内に記憶されるのは、それぞれが256の32ビットマスク数を有する、32セットのリンクマスクであって、各セットの最初のラインは、すべて1から構成される32ビット数からなり、32ビット数の256番目はすべて0から構成され、中間の数は通常0の数が徐々に増加するものである。

【0195】

図15に戻ると、次に、プリンタドライバ25は、印刷される文書ファイルの一部である文書ファイルが、テキストデータあるいは画像データを含むかどうかを判断する(S15-5)。処理される文書の一部がテキストデータを含んでいる場合、テキストプロセッサモジュール86が呼び出され、従来の方法により、テキストデータを画像データに変換するため用いられる(S15-6)。

10

【0196】

それに代え、生成される文書ファイルの一部が、画像を表す場合には、画像プロセッサモジュール88が呼び出され、その詳細を後述するハーフトーンデータを生成するため用いられる(S15-7)。テキストプロセッサ86又は画像プロセッサ88のいずれかによりハーフトーンデータが生成されるとかかるデータは、生成されたハーフトーン画像内で1が表される場合、トナーのドットを記憶することによりハーフトーン画像を印刷するため当該ハーフトーンデータを用いるプリンタ28に送られる(S15-8)。

【0197】

生成されたラインマスクを用いてのマルチレベルデータの変換

ここで、解凍リンクマスクを用いた画像プロセッサ88による、マルチレベル画像データのハーフトーンデータへの変換について、図17から図20を参照して説明する。

20

【0198】

画像プロセッサ88が、最初に呼び出されると、画像プロセッサ88は、処理のため、生画像データ記憶部96から画像データのラインを抽出する(S17-1)。

【0199】

図18は、画像を表すマルチレベル画像データの一部の例である。図17の例において、マルチレベル画像データ154、153、153、154、153、154、20、15…を有する画像データの最初のラインは、画像プロセッサ88により抽出される。従来、これらの数は、0から255の範囲にあり、0は黒を表し、255は白を表す。

【0200】

30

次に、画像プロセッサ88は、ホストコンピュータ3のページメモリ84内に、リンクマスクをロードする(S17-2)。画像の最初のラインである場合、ページメモリ84内に、256の32ビット数であるラインマスクの最初のセットが、ロードされる。

【0201】

次に、マルチレベル画像データプロセッサのラインの、最初の画素が選択される(S17-3)。図18の例の場合、かかる画素は、図17のアレイ内の位置(1,1)、Dにおける154である。

【0202】

次に、画像プロセッサ88は、かかる数を直前に用いられていた値と比較する(S17-4)。かかる値が、前に用いられていた値と等しくない場合、画像プロセッサは、次に、n+1番目の32ビット数を、ページメモリ84からキャッシュメモリ82内に記憶させる(S17-5)。ここで、nは、考慮されるマルチレベル画素の値と等しい。このように、黒を表す0に設定されたマルチレベルデータを処理する場合、一連の1を有する最初の32ビット数が取り込まれる。

40

【0203】

図19は、キャッシュメモリ82内に記憶された32ビット数の例である。

【0204】

画像プロセッサ88は、キャッシュメモリ82内に記憶された32ビット数の入力の一つを識別するため、処理されるマルチレベルデータ用のx座標を用いる(S17-6)。より具体的には、プリンタドライバ25は、画素が処理される度に増やされ32画素のそれぞれが処理された後0にリセットされる、座標値を記憶する。これは、32を法とする数の算法(module 32 ar

50

ithmtic)における現在のx座標と等しい。

【0205】

次に、画像プロセッサ88は、キャッシュメモリ82内の32ビット数中の入力特定するため、32を法とする当該現在のx座標値を用いる。これは、0か1のどちらかであり、かかる値は、現在生成されているライン用のハーフトーン画像データ98に加えられる。

【0206】

次に、画像プロセッサ88は、処理される画像データのラインにおける最後の画素に達したかどうか判断する(S17-7)。達していない場合、前記ラインに沿った現在0位置用の現在のカウンタを、インクリメントし(S17-8)、ページメモリ84から32ビット数を選択し、それをキャッシュメモリ82に記憶し、次に、記憶された32ビット数を、ハーフトーンデータに変換するよう利用するため、マルチレベル画像データの次の要素を、特定し利用する(S17-4からS17-5)。

【0207】

処理されているラインのマルチレベル画素データの最後の入力に達すると、かかるライン用のハーフトーンデータ98は、プリンタドライバ25によって、画像データ内で1に出くわす度にトナー領域を印刷することにより印刷画像の画像ラインを生成するため利用することができる、プリンタ29に出力される(S17-9)。

【0208】

画像プロセッサ88は、マルチレベル画素データの最終ラインが処理されたかどうかを判断する。(S17-10)。未処理の場合、マルチレベルデータをハーフトーンデータで生成するために画像データの新しいセットが用いられる前に、生画像データ96の次のラインが特定され(S17-11)、ページメモリ84内のデータは、解凍されたビットマスクデータからの256の32ビット数の次のセットにおけるリンクマスクデータの次のセットにより上書きされる(S17-1からS17-2)。

【0209】

ラインマスクデータを記憶し、利用することにより、生の画像データ96のラインをハーフトーン画像データに変換するメモリ動作の回数は減少する。具体的には、画像データの各ラインについて、256の32ビット数であるラインマスクデータの単一セットが、ページメモリ94に転送される。かかるデータセットは、マルチレベルデータのすべてのライン用のすべての画像データを、ハーフトーンデータに変換するため利用される。全部のラインが処理された場合にのみ、別のデータがページメモリに書き込まれる。このことは、32×32の異なるビットマスクが、特定の階調の画素が印刷される毎にアクセスされる従来のビットマスク処理と対比される。

【0210】

図20は、ハーフトーン画像に変換された後の、図18に示されたマルチレベルデータの一部の例である。図20と図18を比較することにより判るが、一般に、低いレベルのマルチレベル画像データは暗い色を表す画素のように見られるので、対応するほとんど20のハーフトーンの値は、1に設定される。これとは逆に、高いレベルのマルチレベル画像データは、明るい色を表す画素のように見られるので、対応するほとんどハーフトーンの値は、0に等しい。特定の画素が、0又は1のいずれに変換されているかに拘わらず、前記画素用のマルチレベル値、ならびに、マルチレベル画像データをハーフトーンデータに変換するために用いられた生成されたビットマスクは、画素の両方の位置により決定される。

【0211】

デジタル複製部が、マルチレベル画像データの形式で表される画像内でスキャンされたものだけを処理するよう構成されているので、デジタル複製部用のプリンタドライバ30が、テキスト文書を処理するテキストプロセッサを必要としないことを除き、プリンタドライバ生成コンピュータ2により生成されたデジタル複製部4のプリンタドライバ30は、ホストコンピュータ3に記憶されたプリンタドライバ25と同じように動作する。

【0212】

他の変形例ならびに他の実施形態

上記実施形態においては、生成モジュール8の重み付けマスク記憶部42に記憶されたデータを生成するため、単一の関数を用いられていたが、異なるレベル用の重み付けマップを生成するのに、異なる関数を用いることもできる。より具体的には、ほとんど0からなる(predominately zero)アレイと関連するビットマスク用の重み付けマップを生成する場合、長い距離の後ゼロに下落する、距離により決定される関数を用いることもでき、ビットマスクアレイに多数の1を含むビットマスクデータの複数のレベルについては、より急激にゼロに下落するので、小規模な近接するアレイに基づき値を生成させる関数を用いることもできる。

【0213】

上記実施形態においては、重み付け毎の演算に用いられた関数は、特定の点と1の値を含むアレイの一部との間のユークリッド距離(Euclidian distance)、に反比例するが、他の関数を用いるようにしてもよい。具体的には、

【0214】

【数7】

$$(\alpha^2 + \beta^2)^{1/2}$$

【0215】

ここで、 ∂x は、x軸に沿った2点間の距離であり、 ∂y は、y軸上の2点間の距離であり、上記に比例する関数を有する代わりに、いかなる適切な、例えば、以下の関数を用いるようにしても良い。

【0216】

【数8】

$$(\alpha^n + \beta^n)^{1/n}$$

【0217】

ここで、nは、用いることが可能な選択値である。これに代え、以下のような、より複雑な関数を用いることもできる。

【0218】

【数9】

$$(\max(\alpha, \beta)^n + \min(\max(\alpha, \beta) - 0.1 \min(\alpha, \beta), \min(\alpha, \beta))^n)^{1/n}$$

【0219】

ここで、maxおよびminは、それぞれ、使用可能な最大および最小関数であり、または

【0220】

【数10】

$$e^{-\left(\frac{\delta x^2 + \delta y^2}{R^2}\right)}$$

【0221】

ここで、Rは使用される可能性のある1エントリーを含めることによって影響を受けるエリアの半径を表す。

【0222】

上述の実施形態において、孤立したエントリーが、より大きなエントリークラスタの一部を形成し、小さいエントリークラスタは図7に示されたパターンを構成するようにすることにより、ビットマスク生成システムは、孤立した1と0エントリーを含むビットマスクの生成を防ぐものとして記述されている。他の実施形態においては、最小クラスタの別セットを使用できれば都合がよい。

【0223】

例えば、代替となる実施形態において、ビットマスク生成システムを、隣接した1や0の

10

20

30

40

50

ペアからなるクラスタが許容されているようなシステムで構成させてもよい。またその代わりに、最も小さいクラスタが4つ以上の0か1で構成されているようなビットマスクアレイするようにシステムをアレンジしてもよい。

【0224】

ビットマスクアレイが、0や1のクラスタを生成すべく変更されることを記した上の説明において、変更対象選定は、変更数、重みスコアが基準となる。そして、2つの変更が同数のエントリーを巻き込み、同じ重み付けスコアに関連付けられる場所においては、レファレンスは乱数表に対して行われる。その他の実施形態において、2つの潜在的な変更があったとして、そのどちらかを選ぶ際、他の要素も用いることができる。とくにプリンタによって生成されたトナーのドットの垂直軸/水平軸がイコールでない場合、ビットマスク生成システムは、ページ上のトナーの配列がよりコンパクトになるという結果を導くような変更を、より優先すべき変更として選択するようにアレンジすることもできるであろう。

10

【0225】

例えば、トナーのドットが垂直方向と比べて垂直方向に2倍の距離延び、かつ、少なくとも1エントリーか0エントリーのペアから構成されているビットマスクが生成されているようなプリンタの場合、システムが、垂直方向に隣接しているペアは排除し、水平方向に隣接するエントリーのペアの方が選択されるようにすることも可能であろう。このようにして、システムをアレンジして、トナーかブランク領域で形成する4×1の細長い長方形をした画像を生成するビットマスクではなく、2×2正方形を構成する画像生成がより好まれるようにすることも可能であろう。

20

【0226】

より一般的には、1なり0の配列はそれがどれであれ、それらは、配列をプリントアウトすることによって生成される「トナー結果エリア」（または非印刷エリア）のサイズを示す数値に関連付けすることができる。また、トナー（または非印刷エリア）の配列の中でもよりコンパクトなものを生成するように導いたビットマスクアレイの選択が促されるように、ビットマスク生成システムをアレンジすることができる。

【0227】

上述の実施形態において、システムは以下のようなものとして記述されている。すなわち、もし、アレイにある水平方向または垂直方向に隣接したエントリーが「既に存在している同タイプエントリークラスタ」にある1エントリーであったならば、変更候補エントリーは、既存の1エントリークラスタを延長させるものとして吟味される、と。さて、他の実施形態において、1エントリーからなる既存のクラスタを形成しているエントリーに斜めに隣接している変更候補が、変更対象として吟味されることが許されればより都合がよい。

30

【0228】

さらなる代替案として、「ビットマスク内にあるすべての1エントリーと0エントリーは、図7で示されたタイプのビットマスククラスタの一部でなければならないという条件でビットマスクを生成するビットマスク生成システムを与えられる」という案であろう。このようなシステムにおいては、単に既存のクラスタを延長させただけの0エントリーと1エントリーは変更を許されないことになる。そのような変更が新クラスタを生成しなければという条件付きではあるが。このようなシステムのもつ利点は、0エントリーや1エントリーのクラスタが既述の方法で（既述の方法はあるレーザープリンタにとっては有効であるかもしれない）、システムによって生成された0エントリーや1エントリーのパターンよりもよりコンパクトになるということである。

40

【0229】

さらに、既述の理由で、本発明に従って生成されたビットマスクは特にレーザープリンタのプリンタドライバや印刷システムが用途として特に適しているが、生成されるビットマスクは濃淡を表現するためのドット配列として心地よいものであるから、このビットマスクが、例えばインクジェット印刷システムのような他の印刷システムにおいても使用さ

50

れると都合がよい。

【0230】

上述の実施形態において、圧縮されたビットマスクのセット全体は、印刷の実行時に解凍されるものとして既述されているが、ビットマスクセットの中の選択された部分は、違うタイミングで圧縮された方が都合がよい。より具体的には、処理されているマルチレベル画像の各ライン用に、そのラインをのみを処理するマスクデータは解凍され、マルチレベルデータをバイナリーハーフトーンデータに変換するために使われるかも知れない。マルチレベルデータの次のラインが処理される時、その次のラインを処理するためのビットマスクデータは、それから解凍してもよいのである。このようにすれば、ある画像のあるラインを処理するためのビットマスクデータが解凍される度に、画像データの前ラインを処理するためのビットマスクデータを前回保存したのと同じメモリ位置に保存できるようになるので、ビットマスクデータを保存するための総メモリ量を最小化することが可能になる。

10

【0231】

上述の実施形態においては、 32×32 ビットマスクアレイが生成されるものとして既述されたが、望ましいサイズのアレイを保存するために、乱数表44、現在のマスク記憶部46、作業対象マップ記憶部48、作業対象重み付けマップ記憶部49ををコンフィグすることにより、適したアレイサイズまたはアレイ形状が生成されたならばより都合がよい。

【0232】

上記実施形態においては、直前のビットマスクを示さない1入力の総数が既定値を超えない場合、いつでも、どの1入力の位置でも変更可能なシステムについて説明した。かかる実施形態においては、各インターアクションにおいて、ビットマスク内で1および0の拡散についての改良を最大にする、許容された変更が行われる。かかる位置変更は、1および0の拡散についての改良がなされなくまるまで、あるいは、繰り返しの最大数に達するまで行われる。

20

【0233】

明るい影用に印刷されたドットが、暗い影を印刷するためのドットパターンに現れない場合に、輪郭削りに伴う問題が生じる。直前のビットマスク内に表された1入力が増えるたびに、輪郭削りの潜在的な量を増加させる。それとは逆に、明るい影に現れない暗い影内の印刷ドット数は増加しないので、前のビットマスクに現れていない1入力の位置を変更しても、輪郭削りの量を増加させない。しかし、直前のビットマスク内に現れない1入力の一部を変更することにより、ビットマスクにおける1および0の全体的な拡散について改良が見られる。

30

【0234】

別の実施形態において、変更に供される入力を選択するため、より複雑なルールであって、さらに輪郭削りを制限するもの、を用いてもよい。したがって、例えば、アレイにおいて1入力の全体的な拡散を大きく改良させることのみが生じるよう、1入力の位置変更を限定してもよい。これは、拡散の改良度合を監視し、改良がしきい値を超えるとともに、現在の1入力のみの変更を許容することによって達成することもできる。適切な度合は、現在の1入力用に提案された新しい位置に関連する重みが、少なくとも、現在の1入力の現在位置に関連する重みより1%低い値である。拡散に関する改良が、しきい値に満たないことが判っていた場合、1入力の変更は、前のレベル内の1入力に対応しない1入力に限定してもよい。

40

【0235】

代替のシステムは、本実施形態において、拡散についての最大の改良を生じさせる変更が、最初に起こるよう選択された、という事実を利用することもできる。輪郭削りをさらに減少させるためのシステムにおいては、そのアレイのための新たな入力が増えなかった場合にのみ、移動する現在の1入力を許容してもよい。各アレイ用の新たな入力が増え適切と考えられる初期位置に位置するので、新たな入力の位置を、アレイにおける1入力の拡散についてわずかな変更のみを生じさせるよう変更する。

50

【0236】

しかし、かかる変更は、輪郭削りを増加させないので、新たな入力位置のすべてを最適化することが賢明である。変更用のいずれか新たな入力を選択する前に選択された、現在の入力変更のみを許容することにより、現在の1入力の位置の変更を許容する柔軟性のあるしきい値であって、アレイ用の新たな入力位置の変更から可能になる、拡散について改良を生じさせるレベル設定されるものを得ることができる。

【0237】

上記実施形態は、グレーレベルデータを参照しているが、本発明は、カラー印刷にも同様に適用可能であることが理解される。単一のマルチレベル画像ではなく、カラー印刷の場合、各画像を表現するため、それぞれが印刷される各色用の3又は4のマルチレベル画像が用いられる。これらのマルチレベル画像は、印刷される各色用のハーフトーン画像のセットを生成するため、既述の方法によりビットマスクデータを用い、別途処理される。次に、かかるハーフトーン画像は、1枚のカラー画像を出力するため、互いにその上に印刷される。

【0238】

カラー画像用のビットマスク場合、各色用のハーフトーンデータを生成するため、異なるビットマスクデータが用いられることが好ましい。したがって、最終的な出力画像におけるドットの拡散は、外見上好ましいものとなる。かかるビットマスクのセットを生成するため、すべての色の表現を明らかにする、より複雑な重み付け関数を用いることもできる。

【0239】

それに代えて、カラー画像について、すべての色用にビットマスクの単一セットを用いることもできる。かかるシステムにおいては、異なる色用のビットマスクの使用を相殺する(to offset)よう、マルチレベルデータ用の座標を処理することが望ましい。したがって、例えば、座標 x 、 y の画素の場合、1色で印刷するかどうかは、 y 番目のライン用のラインマスクの x 番目の入力をチェックすることにより決定され、他の色については、 $y+\theta$ 番目のライン用のラインマスクの $x+\epsilon$ 番目の入力をチェックすることにより決定される。ここで、 ϵ および θ は、異なる色についての相違を相殺するものである。

【0240】

上記実施形態においては、マルチレベル画像データの各画素用にギャップ又は単一のトナードットを生成するシステムが説明されていたが、出力画像の解像度は、元のマルチレベル画像以下でもそれ以上でもよいこと、が理解される6出力画像が、元のマルチレベル画像の解像度より低い場合、当該画像をハーフトーン画像に変換するビットマスクを選択するため画素ブロック用の平均されたマルチレベルデータ値を用いてもよい。とれとは逆に、出力画像が、元のマルチレベル画像の解像度より高い場合、出力画像の対応する領域を表すためにビットマスクデータのブロックを識別するよう、画素用のマルチレベルデータ値を用いるようにしてもよい。

【0241】

上記実施形態においては、プリンタドライバについて説明がなされたが、本発明は、ビットマスクを用いて処理されるどのようなハーフトーン画像に対しても同様に適用可能であることが理解される。したがって、例えば、本発明は、ハードウェアあるいはソフトウェアによるラスター画像処理プロセッサ、ならびに、プリンタ自身の中で実行されるハーフトーン処理に適用可能である。

【0242】

図面を参照して説明された本発明複数の実施形態は、コンピュータ装置ならびにコンピュータ装置内で実行されるプロセスを含んでおり、また、当該発明は、コンピュータプログラム、具体的には、キャリア上又はその中のコンピュータプログラムであって、本発明を具現化するもの、にも及ぶ。前記プログラムはソース又はオブジェクトコード、あるいは、本発明にかかるプロセスを実行するために用いられるのに適切であれば、他のどのような形式であってもよい。前記キャリアは、プログラムを搬送することができる実体ある

いは、装置のいずれであってもよい。

【0243】

例えば、前記キャリアは、例えば、ROM、すなわち、CDROM、又は、半導体ROM、フロッピーディスク又はハードディスク等の磁気記録媒体、等の記憶媒体を備えていてもよい。また、かかるキャリアは、電気または光学ケーブル又は無線、あるいは、他の方法を通じて搬送される電氣的または光学的信号等の、送信可能なキャリアでもよい。プログラムが、ケーブル又は他の装置あるいは手段で直接搬送される信号により具現化された場合、前記キャリアは、かかるケーブル又は装置あるいは手段により構成される。

【0244】

なお、かかるキャリアは、その中に前記プログラムが埋め込まれている集積回路であってもよく、当該集積回路は、関連するプロセスを実行するよう、あるいは、かかる実行で使用されるよう改造される。

【図面の簡単な説明】

【0245】

【図1A】本発明の一実施形態に基づき、ビットマスクのセットを用いる印刷に伴うステップの略図を示している。

【図1B】本発明に基づきメモリ内に記憶されたビットマスクのセットブロック図である。

【図2】本発明の実施形態に基づく、ビットマスクアレイ生成システム、プリンタドライバの部品の全体、ならびに、印刷された画像、を示すブロック図である。

【図3】図2のシステムのビットマスク生成コンピュータの一部を構成する、生成モジュールのブロック図である。

【図4】図2のシステムのビットマスク生成コンピュータにより行われる、処理のフロー図である。

【図5】図3の生成モジュールにより生成された、重み付けマスクの一例である

【図6A】図3の生成モジュールにより生成されたビットマスクアレイ内に含まれる複数の位置を決定するための、処理のフロー図である。

【図6B】図3の生成モジュールにより生成されたビットマスクアレイ内に含まれる複数の位置を決定するための、処理のフロー図である。

【図7】ビットマスクにおいて有効な12個のクラスタ形状を示す図である。

【図8】そのためにビットマスクが生成される、異なるレベルの階調用の重み付けマップの生成を変更するための、関数を示すグラフである。

【図9A】生成されたビットマスクの一部を表す数字列において修正された0の例を図示している。

【図9B】図9Aで示された数字列を変更すべく、図5において図示されている重み付けマスクを用いて重み付けマップをアップデートしたことに起因する重み付けマップの諸数値の増加を例として図示している。

【図10A】ビットマスクを変更するための、図3の生成モジュールの処理を示すフロー図である。

【図10B】ビットマスクを変更するための、図3の生成モジュールの処理を示すフロー図である。

【図10C】ビットマスクを変更するための、図3の生成モジュールの処理を示すフロー図である。

【図11】図2のシステムのビットマスク生成コンピュータの一部を構成する、圧縮モジュールのブロック図である。

【図12】図3の生成モジュールにより生成されたビットマスクを、圧縮可能にするための、データの再配置を示すブロック図である。

【図13A】圧縮されるビットマスクを表す、データの一例である。

【図13B】圧縮されるビットマスクを表す、データの一例である。

【図13C】圧縮されるビットマスクを表す、データの一例である。

【図 1 4】図2のプリンタドライバ生成部により生成されたプリンタドライバを含み図2のシステムの一部を構成する、ホストコンピュータのブロック図である。

【図 1 5】図14のプリンタドライバを用いた印刷プロセスのフロー図である。

【図 1 6 A】図14のプリンタドライバによるデータの圧縮の一例である。

【図 1 6 B】図14のプリンタドライバによる、データの圧縮の一例である。

【図 1 6 C】図14のプリンタドライバによる、データの圧縮の一例である。

【図 1 7】図14のホストコンピュータのプリンタドライバによる、ハーフトーン画像の生成フロー図である。

【図 1 8】画像の一部を表すマルチレベル階調値の一例である。

【図 1 9】印刷される画像において画素をどのように表すかを決定するため、図14のホストコンピュータのプリンタドライバを用いた、ビットマスクの一部の例である。

【図 2 0】図18の階調値の配列を変換することにより生成された、パーフトーン画像の一部の例である。

【符号の説明】

【 0 2 4 6 】

4 2 マスク記憶部(weight mask store)

4 4 乱数テーブル

4 6 マスク記憶部

4 7 稼動マスク記憶部

4 8、4 9 マップ記憶部

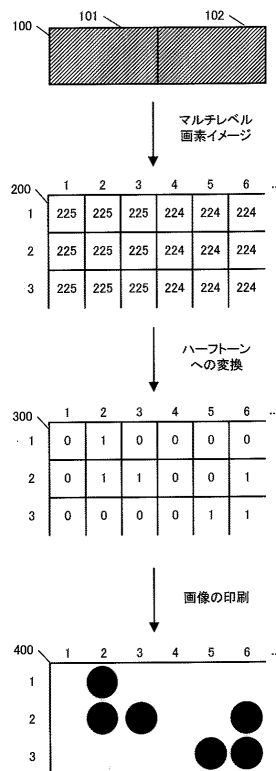
5 0 位置外れリスト

5 2 ニュードットリスト

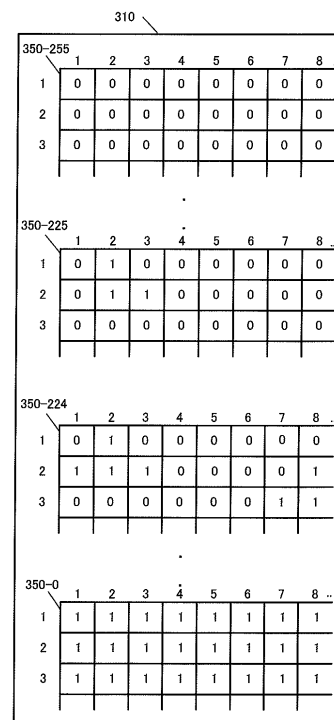
10

20

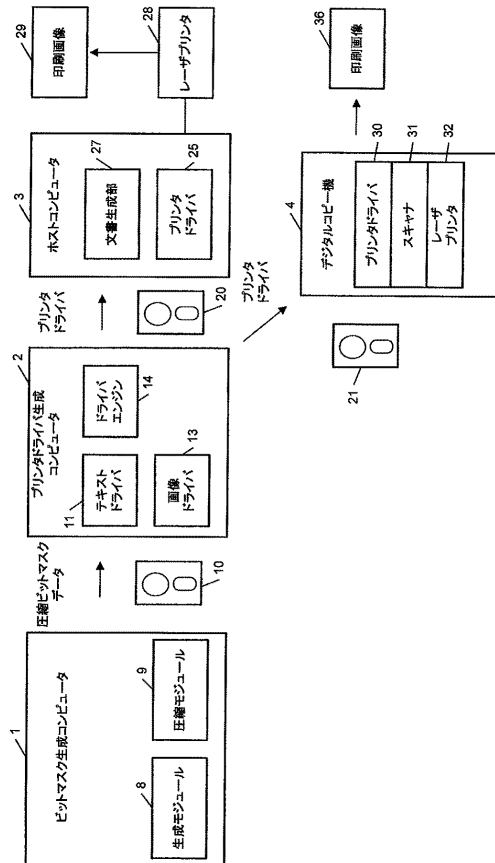
【図 1 A】



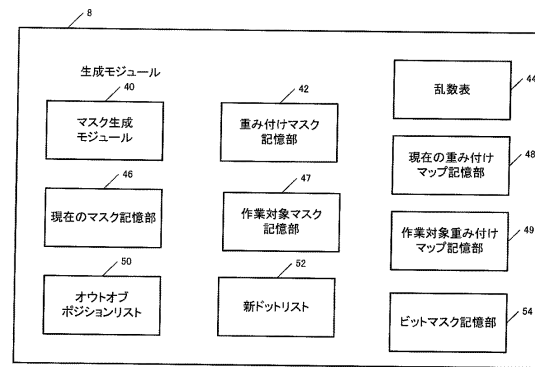
【図 1 B】



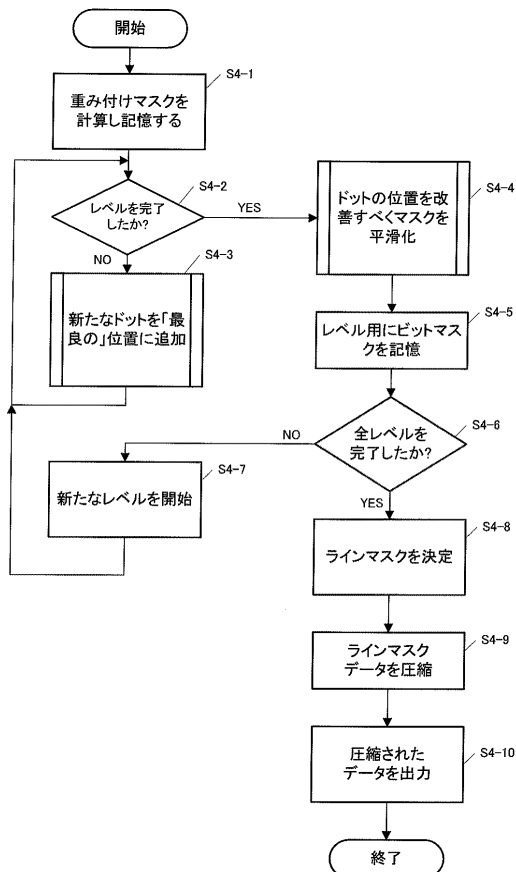
【図 2】



【図 3】



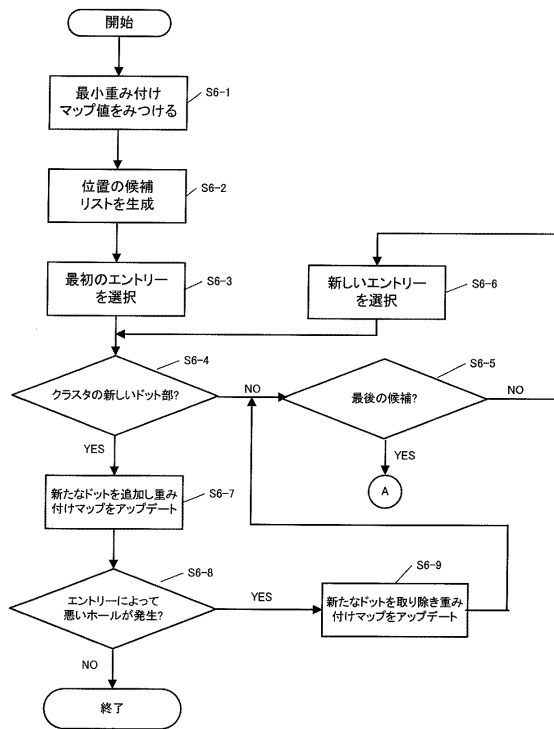
【図 4】



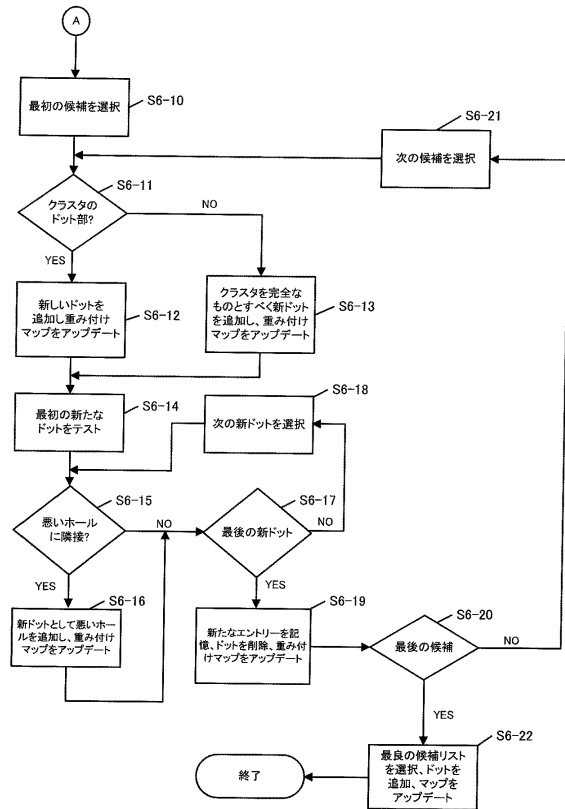
【图 5】

0.14	0.14	0.14	0.13	0.0	0.0	0.0	0.0	7
0.16	0.16	0.16	0.15	0.14	0.0	0.0	0.0	6
0.20	0.20	0.19	0.17	0.15	0.14	0.0	0.0	5
0.25	0.24	0.22	0.20	0.18	0.15	0.14	0.0	4
0.33	0.31	0.27	0.23	0.20	0.17	0.15	0.13	3
0.50	0.44	0.35	0.27	0.22	0.19	0.16	0.14	2
1.0	0.50	0.44	0.31	0.24	0.20	0.16	0.14	1
100.0	1.0	0.50	0.33	0.25	0.20	0.16	0.14	0
0	1	2	3	4	5	6	7	

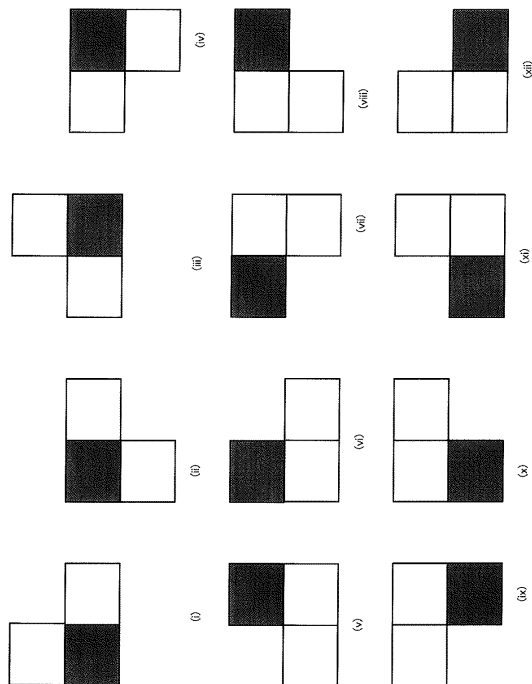
【図 6 A】



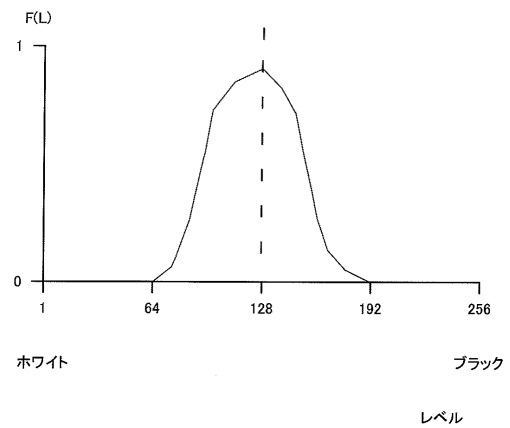
【図 6 B】



【图 7】



【図 8】



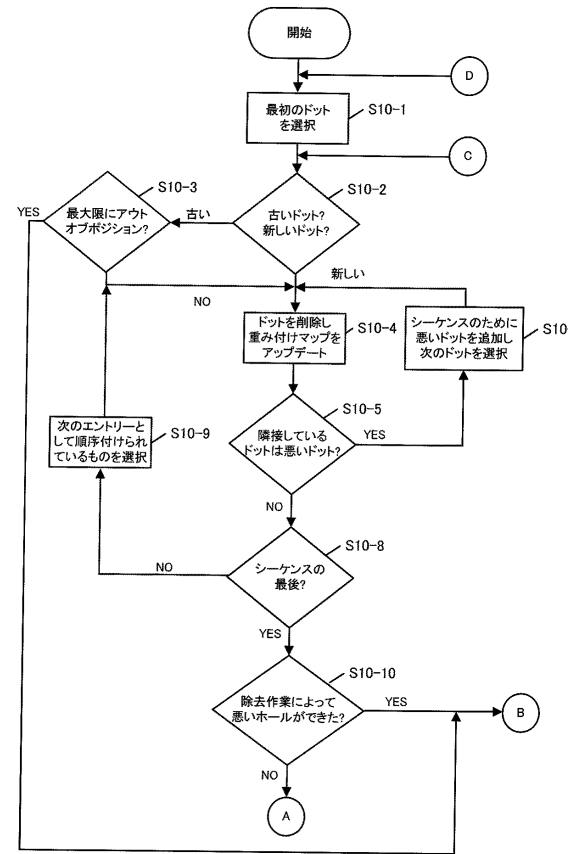
【図 9 A】

[illegible]

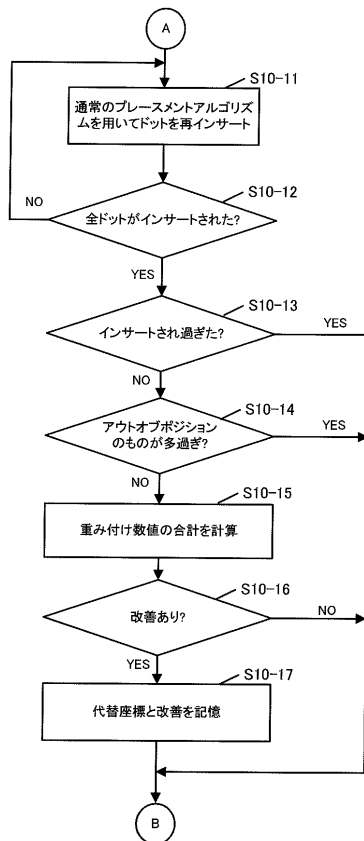
【図 9 B】

0.35	0.44	0.50	0.44	0.35	0.27	0.22	0.20	0.16	0.13
0.44	0.50	1.0	0.50	0.44	0.31	0.24	0.20	0.16	0.14
0.50	1.0	100.0	1.0	0.50	0.33	0.25	0.20	0.16	0.14
0.44	0.50	1.0	0.50	0.44	0.31	0.24	0.20	0.16	0.14
0.35	0.44	0.50	0.44	0.35	0.27	0.22	0.19	0.16	0.13
0.27	0.31	0.33	0.31	0.27	0.23	0.20	0.17	0.15	0.0
0.20	0.24	0.25	0.24	0.22	0.20	0.18	0.15	0.14	0.0
0.17	0.20	0.20	0.20	0.20	0.17	0.15	0.14	0.0	0.0

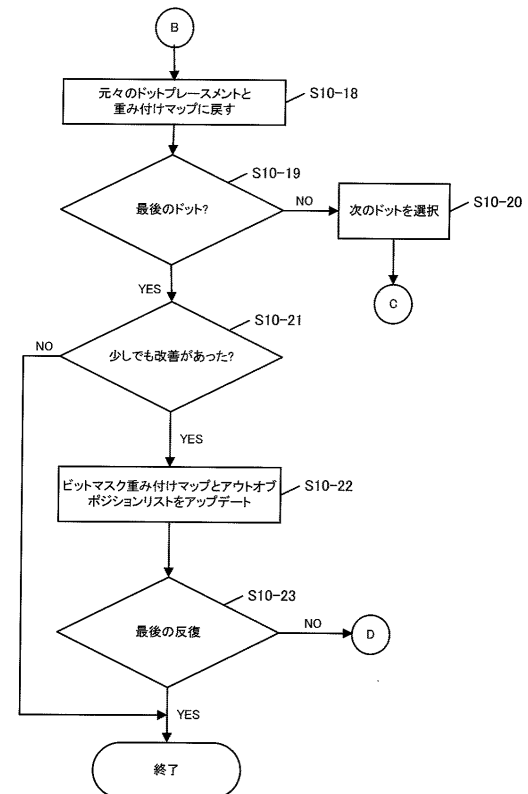
【図 10 A】



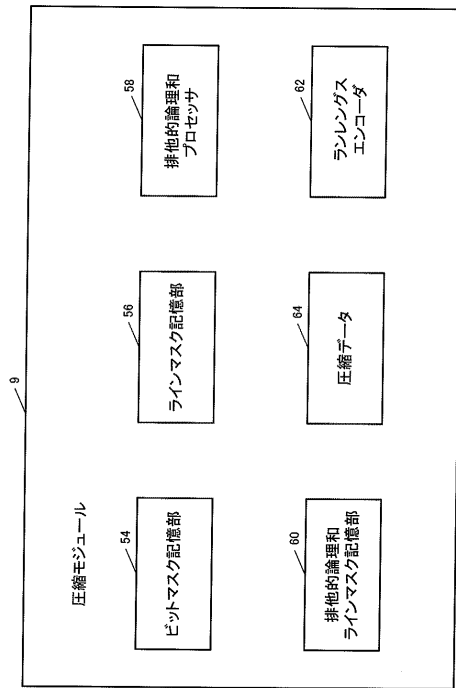
【図 10 B】



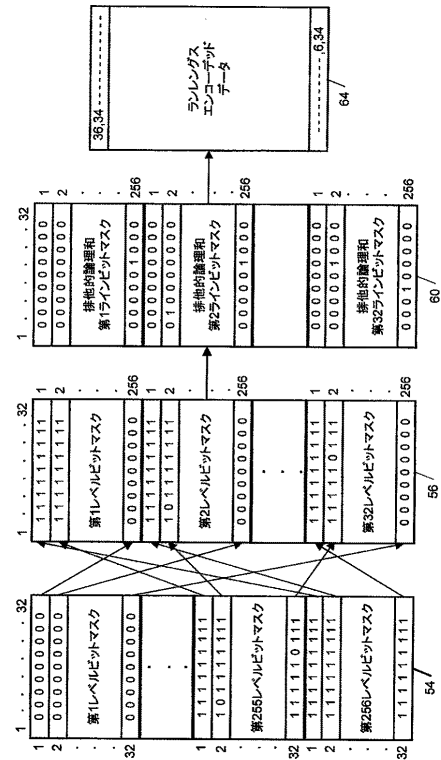
【図 10 C】



【図 1 1】



【図 1 2】



【図 1 3 A】

レベルの増加 ↓

1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	1	1	1	0	1	
1	1	1	1	1	0	1	1	1	0	1	
1	1	1	1	0	1	1	1	1	0	0	
0	1	0	1	0	1	1	1	1	0	0	
0	0	0	1	0	1	1	1	1	0	0	
0	0	0	1	0	1	0	1	1	0	0	

【図 1 3 C】

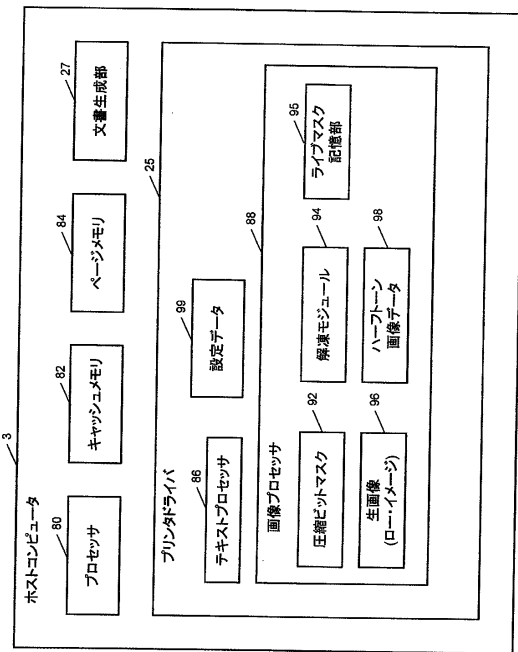
... 16, 3, 16, 0, 4, 0, 1, 9, 15 ...

【図 1 3 B】

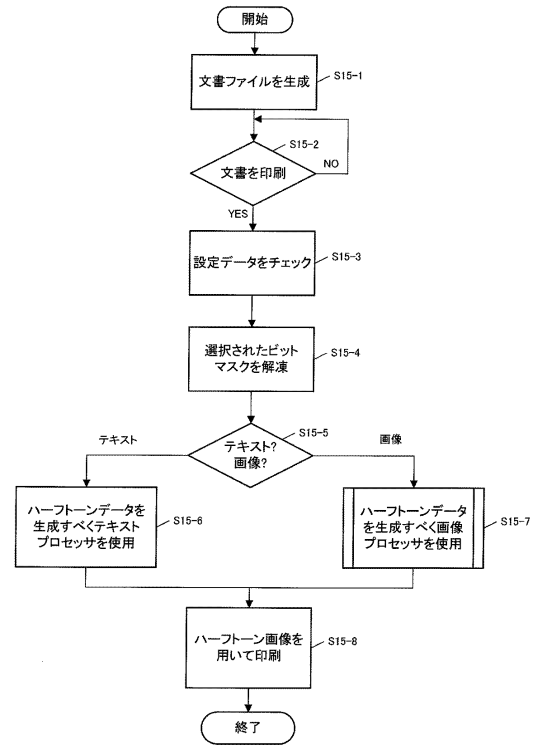
レベルの増加 ↓

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	1	0	
0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	1	1	0	0	0	0	1	
1	0	1	0	0	0	0	0	0	0	0	
0	1	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	1	0	0	0	0	

【図 14】



【図 15】



【図 16 A】

... 16, 3, 16, 0, 4, 0, 1, 9, 15 ...

【図 16 B】

レベルの増加 ↓

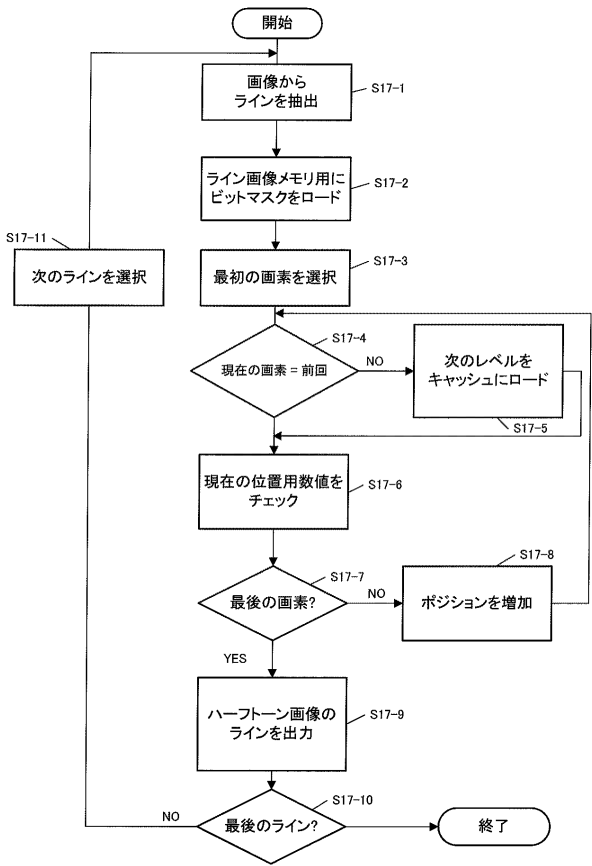
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	1
1	0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0

【図 16 C】

レベルの増加 ↓

1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	1	1	1	0	1	1
1	1	1	1	1	0	1	1	1	0	1	1
1	1	1	1	0	1	1	1	1	0	0	0
0	1	0	1	0	1	1	1	1	0	0	0
0	0	0	1	0	1	1	1	1	0	0	0
0	0	0	1	0	1	0	1	1	0	0	0

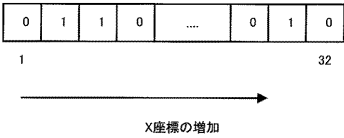
【図 17】



【図 18】

	1	2	3	4	5	6	7	8
1	154	153	153	154	153	154	20	15
2	154	152	152	22	21	20	20	10
3	253	225	15	35	15	35	20	5
4	252	250	15	35	35	15	30	15

【図 19】



【図 20】

	1	2	3	4	5	6	7	8
1	0	1	1	0	1	1	0	0
2	0	0	1	1	1	1	1	0
3	0	0	1	1	1	1	1	1
4	0	0	1	1	0	0	1	1

フロントページの続き

- (72)発明者 マイケル イアン ウッズ
イギリス オーエックス4 2エイチエヌ オックスフォード オックスフォード ビジネス パ
ーク ガーシントン ロード 9400 ソフトウェア 2000 リミテッド内
- (72)発明者 ジュディ ブース
イギリス オーエックス4 2エイチエヌ オックスフォード オックスフォード ビジネス パ
ーク ガーシントン ロード 9400 ソフトウェア 2000 リミテッド内

審査官 加内 慎也

- (56)参考文献 特開2001-036745 (JP, A)
特開2001-326817 (JP, A)
特開2000-307861 (JP, A)
特開2003-101776 (JP, A)
特開2003-046777 (JP, A)
特開2003-319179 (JP, A)

- (58)調査した分野(Int.Cl., DB名)
H04N 1/405