(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3839942号
(P3839942)
(45)発行日　平成18年11月1日(2006.11.1)　(24)登録日　平成18年8月11日(2006.8.11)

(51)Int.Cl.			F I		
H04N	1/387	(2006.01)	H04N	1/387	101
G06T	1/60	(2006.01)	G06T	1/60	450E
H04N	1/60	(2006.01)	H04N	1/40	D
H04N	1/46	(2006.01)	H04N	1/46	Z

請求項の数 5　(全 27 頁)

(21)出願番号　特願平9-339824
(22)出願日　平成9年12月10日(1997.12.10)
(65)公開番号　特開平10-210279
(43)公開日　平成10年8月7日(1998.8.7)
審査請求日　平成16年11月29日(2004.11.29)
(31)優先権主張番号　785,144
(32)優先日　平成9年1月13日(1997.1.13)
(33)優先権主張国　米国(US)

(73)特許権者　398038580
ヒューレット・パッカード・カンパニー
HEWLETT-PACKARD COMPANY
アメリカ合衆国カリフォルニア州パロアルト　ハノーバー・ストリート　3000
(74)代理人　100081721
弁理士　岡田　次生
(72)発明者　ロバート・ジー・ガン
アメリカ合衆国80512コロラド州ベルビュー、リスト・キャニオン・ロード　11491

最終頁に続く

(54)【発明の名称】画像の複数描画生成方法

(57)【特許請求の範囲】

【請求項１】

スキャナを使用して原画像の複数の画像を生成する方法であって、

上記スキャナを使用して上記原画像の単一走査を実行するステップと、

上記スキャナが、上記単一走査で得られる上記原画像から２以上の画像を走査中に並行して生成するステップと、を有し、

前記画像は、少なくとも１つの高解像度グレイスケール画像および少なくとも１つの低解像度高ビット・カラー画像を含む、スキャナを使用して原画像の複数画像を生成する方法。

【請求項２】

更なる処理のために、上記スキャナが、上記単一走査から得られる上記原画像から生成された２以上の画像をホスト・コンピュータに送信するステップを更に含む、請求項１に記載のスキャナを使用して原画像の複数の画像を生成する方法。

【請求項３】

単一走査で得られる原画像から２以上の画像を走査中に同時に生成するよう動作可能な画像処理装置と、

単一走査された原画像を２以上の画像に分割するよう動作可能なプレスケーラと、を有し、

前記画像は、少なくとも１つの高解像度グレイスケール画像および少なくとも１つの低解像度高ビット・カラー画像を含む、スキャナ。

【請求項４】
原画像の複数の画像を生成する方法であって、
スキャナを使用して上記原画像の単一走査を実行するステップと、
上記スキャナが、上記単一走査された原画像の２以上の画像を表す２以上のデータ信号を同時に生成するステップと、を有し、
前記画像は、少なくとも１つの高解像度グレイスケール画像および少なくとも１つの低解像度高ビット・カラー画像を含んでおり、
上記スキャナが、上記単一走査された原画像の２以上の画像を表す上記２以上のデータ信号をインターリーブし、単一のインターリーブされたデータストリームを生成するステップと、
を有する原画像の複数の画像を生成する方法。
【請求項５】
上記単一走査された原画像の２以上の画像の更なる処理のため、上記スキャナが、上記インターリーブされた単一データストリームをコンピュータに送信するステップを更に有する、請求項４に記載の原画像の複数の画像を生成する方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、一般的には光学スキャナに関するもので、特に単一走査からカラーおよび白黒画像を含む複数の画像を取得する方法およびシステムに関するものである。
【０００２】
【従来の技術】
光学スキャナは、画像を捕捉しデジタル化するために使用される。例えば、光学スキャナを使用して１枚の紙の上の印刷物の画像を捕捉することができる。デジタル化された画像は、次に電子的に記憶され、更に文字認識ソフトウェアを用いてテキストを生成するように処理されることもできる。大部分の光学スキャナは、照明および光学システムを使用して、オブジェクトを照射して、通常「走査行」と呼ばれる照射されたオブジェクトの小さい領域を光センサ・アレイ上へ集束させる。照明および光学部品に対してオブジェクトを移動させるか、あるいはオブジェクトに対して照明および光学部品を移動させるかいずれかによって、照射される走査行をオブジェクト全体にわたって掃くように移動させることによってオブジェクト全体の走査が実行される。
【０００３】
典型的スキャナ光学システムは、照射される走査行の画像を光センサ・アレイ表面に集束させるためのレンズ部品を含む。特定の設計によっては、スキャナ光学システムは、また、複数のミラーを備えて、光線経路を折り曲げることによって光学システムを比較的小さい筐体内に実装することができるようにすることもある。
【０００４】
種々のタイプの光センサ装置が光学スキャナで使用される可能性があるとはいえ、一般的に使用されるセンサは電荷結合素子(charge coupled device)すなわちＣＣＤである。よく知られているように、ＣＣＤは、各々が光の露出に反応して電荷を収集または構築する多数の個別セルあるいはピクセルを含む。所与のセルまたはピクセルに蓄積される電荷の大きさは光露出の強度および持続時間に関係しているので、集束される画像上の明るいスポットおよび暗いスポットを検知するためＣＣＤが使用される。典型的スキャナ・アプリケーションにおいては、ＣＣＤセルまたはピクセルの各々に構築される電荷が測定され、典型的スキャナで約５ミリ秒程の露出時間またはサンプリング間隔として知られる規則的間隔で放電される。荷電(すなわち画像データ)は露出時間の間にＣＣＤセルに同時に収集されるので、ＣＣＤセルからの同時または並列的データを連続または逐次データストリームに変換するためアナログ・シフト・レジスタがＣＣＤにおいて使用される。
【０００５】
典型的アナログ・シフト・レジスタは、各々が個別セルに接続されている複数の電荷転送

バケットを含む。露出時間の最後に、ＣＣＤセルの各々によって収集された電荷が同時に電荷転送バケットへ転送され、ＣＣＤセルは次の露出に備える。次に、各バケットの電荷が、ＣＣＤセルが次の走査行に対して露出されている間に、バケットからバケットへバケツ・リレーのように伝送される。次に、順に並べられたＣＣＤセルからの電荷は、１つずつ、適切なアナログ／デジタル変換器によってデジタル信号に変換される。

【０００６】

大部分の光学スキャナのアプリケーションにおいては、ＣＣＤの個別ピクセルの各々は線形アレイを形成するように、順に配列される。このように、ＣＣＤアレイにおける各ピクセルは、照射された走査行の関連ピクセル部分に対応する。線形光センサ・アレイにおける個別ピクセルは、一般的に、「横方向」、すなわちオブジェクトを横切って進む(照射された)走査行の移動方向に対し垂直な方向に配列される。従って、線形光センサ・アレイの各ピクセルは、横方向に測定される長さおよび走査方向に測定される幅を持つ。大部分のＣＣＤアレイにおいて、ピクセルの長さおよび幅は等しく、典型的にはそれぞれ約８ミクロンである。

【０００７】

横方向のサンプリング率は、ＣＣＤにおける個別セルの数の関数である。例えば、一般に広く使用されているＣＣＤ光センサ・アレイは、約６００ＰＰＩ(ピクセル／インチ)という横方向におけるサンプリング率を可能にするために十分な数の個別セルを含む。このようなサンプリング率は横方向における本来的サンプリング率と呼ばれる。

【０００８】

走査方向のサンプリング率は、走査行移動速度とＣＣＤ露出時間の積(サンプリング間隔)に逆比例する。従って、走査方向のサンプリング率は、走査行移動速度またはＣＣＤ露出時間、あるいはその両者を減少させることによって増加させることができ、逆に、走査行移動速度またはＣＣＤ露出時間、あるいはその両者を増加させることによってサンプリング率を減少させることができる。所与の露出時間に関する「走査方向の最小サンプリング率」は、露出時に最大走査行移動速度で走査する時に達成されるサンプリング率である。例えば、毎秒３.３３インチの走査行移動速度および約５ミリ秒の最大露出時間は、約６００ＰＰＩという走査方向における最小サンプリング率を発生する。

【０００９】

【発明が解決しようとする課題】

現在では、光学文字認識(ＯＣＲ)は、正確な結果を得るため、３００ＰＰＩサンプリング率を必要とする。このように、高解像度だが低ビット深度である３００ＰＰＩ４ビットグレー走査(8.5 X 11)は約４.２メガバイトである。精度の高いカラーは２４ビット・カラーを必要とする。低解像度だが高ビット深度である１５０ＰＰＩ２４ビットカラー走査(8.5 X 11)は、約６.３メガバイトである。カラー画像およびＯＣＲを必要とする書き込み文字両方を含むドキュメントの走査を行うためには、走査は、３００ＰＰＩ２４ビット(8.5 X 11)でなければならず、これは、約２５.２４メガバイトのメモリに相当する。従って、テキストおよび画像を含むドキュメントの走査は多くのメモリを必要とする。しかし、コンピュータ上のソフトウェアは、カラー画像のサンプリングを約６.３メガバイトに下げ、カラー画像を放棄してテキストを取得する。このプロセスは、ソフトウェアで実行するには非常に遅く、また多量のメモリを不必要に消費する。代替方法としては、テキストまたはグラフィックスのいずれかを先ず走査し次に別の一方の走査を実行し、その後ドキュメントはソフトウェアによって再生成される。しかしながら、この代替方法も、多くのメモリを使用することに加えて非常に時間のかかるドキュメント走査方法である。

【００１０】

従って、テキストおよびグラフィックスの両方を含むドキュメントを走査し、(現行では速度的に制約のある)スキャナからホスト・コンピュータへ送られるデータの総量を減少させ、ホスト・コンピュータのソフトウェアによって処理されるデータの総量を減少させることができるスキャナが必要とされている。

【００１１】

【課題を解決するための手段】
本発明において、上記課題の解決は、複数画像スキャナにおいて達成される。該スキャナは、(例えばテキストおよびグラフィックスのような)複数のタイプの画像を含むドキュメントの単一走査を実行し、該スキャナからホスト・コンピュータへ同一画像の複数の描画(例えば１つの高解像度グレイスケール画像および１つの低解像度高ビット深度カラー画像)を送り、それによって、ホスト・コンピュータへ送られ処理されるデータの総量を大幅に減少させる。データ量の減少は、例えば、約２５．２４メガバイトが、(３００ＰＰＩグレー画像(8.5 X 11)の４．２メガバイトおよび１５０ＰＰＩ２４ビット・カラー(8.5 X 11)の６．３メガバイトの合計である)約１０．５メガバイトに減少する。すなわち２．４対１の減少である。

【００１２】
本発明のスキャナは、例えば画像間の解像度比率が２であるとすれば高解像度グレー・データの２行および低解像度カラー・データの１行を行単位にインターリーブした形態で複数画像をホスト・コンピュータへ送る。本発明の利点は、ホスト・ソフトウェアが処理し保存すべき情報も、(スキャナ速度の制約を受ける)スキャナからホストへ送られるべき情報も、従来技術に比較して少ない。

【００１３】
更に、本発明は、インターリーブされた画像データストリームを、ホスト・コンピュータによる走査および更なる処理に備えて個別画像に解析することができる機能を持つホスト・コンピュータ・ソフトウェアを含む。

【００１４】
本発明は、スキャナを使用して１つの画像の複数描画を生成する方法を含む。該方法は、上記スキャナを使用して単一の走査を実行するステップ、および、上記単一走査によって取得された画像の複数の描画を上記スキャナが生成するステップを含む。単一走査によって取得された画像の複数の描画を生成する上記ステップは、少くとも１つのグレイスケール画像および少くとも１つのカラー画像を生成するサブステップを更に含み、更にこのサブステップは、少くとも１つの高解像度グレイスケール画像および少くとも１つの低解像度高ビット深度カラー画像を生成するサブステップを更に含む。上記スキャナが上記単一走査画像の上記複数描画を生成すると、上記スキャナは、上記単一走査画像から取得された上記画像の上記複数描出を更なる処理のためホスト・コンピュータへ送信する。

【００１５】
本発明は、単一走査を基に１つの画像の複数の描画を生成する画像処理装置を備えるスキャナを含む。上記画像処理装置は、単一走査画像を該単一走査画像の複数描画に分割するプレスケーラを含む。上記プレスケーラは、また、上記単一走査画像を、上記単一走査画像の少くとも１つの白黒描画と上記単一走査画像の少なくと１つのカラー描画に分割する機能を含む。更に、上記プレスケーラは、少くとも１つの高解像度グレイスケール画像および１つの低解像度高ビット深度カラー画像に分割する機能を含む。

【００１６】
【発明の実施の形態】
本発明は、単一の走査から２つのセットの走査データを同時に取得することができるスキャナを提供する。走査の区域が同一であるにもかかわらず、データ・タイプおよび解像度の観点から見たビュー(すなわち画像の様相)は変わる。例えば、ドキュメントの走査が実行される時、該スキャナの中の画像処理装置が、走査されたドキュメントから１つの高解像度グレイスケール画像および１つの低解像度２４ビット・カラー画像を生成して、１つのデータストリームにインターリーブされる形態でその両方の描画をホスト・コンピュータに送る。次に，ホスト・コンピュータは、走査されたドキュメントの２つのビューを再作成するためその両方の描画を解析する。

【００１７】
図１は、２画像スキャナに関するデータ経路のブロック図であって、点線部分１００は画像処理装置によって実施されるスキャナ機能であり、点線部分１０２はホスト・コンピュ

ータ機能である。本実施形態が白黒およびカラー画像を含む２画像スキャナではあるけれども、概念は複数画像に拡張することができる点は注意されるべきである。画像処理装置１００は、図２および図３を参照しながら後述される機能上の細部を備えるＡＳＩＣの形態で実施される。ドキュメントが走査される時、ＣＣＤアレイ１０４またはその他の既知の光センサ装置は、赤データ信号１０６、青データ信号１０８および緑データ信号１１Ｏを出力する。次に、これら３つのデータ信号は、画像プレプロセッサ１１２によって、カラー画像データ信号１２８および単色画像データ信号１３０に変換される。画像プレプロセッサ１１２は、カラー画像データ信号１２８をバッファ１１４のカラー・バッファ部分１１８に書き込み、単色画像データ信号１３０をバッファ１１４の単色バッファ部分１１６に書き込む。

【００１８】

次に、画像ポストプロセッサ１２０が、カラー・バッファ１１８および単色バッファ１１６にある画像データを使用して、ホスト・コンピュータの指定に従って、スキャナによって走査された１つまたは２つのいずれかのビューのウィンドウを生成する。２つのビューが生成される場合、画像ポストプロセッサは、インターリーブされたデータストリーム１３６という形態でその２つのビューをホスト・コンピュータに送信する。

【００１９】

ホスト・コンピュータが走査ウィンドウの２つのビューを受け取ると、データ解析機構１２２は、データストリーム１３６を解析してカラー画像１２４および単色画像１２６に分解する。この時点で、２つのビューはホスト・コンピュータで処理される準備ができる。この利点は、ホスト・ソフトウェアが処理し保存すべき情報も、(スキャナ速度の制約を受ける)スキャナからホストへ送られるべき情報も、従来技術に比較して少ない。

【００２０】

I.複数画像スキャナ

２画像スキャナは、１回の走査で完全なページ(テキストおよび図形／画像)情報をホスト・コンピュータに送るために必要とされるデータを減少させる。現行の光学文字認識は、正確な結果を得るためには３００ＰＰＩサンプリング率を必要とする。カラーを忠実に表現するには、２４ビット・カラー走査が必要である。これを実行するための典型的走査は、(8.5 X 11の場合)２４ビット深度の３００ＰＰＩであり、これは２５.２４メガバイトである。同じページの２つの描画、すなわち１つの高解像度グレイスケール画像および１つの低解像度２４ビット・カラー画像をスキャナが送出することを可能とさせることによって、ホスト・ソフトウェアに送られそれによって処理されるデータの総量は大幅に減少する。３００ＰＰＩ４ビットのグレー画像(8.5 X 11)は約４.２メガバイト、１５０ＰＰＩ２４ビット・カラー(8.5 X 11)は６.３メガバイトで合計約１０.５メガバイトであり、２.４対１のデータ減少である。

【００２１】

本発明の複数画像スキャナは、上述の動作を非常に迅速に実行する画像処理装置を持つ。画像処理装置は、本明細書で記述される機能を持つ従来技術のフィールド・プログラム可能ゲートアレイまたはＡＳＩＣまたはその他の形態で実現することができる。複数画像スキャナの画像処理装置は、それぞれ詳細は後述される画像プレプロセッサ１１２、バッファ１１６および画像ポストプロセッサ１２０を備える。

【００２２】

A.画像プレプロセッサ

図２に示されるように、プレプロセッサ１１２は、ＣＣＤ１０４から(赤１０６、緑１０８および青１１０からなる)ＲＧＢ画像データを取り出し、その画像の１つの表現または２つの異なる表現のいずれかをバッファ１１４へ書き込む。可能な表現の１つはカラー画像であり、他方は、３つのカラー・チャネルから１つ(赤，緑または青)だけを選択することによって生成される単色画像である。２つの異なるバッファ画像は、同じｙ解像度を持つが、ｘ解像度およびビット深度は異なる。各画像のビット深度は１２ビットまたは８ビットのいずれかに選択することができ、またｘ解像度は、ＣＣＤ解像度を１から６３まで

の整数係数によって除した数値とすることができる。ＣＣＤ解像度は、両方の画像に対して同一であり、３００ＰＰＩまたは１５０ＰＰＩのいずれかである。例えば、画像プレプロセッサ１１２は、３００ＰＰＩ(y)×３００ＰＰＩ(x)８ビット単色(緑経路)データおよび３００ＰＰＩ(y)×１００ＰＰＩ(x)１２ビット・カラー・データをバッファに書き込むようにプログラムすることができる。必ずしも両方の画像が生成されなくてもよい点に注意する必要がある。プレプロセッサは、カラー表現だけまたは単色表現だけ、あるいはその両方を生成するようにプログラムすることができる。本発明の画像プレプロセッサによって実行される４つの主要機構を図１および図２を参照して以下に記述する。

【００２３】

1.Ａ／Ｄ変換器

Ａ／Ｄ変換器は、ＣＣＤアレイから３００ＰＰＩまたは１５０ＰＰＩアナログＲＧＢ画像データを取り出して、各ピクセル毎に３つのカラー値すべてのうちの１つまたは３つのいずれかを１０ビット・デジタル数へ変換する。これらのデジタル値は埋め込みビットを含む１２ビットにされてダーク・クランプに送られる。

【００２４】

2.ダーク・クランプ

ダーク・クランプは、当該行に関する平均ダーク電流を各ピクセル値から減ずる。次に、所望の走査ウィンドウの内側のピクセルが補正器に送られる。

【００２５】

3.補正器

補正器は、不均一な照射およびＣＣＤ反応に起因したピクセル変動を削除する。補正されピクセル値は次にプレスケーラに送られる。

【００２６】

4.プレスケーラ

プレスケーラは補正されたピクセル値を単色画像およびカラー画像に分割して、異なる整数係数によって各画像のｘ解像度を減少させ、１つまたは２つの生成画像をバッファへ書き込む。プレスケーラは補正器からの画像データを単色ストリームおよびカラー・ストリームに分割して、異なる整数係数によって各ストリームのｘ解像度を減少させ、１つまたは２つの生成画像をバッファへ保存する。これによって、画像データの各行によって消費されるバッファ空間量が最小限にとどめられる。この効果は特に２つのビューがホストによって要求される時顕著である。例えば、プレスケーラが備えられず、ホストが８.５インチ幅の操作ウィンドウの３００ＰＰＩ単色表現および１００ＰＰＩカラー表現を求めるとすれば、画像の各行は、１１.４７５ＫＢのバッファ空間（８.５インチ３００ＰＰＩ３カラー／ピクセル１.５バイト／カラー)を消費する。しかし、プレスケーラを使用すれば、各行は、６.３７５ＫＢ(８.５インチ３００ＰＰＩ１カラー／ピクセル＋８.５インチ１００ＰＰＩ３カラー／１.５バイト／カラー)を消費するにすぎない。

【００２７】

プレスケーラは全１２ビット・データまたは切り捨てを行った８ビット・データのいずれかをバッファに保存することができる点に注意する必要がある。データ幅は、各ストリーム毎に独立して指定できる。プレスケーラは、２つのデータストリームのｘ解像度を減少する際、単にピクセルを落とすのではなく、隣接するピクセルを一緒に平均する。これは、実質的には、より低い解像度で再サンプリングされる前に画像データをローパス・フィルタしているもので、エリアジングの減少に役立つ。プレスケーラは、１から６３の整数係数によってｘ解像度を減じることができる。

【００２８】

B.画像ポストプロセッサ

画像ポストプロセッサは、バッファに保存された画像を使用して、ホストによって指定された走査ウィンドウの１つまたは２ついずれかのビューを生成する。各ビューは、カラー画像または単色画像から生成される。２つのビューは、同一画像あるいは異なる画像いずれからでも生成することができる。例えば、１つのビューをカラー画像から生成し別のビ

ューを単色画像から生成することができるし、また、両方のビューをカラー画像から生成することができるし、また、両方のビューを単色画像から生成することもできる。２つのビューは異なるｙ解像度を持つことができるが、各ビューのｙ解像度は、バッファのｙ解像度を１から１６の間の整数で除したものでなければならない。また、２つのビューのｘ解像度は異なるものとすることができるが、それらｘ解像度は、バッファにある画像のｘ解像度の１／４から２倍である。また、ビューの各々は、異なるマトリックス、階調マップ、データ・タイプ等々を持つことができる(後述するような制約はある)。

【００２９】

２つのビューは、生成されると、ｙ解像度の比率に従って、行単位でインターリーブされる形態で送出される。例えば、ビュー１のｙ解像度が３００でビュー２のｙ解像度が１５０であれば、データはビュー１の画像データの２行が送られると、次にビュー２の１行が送られる(開始時点を除く)。以下の表１は、種々のｙ解像度比率の場合のインターリーブされるデータの例である。

【００３０】

【表１】

(2対1比率)→300対150出力比:300,150,300,300,150,300,300....

(1対2比率)→300対150出力比:300,150,300,300,150,300,300....

(3対1比率)→300対100出力比:300,300,100,300,300,300,100...

(1対3比率)→100対300出力比:300,100,300,300,300,100,300,300...

(4対1比率)→300対75出力比:300,300,75,300,300,300,300,75,300...

(1対4比率)→75対300出力比:300,75,300,300,300,300,75,300,300...

(5対1比率)→300対60出力比:300,300,300,60,300,300,300,300,300,60...

(1対5比率)→60対300出力比:300,300,60,300,300,300,300,300,60,300...

注：上記は単に典型的例にすぎない。その他１対９の比率まで実施可能である。

【００３１】

本発明の画像ポストプロセッサによって実行される５つの主要機能を図１および図３を参照して以下に記述する。

1.行合併器

行合併器は、該当する画像(カラーまたは白黒)から１ないし９行を読み取り、いくつかの行を平均化して、マトリックス器へ送る。行合併器は、走査された画像の複数の描画を、ホスト・コンピュータに送られるべき１つのデータ・ストリームにインターリーブする。走査された画像の複数の描画を行毎にインターリーブすることによって、すべての描画が生成され、インターリーブされてホストへリアルタイムすなわち行毎に送られるので、スキャナは必ずしもすべての描画を保存しなくてもよい。

【００３２】

行合併器は、カラー画像または単色画像から１かないし９行のデータを読み取り、行のいくつかを平均して、マトリックス器によって実行されなければならない計算処理の数を減少させる。加えて、行合併器は、追加の隣接行を平均することによって単純な形式のｙ解像度事前縮尺を提供することができる。例えば、行合併器は、バッファから９行を読み取り、各３隣接行を平均して３というファクタでｙ解像度を有効に減少させ、中心行対称のため残りの外側の２行を平均して、結果として生成した２行をマトリックス器へ送る。従って、この場合マトリックス器によって処理されなければならない行数を９から２へ減少させた。

【００３３】

(Ｎをバッファから読みとられる９までの行数として)１からＮまでのいかなる隣接行数も一緒に平均することができる。しかしながら、隣接行が平均された後結果として生成され

る行の数は、１、３、５、７または９であり、その左右対称行は、マトリックス器に渡される前に更に平均される。これは、隣接行が平均されない場合１、３、５、７または９のいずれかの中心行を使用することができることを意味する。しかし、２または３の隣接行が平均されるとすれば、１または３行の中心行だけを使用することができる。４ないし９の隣接行が平均されれば、１つの中心行だけを使用することができる。

【００３４】

行は、各行の対応するピクセルを加算して１、２、４または８の縮尺係数で除することによって平均される。従って、１、２、４または８行を平均する時にのみ正確な平均が得られる。縮尺係数で除された加算行数が２を超える場合、行合併器は桁あふれする。従って、３、５、６、７または９行の平均が求められるとすれば、最も近い小さめの縮尺係数が使用されるべきであり、マトリックス係数は、行合併器において出される余剰「利得」を埋め合わせるように減少されなければならない。この規則に対する唯一の例外は、バッファから９行を読み取り、各隣接３行を平均し、次に中心行３を使用する場合である。この場合、行合併器は、９行を２行に減少する。これらの２行のうち１つ(すなわち中心に対応する行)は、バッファからの中央３行を加えることによって得られる。もう１つの行は、バッファからの外側６行を加えることによって得られる。しかし、行合併器から出力される行が２を越える数で乗じられないように、ただ１つの除数を選択することができるだけである。従って、この例では、４という除数が選択されなければならない。この例では行合併器の利得は３／４であるので、マトリックス器係数は、埋め合わせのため増加される必要があるかもしれない。

【００３５】

使用されるべきバッファ画像(カラーまたは単色)、その画像から読み取られる行の数、平均される隣接行の数およびスケール係数は、生成されるべきビュー各々毎に独立して指定することができる。

【００３６】

2.マトリックス器

マトリックス器は、各ピクセルの３つのカラーを混合して階調マップ器に行を送る前に単色または３色を生成する。

【００３７】

3.階調マップ器

階調マップ器は、参照テーブルおよび補間を使用して、画像コントラストおよび強度を調整して、その後フォーマッタにデータを送る。出力されるピクセルは階調マップ器によって更に変換され、階調マップと呼ばれる移転曲線を含む参照テーブルを通過する。階調マップは、ガンマ補正の実行、コントラストおよび強度の調整、陰影の強化などを行うため使用される。

【００３８】

4.フォーマッタ

フォーマッタは、指定されたデータ幅に納める前に、データを選択的にしきい値にかける。この段階は、また、圧縮エンジンに送る前に、必要に応じてデータを反転させることができる。

【００３９】

5.圧縮エンジン

圧縮エンジンは、圧縮アルゴリズムを使用して、データをホストに送る前に、画像データ行を選択的に圧縮する。

【００４０】

II.単一走査ウィンドウから複数ビューの生成

本発明の複数画像スキャナは、単一走査から２つの画像を取得する能力を持つ。複数画像の走査コマンド言語(略してＳＣＬと呼ぶ)の実施は、走査ウィンドウの概念をそのウィンドウの特定のビューと切り離す。ＳＣＬはスキャナのために使用されるコマンド言語であり、一般によく知られている。どのようなコマンド言語でも使用することができる点に留

意する必要がある。しかし、本発明の好ましい実施形態においては、ＳＣＬは１つの走査ウィンドウから複数のビューを生成するために使用される。走査ウィンドウは、(ｘおよびｙ位置および範囲によって規定され)走査されるページの物理的領域である。走査ウィンドウのビューは、物理的領域の内側に含められ(解像度、データ・タイプ、ミラー、反転等々によって定義される)データである。ウィンドウおよびビューとは独立した(ＡＤＦ使用、連番などの)スキャナ設定項目がある。ビューおよびウィンドウを分離する利点は、複数のビューの定義を複数のビューを含む複数のウィンドウに拡張することができる点である。本発明以外のいかなるスキャナも、複数画像能力または複数のビューの実施形態を備えていない。注意されるべき点ではあるが、複数のビューという本実施形態において、単一の走査を使用して同一ドキュメントの複数のビューを取得しているが、単一または複数の走査を使用して同一ドキュメントの複数のビューを取得するか否かは問題ではない。データが(単一または複数の走査から)どのように取得されようとも、重要なのは、１つ以上のビューというＳＣＬ実施形態によって異なるデータが同じ空間を表現することが可能にされている点である。

【００４１】

本発明の１つの好ましい実施形態では、ヒューレット・パッカード社のスキャンジェット５ｐ(ScanJet 5p)が利用され、データの２つのビューの同時取得が可能とされる。しかし、どのような製造業者のどのようなカラー・スキャナでも本発明を実施することができるように修正することができることは留意されるべきである。走査の区域が同じものであるが、２つのビューのデータ・タイプおよび解像度は変わることができる。２つのビューに対して設定される変数に関する制限は、ＳＣＬにおけるSetViewTypeマクロを用いて次の表２のように記述される。

【００４２】

【表２】

```
INT16 DualScan(INT16 Phase,
      PUINT8 pBufferView0,
      INT32 LengthView0,
      PINT32 pReceivedView0,
      PUINT8 pBufferView1,
      INT32 LengthView1,
      PINT32 pReceivedView1);
```

パラメータは次のように設定される。

Phase -- これがシーケンスの最初の伝送であるかどうかを示すフラグ。次の値のいずれかでなければならない。SL_FIRST or SL_SCAN：伝送における最初のバッファ,SL_ADF_SCAN：ＡＤＦ伝送における最初のバッファ(現時点ではHP 1750AおよびHP5110Aによってのみサポートされている),SL_NEXT：各追加バッファ伝送;

PbufferView0 -- ビュー０データ・バッファへのポインタ;

LengthView0 -- ビュー０データ・バッファのサイズ;

pReceivedView0 -- ビュー０に関してスキャナから実際に送られるバイト数のポインタ;

pBufferView1 -- ビュー１データ・バッファへのポインタ;

LengthView1 -- ビュー１データ・バッファのサイズ;

pReceivedView1 -- ビュー１に関してスキャナから実際に送られるバイト数のポインタ;

【００４３】

この関数は、スキャナの２つのビューモードをサポートしている点を除いて、ＳＣＬに関してよく知られているScan()関数に類似している。

【００４４】

単一走査の２つのビューに関するデフォルト設定項目のいくつかを下記表３に示す。

【００４５】

【表３】

パラメータ	デフォルト：画像１	デフォルト：画像２
・データ・タイプ	0(単色しきい値)	5(カラー)
・データ幅	1ビット／ピクセル	24ビット／ピクセル
・単色ディザ・パターン	0(粗く太く)	0(粗く太く)
・係数マトリックス	2(緑色のみ)	0(NTSCカラー)
・階調マップ	0(コントラスト／強度)	0
・強度	0	0
・コントラスト	0	0
・ミラー	0(オフ)	*
・自動背景	0(オフ)	*
・反転画像	0(オフ)	0(オフ)
・Ｘ解像度	300DPI	100DPI
・Ｙ解像度	300DPI	100DPI
・Ｘ縮尺因子	100%	100%
・Ｙ縮尺因子	100%	100%
・Ｘ位置	0	*
・Ｙ位置	0	*
・Ｘ範囲	2550(ピクセル)	*
・Ｙ範囲	3300(ピクセル)	*
・ランプ	0(オフ)	*
・走査バー位置	その場で停止	*
・ダウンロード・タイプ	0(単色ディザ・パターン)	0(単色ディザ・パターン)
・ダウンロードされる 8X8単色ディザ・パターン	なし(旧パターン削除)	*
・ダウンロードされる８ビット階調マップ	なし(旧マップ削除)	*
・ダウンロードされるカラー・		

マトリックッス	なし(旧マトリックス削除)*	
・ダウンロードされる調整ストリップ・パラメータ	なし(旧パラメータ削除) *	
・ダウンロードされる16X16単色ディザ・パターン	なし(旧パターン削除) *	
・ダウンロードされる単色マトリックッス	なし(旧マトリックス削除)*	
・ダウンロードされる１０ビット・カラー・マトリックッス	なし(旧パラメータ削除) *	
・ダウンロードされるＲＧＢ階調マップ	なし(旧パラメータ削除) *	
・ダウンロードされる単色階調マップ	なし(旧パラメータ削除) *	
・ダウンロードされるＲＧＢ利得	なし(旧パラメータ削除) *	
・調整Ｙ開始	-1(白ストリップ)	*
・調整ストリップ・パラメータ	0(自動-選択)	*
・調整モード	0(自動-データ幅依存)	*
・速度モード	0(自動)	*
・圧縮	0(オフ)	0(オフ)
・ビュー数選択	1	該当せず
・選択されるビュー	1(オフ)	該当せず
・調整ストリップ低位	0(計算済み)	*
・最高行数／バッファ	0	*

【００４６】

上記表３において、*は、そのパラメータが２つのビューに対して別々に設定されないことを示す。しかし、２つのビューに対して別々に設定することができるように修正することはできる。また、これらのパラメータ設定およびＳＣＬコマンドは本発明の実施形態を反映しているが、本発明の単一走査概念に基づいて複数のビューを可能にするパラメータ設定またはＳＣＬコマンドの可能な組合せにだけ限定するようには意図されていない。

【００４７】

スキャナによってホスト・コンピュータへ送られるべきビューの数を選択する以下のコマンドが追加された。

コマンド：ビューの数選択
エスケープ・シーケンス：Esc*f#A
照会ＩＤ：10466 範囲：1-2
デフォルト：1
マクロ：SetNumberofViews(x)
送付コマンド：SendCommand(SL_NUM_OF_VIEWS, x)
但しxは1または2。
【００４８】
このコマンドは、スキャナによって送られるビューの数を選択する。デフォルトは１である。本実施形態がサポートする値は、1=１つのビュー、2=２つのビュー、である。２つのビューが使用可能にされると、２つのビューに関するデータが、それらのＹ解像度に応じて行単位でインターリーブされて送られる。例えば、ビュー１のＹ解像度が３００でビュー２のＹ解像度が１５０であれば、上述のように、(最初を除いて)ビュー１が２行送られると次にビュー２が１行送られるというように交互に伝送される。
【００４９】
以下のようにビュー選択コマンドを使用して、動作させたい所望のビューに関してスキャナの設定を行うことができる。
コマンド：ビュー選択
エスケープ・シーケンス：Esc*f#B
照会ＩＤ：10467
範囲：0-1
デフォルト：0
マクロ：SetViewType(x)
送出コマンド：SendCommand(SL_VIEW_TYPE, x)
但し、xは、ビュー１に対しては0、ビュー２にに対しては1である。
【００５０】
ビューは、スキャナ設定を選択する前に選択されなければならない。このコマンドは、後続のＳＣＬコマンドまたは照会がどのビューを参照するかを選択する。サポートされる値は、0(ビュー１)および1(ビュー２)である。デフォルトはビュー１である。
【００５１】
III.複数画像走査データ解析方法
ホスト・コンピュータ・ソフトウェアは、スキャナによって送られたインターリーブされた複数画像データを解析する。本実施形態において、ソフトウェア・プログラムDualScanおよびそのサポート・モジュールは、複数画像データ形式の詳細の知識を持たないユーザがフォトショップ(Photo Shop)のようなプログラムで使用できるようにスキャナからの複数画像データを解析して周知のＴＩＦＦファイルに変換することを可能にする。DualScanはヒューレット・パッカード社から販売されているソフトウェア・プログラムである。画像の一方または両方が解析問題を一層複雑にする圧縮データを含む可能性がある。DualScanは、スキャナから受け取る(インターリーブされた)データストリームを解析することによって、ハードウェアとして複数画像機能を持つスキャナの使用を可能にする。本実施形態に記述される２つのビューではなく、複数のビュー・データの解析を含むようにDualScanを拡張することは当業者にとって容易であろう。
【００５２】
図４は、２画像データストリームのブロック図であって、データストリーム１３６がスキャナ(図示されていない)から送られ、データストリーム解析機構１２２によって解析されてホスト・コンピュータ(図示されていない)上のビュー１データ(単色画像)１２６およびビュー２データ(カラー画像)１２４が生成される。データ・ストリーム解析機構の動作の詳細は図５に流れ図として示されている。
【００５３】
初期的に、ステップ２００において、画像の各々について解像度、圧縮状態および行バイ

ト数がデータ解析機構によって照会される。次に、各ビューに関する画像比率がデータ解析機構によって計算される(ステップ２０２)。次に、ステップ２０４において、インターリーブされた画像データがデータ解析機構によってスキャナから受け取られる。ステップ２０６において、データ解析機構は、走査データの１行を適切な出力バッファにコピーする(必要に応じてそのデータを伸張する)。次に、データ解析機構は、ステップ２０２で計算した画像比率に基づいてビューを切り替える(ステップ２１０)。エラーが検出されずかつデータストリームの解析が完了していなければ(ステップ２１２)、データ解析機構はステップ２０４に戻り、更に解析すべきデータをスキャナから受け取る。エラーが検出されるかあるいはデータストリームの解析が完了してれば(ステップ２１２)、データ解析機構は、適切な措置を講ずる呼び出し元関数にこの状態を返す。エラー条件には、(1)出力バッファ容量不足、(2)スキャナからの不具合応答、(3)伸張エンジンおよびスキャナにデータなし、または(4)走査完了が含まれる。図５の処理ステップの詳細は、後述の付録Ａに原始コードとして示されている。

【００５４】

本明細書に記述されている実施形態は、２つの走査、２つのビューおよび２つのインターリーブされたデータストリームの解析を記述しているが、これらの概念は、複数の走査、複数のビューおよび複数のインターリーブされたデータストリームの解析に容易に拡張することができる点は留意されるべきである。

【００５５】

本発明の上記記述は、例示の目的のために提示されたものであって、本発明を上述されたものに限定するように意図されてはいないし、上記実施形態の種々の修正および変形は可能である。例えば、データストリーム解析機構は、複数セットのデータを含むインターリーブされたデータストリームを解析する能力を有している。データストリームは、走査された画像データストリームである必要はない。また、同一ウィンドウの複数のビューのＳＣＬ実施形態は複数のビューがどのように取得されるかは関係ない。ビューが同一領域の複数の走査から取得される場合でも実施形態は同じである。上記実施形態は発明の原理およびその実際的応用を最もよく説明するため選択され記述されたものであって、これによって、当業者が、特定の使用に合致するように種々の実施形態および種々の修正の形態で本発明を利用することが可能であろう。

【００５６】

【表４】

付録A

```
INT16 FP DualScan(INT16 Phase,
          PUlNT8 pBufferView0,INT32 LengthView0,PINT32 pReceivedView0,
          PUINT8 pBufferView1,INT32 LengthView1,PINT32 pReceivedView1)
/*
** パラメータ：
** Phase    シーケンスの最初の伝送であることを示すフェーズ・フラグ。
**          このフラグの値は次のいずれかである。
**          SL_FIRSTまたはSL_SCAN，伝送における最初のバッファ。
**          SL_ADF_SCAN，ADF伝送における最初のバッファ。
**          SLNEXT，各追加バッファ伝送。
** pBufferView0  データを受けるためビュー０データ・バッファを
**               指し示すポインタ。
** LengthView0   ビュー０データ・バッファのサイズ(バイト)。
** pReceivedView0  ビュー０に関してスキャナから受け取る実際のバイト数を
**                 指し示すポインタ。
** pBufferView1    データを受けるためビュー１データ・バッファを指し示す
**                 ポインタ。
** LengthView1     ビュー１データ・バッファのサイズ(バイト)。
** pReceivedView1  ビュー１に関してスキャナから受け取る実際のバイト数を
**                 指し示すポインタ。
**大域変数：なし
** 動作：SL_ADF_SCANはHP1750AおよびDominoによってだけサポートされる。
** このプロシージャはスキャナの２つのビューモードをサポートする点を除い
** て"Scan()"関数とほとんど同一である。
**  このコマンドは、"Domino"のような２つのビューをサポートするスキャナに
**  関してのみサポートされている。
**
**  注：このルーチンは、また、圧縮された入力データを取り扱うことができる
```

```
**。
** 戻り値：
**   status(状態)  OKAY スキャナへのコマンド送付成功。
**                 SL_BADACCESS 伝送エラー。
**                 SL_BADRESPONSE 照会への応答不正。
**                 SL_NULLRESPONSE 照会への応答はヌル。
**                 SL_ILLEGALCMD スキャナへ不適当なコマンド送付。
**                   (スキャナは当該ＳＣＬコマンドをサポートしていない)
**。
**                 SL_SCANERROR 圧縮走査エラー。
*/
{
INT16 status = OKAY;
INT16 num_views;   /* 現在活動的なビューの番号 */
INT16 active_view; /* このルーチンへの入り口における */
                   /* 活動的ビュー(０または１) */
INT32 received = -1; /* スキャナから実際に受け取るデータ量 */
                     /* (検査のため初期値-1) */
INT16 YRes_view0,    /* ビュー０および１に関するＹ解像度 */
   YRes_view1;
INT16 ImageRatio_view0, /* ビュー０および１に関する画像比率 */
   ImageRatio_view1;
static INT16 Compression_view0, /* ビュー０および１に関する */
   Compression_view1;           /* 走査データ圧縮フラグ */
static INT16 WidthBytes_view0,  /* ビュー０および１に関する */
    WidthBytes_view1;           /* 単一走査行のバイト幅 */
static INT16 cur_view;       /* 現在処理されているビュー */
static INT16 view_count[2];
static INT16 imageratio[2];
```

```
static PUlNT8 pScanBuffer;  /* 静的に割り当てられた入力走査バッファ */
                            /* における現在位置へのポインタ */
static INT16 bytes_avail;   /* 入力バッファにおける利用可能バイト数 */
/* ２画像走査処理の確認 */
if (Phase != SL_NEXT)
{
status = InquireNumber(SL_VIEW_TYPE, SL_NOM;(PINT16)&active_view);
if (status != OKAY)
    return (SL_ILLEGALCMD);
/* ビュー#0(第１のビュー)に対して適用できる走査状態情報を取得する */
SetViewType(0);
if ((status=InquireNumber(SL_NUM_OF_VIEWS,SL_NOM&num_views)!=
   OKAY) || (num_views != 2) ||
   (status=InquireNumber(SL_Y_RBSOLUTION,SL_NOM,&YRes_view0) != OKAY)
   || (status = InquireNumber(SL_COMPRESSION,SL_NOM,&Compression_view0)
   != OKAY) ||
   (status = InquireNumber(SL_BYTES_PER_LINE,SL_NOM,&WidthBytes_view0)
   != OKAY))
      return (SL_ILLEGALCMD);
/* ビュー#1(第２のビュー) に対して適用できる走査状態情報を取得する */
SetViewType(1);
if ((status = InquireNumber(SL_Y_RESOLUTION,SL_NOM,&YRes_view1)
   != OKAY) ||
   (status = InquireNumber(SL_COMPRESSION,SL_NOM,&Compression_view1)
   != OKAY)) ||
   (status = InquireNumber(SL_BYTES_PER_LINE,SL_NOM,&WidthBytes_view1)
   != OKAY))
      return (SL_ILLEGALCMD);
   /* Ｙ解像度を使用しているビュー各々に関してＹ画像比率を計算する */
```

```
   if (YRes_view0 <= YRes_view1)
   {
      ImageRatio_view0 = 1;
      ImageRatio_view1 = YRes_view1/YRes_view0;
#ifdef VORTEX_WORKROUND
    view_count[0] = 0;
    view_count[1] = ((INT16)((ImageRatio_view1/2) + 1))
                    %ImageRatio_view1;
if (view_count[1]==0)  /* 特別ケース：１：２比率 */
      cur_view = 0; /* 開始のビューを設定 */
   else cur_view = 1;
#endif
      }
      else
      {
          ImageRatio_view0 = YRes_view0/YRes_view1 ;
          ImageRatio_view1=1;
#ifinddef VORTEX_WORKROUND
           view_count[0] = (INT16)(ImageRatio_view0/2);
           view_count[1] = 0;   /* 開始ビュー設定 */
           cur_view = 0
#endif
     }
#ifdef VORTEX_WORKROUND
 /* (サポートしている画像比率に関して)ビュー出力カウンタを初期化する */
 view_count[0] = 1; /* =ImageRatio_view0-1; */
                /* (最終的Vortexリリースに関して) */
 view_count[1] = ImageRatio_view1;
 cur_view = 0;   /* 開始ビュー設定 */
```

```
#endif
 imageratio[0] = ImageRatio_view0;
 imageratio[1] = ImageRatio_view1;
/* このルーチンに入る時、ビューを初期活動ビューにリセットする */
 SetViewType(active_view);
/* 走査入力バッファへのポインタを初期化する */
 pScanBuffer = (PUINT8)&ScanBuffer;
}  /* if(Phase != SL_NEXT) */
/* 両方のビューに関して受け取る実際のバイト数を初期化する。
* 注：これらの２つの関数戻り引数がこのルーチン内で初期化されるが、
*     それらは、このルーチンによって呼び出されるもう１つの
*     関数CopyScanLine()内で実際に設定される。
*/
  *pReceivedView0=0;
  *pReceivedView1=0;
/* スキャナが通信中でありエラー状態をクリアしていることを確認する */
 if (Phase==SL_NEXT
    ||((status = InquireOldestError0) != SL_BADACCESS && status !=
      SL_BADRESPONSE && status != SL_NULLRESPONSE && (status =
      SendCommand(Phase,0))==OKAY))
 {
  if ((Phase==SL_FIRST) || (bytes_avail==SCAN_BUFSIZE))
  {  /* バッファに記憶 */
        bytes_avail=SCAN_BUFSIZE;
        status=SCANBUFEMPTY;
  }
  while ((status != OUTBUFFERFULL) && (status != SL_SCANERROR) &&
        (received != 0))
  {
```

```
    /* 必要なら走査入力バッファに再記憶 */
    if (status==SCANBUFEMPTY)
    {
     memmove(ScanBuffer,pScanBuffer,SCAN_BUFSIZE-bytes_avail);
     pScanBuffer=&ScanBuffer[SCAN_BUFSIZE-bytes_avail];
     received = RecFromScanner(pScanBuffer,bytes_avail,SL_ALLCHAR);
     if (received < 0)  /* 走査データ読み取りエラー */
     {
      *pReceivedView0= 0;
      *pReceivedView1= 0;
      return(SL_BADACCESS);
    }
#ifdef DEBUG_DI
  if (debug_flag) {
     printf("bytes_avail A1= %d¥n",bytes_avail);
     }
#endif
     bytes_avail = bytes_avail - (INT16)received;
/* 利用可能バイト数のカウントをリセットする */
#ifdef DEBUG_DI
       if (debug_flag) {
       printf("bytes_avail A2 = %d¥n" bytes_ avail)
       }
#endif
       pScanBuffer = &ScanBuffer[0];
       status = OKAY;
       } /* if(..) */
       if (bytes_avail==SCAN_BUFSIZE) break;
       /* 適切な入力走査バッファに記憶 */
```

```
        if (cur_view==0)
        {
        status = CopyScanLine(Compression_view0,WidthBytes_view0,
                 &pScanBuffer,&bytes_avail,&pBufferView0,LengthView0,
                 pReceivedView0);
        }
        else /* cur_view1 */
        {
         status = CopyScanLine(Compression_view1,WidthBytes_view1,
                  &pScanBuffer,&bytes_avail,&pBufferView1,LengthView1,
                  pReceivedView1);
        }
        /* プロセスすべき次の画像(ビュー０または１)に切り替える */
        if ((status != SCANBUFEMPTY) && (status != OUTBUFFEREULL)
            && (status != SL_SCANERROR))
        {
#ifdef VORTEX_WORKAROUND
        --view_count[cur_view];
        if (view_count[cur_view]==0)
        {
             view_count[cur_view]=imageratio[cur_view];
#else
        ++view_count[cur_view];
        if (view_count[cur_view]==imageratio[cur_view])
        {
             view_count[cur_view]= 0;
#endif
             cur_view = ((cur_view + 1)%2);
         }
```

```
      }
    } /* while() */
#ifdef DEBUG_DI
   if (debug_flag)
      printf("[DualScan] status = %d; bytes_avail A3=%d¥n",status,
            bytes_avail);
#endif
    /* 行を完了するため伸張エンジンが更にデータを必要としているが  */
    /* するがスキャナが送るべきデータを持っていないことを検査する。*/
    if (((status==SCANBUFEMPTY) && (received==0)) ||
       (status==SL_SCANERROR))
        status=SL_SCANERROR;
    else status = OKAY;
  } /* if(..) */
  return status;
} /* DualScan */
static INT16FP CopyScanLine(INT16 Compression,INT16 WidthBytes,
              PUINT8 *ScanBuffer,PINT16 bytes_avail,PUINT8 *pBuffer,
              INT32 Length,PINT32 pReceived)
/*
** パラメータ：
**    Compression 入力走査データ圧縮フラグ。
**    WidthBytes  単一走査行の幅(バイト)。
**    ScanBuffer  走査入力バッファ内の現在時バイトを含むポインタを
**                指し示すポインタ。
**                [出口において--間接的に参照されたバイト・ポインタは、
**                入力バッファから処理された最後のバイトの直後の
**                バイト・アドレスに設定される。]
**    bytes_avail 入力バッファにおいて利用可能なバイト数を指し示す
```

```
**                  ポインタ(この引数は、この関数が最初に呼び出される時の
**                  SCAN_SIZEに常に等しい)。
**                  [出口において--  このカウンタは、このルーチンによって
**                  成功裡に処理された入力バイト数だけ増分される。]
**    pBuffer       処理されたデータが記憶されるべき出力バッファ内に
**                  おける現在位置を含むポインタを指し示すポインタ。
**                  [出口において--間接的に参照されたバイト・ポインタは、
**                  出力バッファに追加されるべき次のバイトのアドレスに
**                  設定される。]
**    Length        出力バッファ'pBuffer'の全体サイズ(バイト)。
**                  (注：これは'pBuffer'の全体サイズであってバッファに
**                  残る利用可能バイト数ではない。)
**    pReceived     'pBuffer'によって指し示される出力バッファにおいて
**                  検出される実際のバイト数を指し示すポインタ。
**                  [出口において--このカウンタは、出力バッファに現在
**                  あるバイトの総数を含む。]
**
** 大域変数：なし
** 動作：このプライベート関数は、入力バッファ'ScanBuffer'から
**       出力バッファ'pBuffer'へ単一走査行データをコピーする。
** 戻し値：
**   status  OKAY, 出力バッファへの書き込み成功。
**           SCANBUFEMPTY,  入力走査バッファは空である
**                          (入力バッファは、単一の完全な走査行として
**                          十分なデータを持っていない)。
**           OUTBUFFERFULL, 出力走査バッファはいっぱいで
**                          処理された(すなわち２つに分解されたまたは
**                          伸張された)データを格納することができない
**                          (新しい完全な単一走査行を格納できる十分な
```

```
**                          空間を出力バッファは持っていない)。
*/
{
INT16 status = OKAY; /* この関数からの戻り値 */
                     /* (デフォルト=OKAY) */
/*
*  完全な単一走査行データを記憶するのに十分な出力バッファの空間が
*  あるかを検査する。
*/
if ((*pReceived + WidthByies) > Length) status = OUTBUFFERFULL;
else {          /*  出力バッファに空きあり! */
  if (!Compression) {  /*非圧縮データ */
    /* 入力バッファが完全な単一走査行データを含んでいるか検査する。*/
if ((SCAN_BurSIZE - *bytes_avail) < WidthBytes)
     status = SCANBUFEMPTY;
   else{     /* すべてOK--データをコピーする */
        memmove ((PUINT8) *pBuffer,(PUINT8)*ScanBuffer,WidthBytes);
        *ScanBuffer += WidthBytes;
        *bytes_avail += WidthBytes;
        *pBuffer += WidthBytes;
        }
   }
   else {      /* 圧縮データ */
           /* 入力単一走査行を伸張する */
     status = Decomp(ScanBuffer,bytes_avail,WidthBytes,pBuffer);
   }  /* if() else */
   if (status OKAY) {
        *pReceived += WidthBytes;
   }

 }
 return status;
} /* CopyScanLine */
```

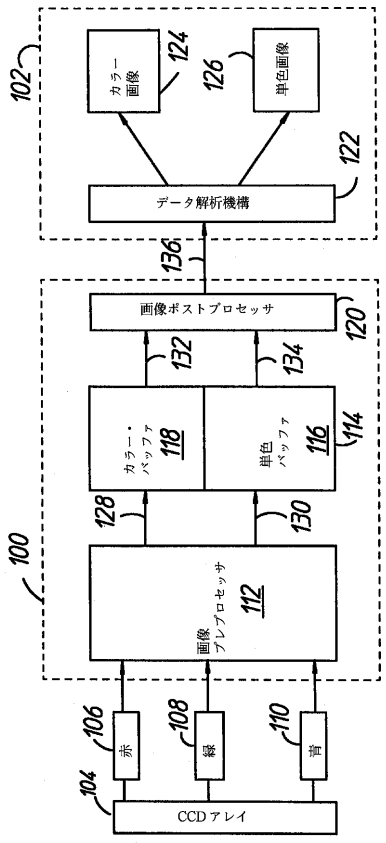
【００５７】

本発明には、例として次のような実施様態が含まれる。

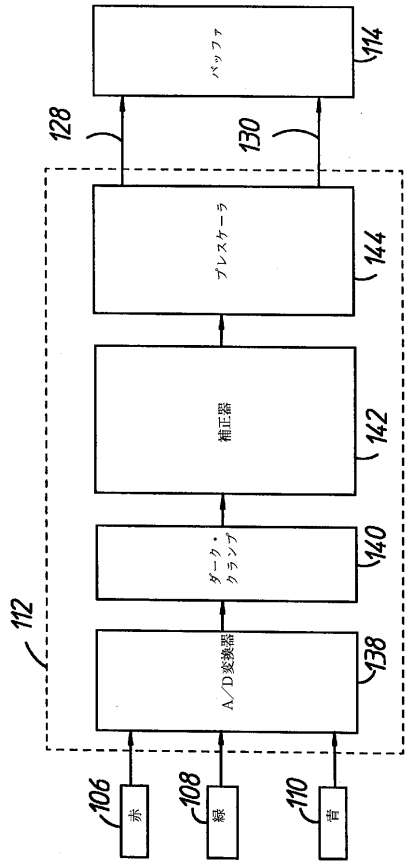
（１）スキャナを使用して１つの画像の複数描画を生成する方法であって、上記スキャナを使用して上記画像の単一走査を実行するステップ(a)と、上記単一走査から上記画像の複数の描画を上記スキャナが生成するステップ(b)と、を含む画像の複数描画生成方法。
（２）上記ステップ(b)が、少くとも１つのグレイスケール画像および少くとも１つのカラー画像を生成するステップを含む、上記（１）に記載の画像の複数描画生成方法。
（３）上記ステップ(b)が、少くとも１つの高解像度グレイスケール画像および少くとも１つの低解像度高ビット・カラー画像を生成するステップを含む、上記（１）に記載の画像の複数描画生成方法。
（４）上記単一走査から生成された上記複数の描画をホスト・コンピュータでの更なる処理のため上記スキャナがホスト・コンピュータへ送信するステップ(c)を更に含む、上記（１）に記載の画像の複数描画生成方法。
【００５８】
（５）単一走査から１つの画像の複数の描画を生成する画像処理装置を備えるスキャナ。
（６）上記画像処理装置が、上記単一走査画像を複数の描画に分割する機能を持つプレスケーラを含む、上記（５）に記載のスキャナ。
（７）上記プレスケーラが、上記単一走査画像の少くとも１つの単色描画および上記単一走査画像の少くとも１つのカラー描画に上記単一走査画像を分割する、上記（６）に記載のスキャナ。
（８）上記プレスケーラが、少くとも１つの高解像度グレースケール画像および少くとも低解像度高ビット・カラー画像に上記単一走査画像を分割する、上記（６）に記載のスキャナ。
【００５９】
（９）１つの画像の複数の描画を生成する方法であって、スキャナを使用して上記画像の単一走査を実行するステップと、上記単一走査画像の複数の描画を表す複数のデータ信号を上記スキャナが生成するステップと、上記単一走査画像の複数の描画を表す上記複数のデータ信号をインターリーブすることによって単一のインターリーブされたデータストリームを上記スキャナが生成するステップと、を含む画像の複数描画生成方法。
（１０）上記単一走査画像の複数の描画の更なる処理のため上記インターリーブされた単一データストリームをコンピュータへ上記スキャナが送信するステップを更に含む、上記（９）に記載の画像の複数描画生成方法。
【００６０】
【発明の効果】
本発明に従うスキャナの利用によって、テキストとグラフィックスのような異なるタイプの画像を含むドキュメントの１回の走査から、例えば高解像度グレースケール画像と低解像度高ビット深度カラー画像という２つの描画画像を同時に出力することが可能となり、それによって、ホスト・コンピュータへ送られそこで処理されるデータの総量が大幅に減少する。
【図面の簡単な説明】
【図１】本発明に従った複数画像スキャナに関するデータ経路のブロック図である。
【図２】本発明に従った複数画像スキャナに関する画像プレプロセッサのデータ経路の一層詳細なブロック図である。
【図３】本発明に従った複数画像スキャナに関する画像ポストプロセッサのデータ経路の一層詳細なブロック図である。
【図４】本発明に従った複数画像スキャナからホスト・コンピュータへのデータ経路の一層詳細なブロック図である。
【図５】本発明に従って、複数セットのデータを含むデータストリームを解析する方法の流れ図である。
【符号の説明】
１００　スキャナ
１０２　ホスト・コンピュータ

１０４　ＣＣＤアレイ
１１２　画像プレプロセッサ
１１４　バッファ
１１６　単色バッファ
１１８　カラー・バッファ
１２０　画像ポストプロセッサ
１２２　データ解析機構
１２４　カラー画像
１２６　単色画像
１２８　カラー画像信号
１３０　単色画像信号
１３６　データストリーム

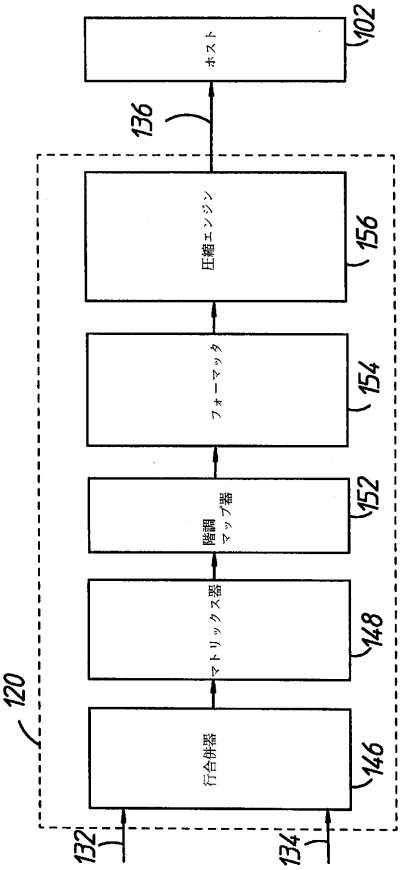
【図１】



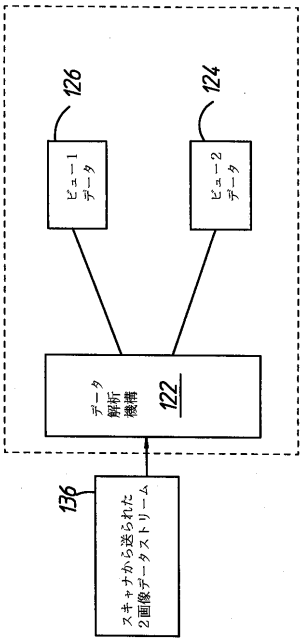
【図２】



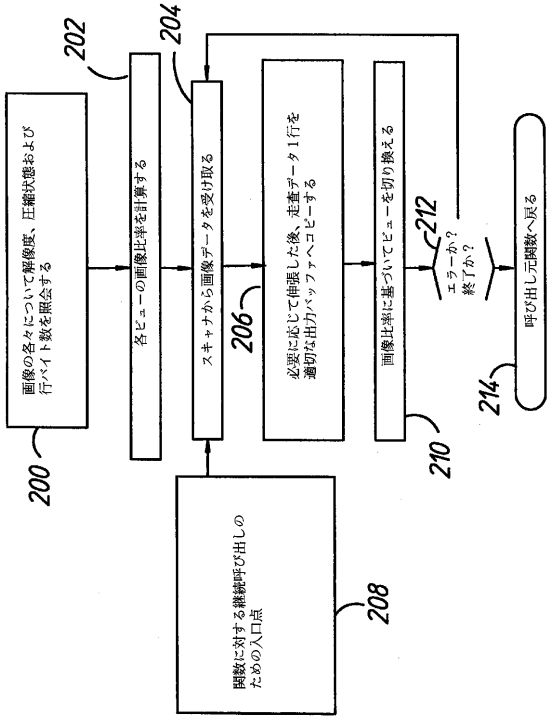
【図３】


ホスト
102
136
圧縮エンジン
156
フォーマッタ
154
階調マップ器
152
マトリックス器
148
行合併器
146
120
132
134

【図４】


126
ビュー1データ
124
ビュー2データ
データ解析機構
122
136
スキャナから送られた2画像データストリーム

【図５】


画像の各々について解像度、圧縮状態および行バイト数を照会する
200
各ビューの画像比率を計算する
202
スキャナから画像データを受け取る
204
206
必要に応じて伸張した後、走査データ1行を適切な出力バッファへコピーする
画像比率に基づいてビューを切り換える
210
エラーか？
終了か？
212
呼び出し元関数へ戻る
214
関数に対する継続呼び出しのための入口点
208

フロントページの続き

(72)発明者 スティーブン・エル・ウェブ
アメリカ合衆国８０５３８コロラド州ラブランド、シルバーリーフ・ドライブ １８２８
(72)発明者 グレッグ・エー・デギ
アメリカ合衆国８０５２５コロラド州フォート・コリンズ、スプリングウッド・ドライブ １３０７
(72)発明者 グレゴリー・エー・ブレイク
アメリカ合衆国８０５２６コロラド州フォート・コリンズ、ウェイクロビン・レーン １３１２
(72)発明者 ケヴィン・ジェイ・ヤンガース
アメリカ合衆国８０６３１コロラド州グリーリー、２８アベニュー １９５１、ナンバー ２９
(72)発明者 ダン・エス・ジョンソン
アメリカ合衆国８０６３４コロラド州グリーリー、エヌ・ウィンダム・アベニュー ４０９

審査官 手島 聖治

(56)参考文献 特開平０２－０３９３８４（ＪＰ，Ａ）
特開平０７－０３８７６８（ＪＰ，Ａ）
特開平０２－１８９０８９（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
H04N1/00
H04N1/04-1/207
H04N1/38-1/409
H04N1/46-1/64
G06T1/00-1/60
G06T3/00-5/50
G06T9/00-9/40