(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2005-503851
(P2005-503851A)
(43)公表日　平成17年2月10日(2005.2.10)

(51)Int.Cl.[7]	F I		テーマコード（参考）
A61B 5/00	A61B 5/00	D	4C027
A61B 5/0432	G06F 17/60	126Q	4C117
A61B 5/0476	A61B 5/04	314A	
G06F 17/60	A61B 5/04	324	

審査請求 有　予備審査請求 未請求　（全 83 頁）

(21)出願番号　特願2003-525783(P2003-525783)
(86)(22)出願日　平成14年7月1日(2002.7.1)
(85)翻訳文提出日　平成16年3月1日(2004.3.1)
(86)国際出願番号　PCT/US2002/020651
(87)国際公開番号　WO2003/021517
(87)国際公開日　平成15年3月13日(2003.3.13)
(31)優先権主張番号　60/316,602
(32)優先日　平成13年8月31日(2001.8.31)
(33)優先権主張国　米国(US)
(31)優先権主張番号　10/107,229
(32)優先日　平成14年3月27日(2002.3.27)
(33)優先権主張国　米国(US)
(81)指定国　EP(AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,IE,IT,LU,MC,NL,PT,SE,SK,TR),CA,JP

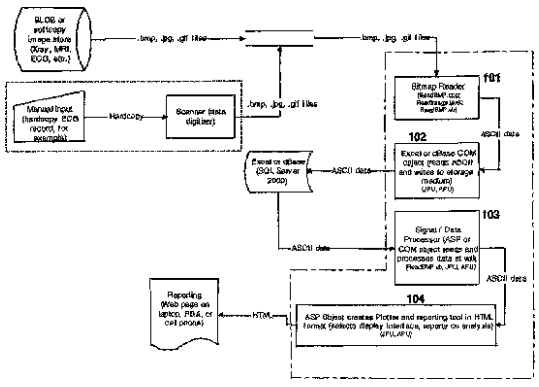
(71)出願人　502144899
シーメンス　メディカル　ソルーションズ　ヘルス　サーヴィシズ　コーポレイション
アメリカ合衆国　ペンシルヴァニア　マルヴァーン　ヴァレー　ストリーム　パークウェイ　５１
(74)代理人　100088454
弁理士　加藤　紘一郎
(72)発明者　ザレンスキ，ジョン，アール
アメリカ合衆国　ペンシルベニア州　１９３２０　ウエスト・ブランディワイン　エルムウッド・レーン　２１９

Fターム(参考)　4C027 AA02 AA03 FF01 HH08 HH11 KK03 KK05

最終頁に続く

(54)【発明の名称】画像表示データの処理システム

(57)【要約】
本発明は、収集した患者データの線グラフ表示をデジタル信号処理に適したデジタルデータへ変換するシステムであって、収集した患者データの線グラフ表示を含む画像を表すデータを受け、受けたデータを分析して画像表示データ中の線グラフ表示を構成する個々のピクセルを同定し、基準点に関する個々のピクセルの相対的位置を特定する複数のピクセル表示データ要素を所望のデータフォーマットで発生させ、個々のピクセルの相対的位置を特定し、（ａ）個々のピクセルのカラー及び（ｂ）個々のピクセルの輝度振幅のうち少なくとも１つを特定することにより所望のデータフォーマットを発生させ、発生させたピクセル表示データ要素をファイルに蓄積するステップより成る方法に係る。さらに、受けたデータをフォーマットして、線グラフ表示を表すデータを処理して線グラフ表示を表示画像の形で提示するに適した処理済みデータに変える遠隔場所での処理手順に適合するフォーマットを発生させ、ユーザー命令に応答してフォーマット済みデータを遠隔場所に送り、フォーマット済みデータを用いて遠隔場所で表示画像を発生させる。



【特許請求の範囲】
【請求項１】
収集した患者データの線グラフ表示をデジタル信号処理に適したデジタルデータへ変換する方法であって、
収集した患者データの線グラフ表示を含む画像を表すデータを受け、
受けたデータを分析して画像表示データ中の線グラフ表示を構成する個々のピクセルを同定し、
基準点に関する個々のピクセルの相対的位置を特定する複数のピクセル表示データ要素を所望のデータフォーマットで発生させ、
個々のピクセルの相対的位置を特定し、（ａ）個々のピクセルのカラー及び（ｂ）個々のピクセルの輝度振幅のうち少なくとも１つを特定することにより所望のデータフォーマットを発生させ、
発生させたピクセル表示データ要素をファイルに蓄積するステップより成る、線グラフ表示をデジタルデータへ変換する方法。
【請求項２】
線グラフ表示の複写紙をスキャンして線グラフ表示を含む画像を表すデータを得るステップを含む請求項１の方法。
【請求項３】
線グラフ表示を含む画像を表すデータは、（ａ）ビットマップファイルフォーマット、（ｂ）ＧＩＦフォーマット、（ｃ）ＪＰＥＧフォーマット、及び（ｄ）ＴＩＦＦフォーマットのうちの少なくとも１つのデータフォーマットの形態である請求項１の方法。
【請求項４】
所望のデータフォーマットは、(x , y, color)iより成るＡＳＣＩＩフォーマットであり、(x)はi番目のピクセルのx座標位置を表し、(y)はi番目のピクセルのy座標位置を表し、colorはi番目のピクセルＲＧＢカラーを表す請求項１の方法。
【請求項５】
医学的パラメータの線グラフ表示は、（ａ）心電図記録、（ｂ）脳電図記録、（ｃ）呼吸速度、１回呼吸器量、分時呼吸量、脈拍酸素飽和度などより成る群から選択される少なくとも１つの呼吸パラメータ記録、及び（ｄ）血液パラメータ記録のうちの少なくとも１つより成る請求項１の方法。
【請求項６】
受けたデータをフォーマットして、線グラフ表示を表すデータを処理して線グラフ表示を表示画像の形で提示するに適した処理済みデータに変える遠隔場所での処理手順に適合するフォーマットを発生させ、
ユーザー命令に応答してフォーマット済みデータを遠隔場所に送り、
フォーマット済みデータを用いて遠隔場所で表示画像を発生させるステップをさらに含む請求項１の方法。
【請求項７】
データフォーマットを復号して線グラフ表示データを運ぶデータフィールドを組み込んだ復号済みフォーマットを発生させ、
線グラフ表示データを運ぶデータフィールドにアクセスする処理手順の実行を始動するステップをさらに含み、その処理手順はフォーマット済み及び符号化済みデータへのアクセスを可能にする特注のセキュリティープロトコル設定の確立を除外し、その処理手順は線グラフ表示データを処理して線グラフ表示を表示画像で提示するに適した処理済みデータを与える請求項１の方法。
【請求項８】
線グラフ表示をデジタル信号処理に適したデジタルデータへ変換する方法であって、
線グラフ表示を含む画像を表すデータを受け、
受けたデータを分析して画像表示データ中の線グラフ表示を構成する個々のピクセルを同定し、

カラーを特定するユーザーカラー選択情報を受け、
基準点に関する個々のピクセルの相対的位置を表す特定カラーのピクセルより成る複数のピクセル表示データ要素を所望のデータフォーマットで発生させ、
発生させたピクセル表示データ要素をファイルに蓄積するステップより成る線グラフ表示をデジタルデータへ変換する方法。
【請求項９】
蓄積ステップは、（ａ）テキストファイルフォーマット、（ｂ）ワードファイルフォーマット、（ｃ）スプレッドシートフォーマット、及び（ｄ）データベース適合フォーマットを含むフォーマットからユーザーが選択するフォーマットのファイルに発生させたピクセル表示データ要素を蓄積するステップより成る請求項９の方法。
【請求項１０】
ファイルヘッダー情報から受けたデータのファイルサイズを突き止め、
突き止めたサイズのファイルの２進読み取りを行うことにより前記発生ステップに用いるピクセルデータを抽出するステップをさらに含む請求項９の方法。
【請求項１１】
ピクセルデータを１ピクセルにつき３つの８ビットセグメントで表す生の２進データを抽出し、
抽出したピクセルデータをユーザーが特定する所望のカラーマップと比較して所望のカラーのピクセルに対応するピクセルを選択的にフィルタリングし、
複数のピクセル表示データ要素を発生させる前記ステップに用いる所望のフォーマットでフィルタリング済みの抽出ピクセルデータの座標を発生させるステップをさらに含む請求項９の方法。
【請求項１２】
線グラフ表示のセグメントを表す発生させたピクセル表示データ要素を分析して線グラフ表示セグメント中の特徴部の頻度をチェックするステップをさらに含む請求項９の方法。
【請求項１３】
線グラフ表示は医学的パラメータを表し、医学的パラメータの頻度特性の変化を予想するステップを含む請求項９の方法。
【請求項１４】
医学的パラメータの線グラフ表示は、（ａ）心電図記録、（ｂ）脳電図記録、（ｃ）呼吸パラメータ記録、及び（ｄ）血液パラメータ記録のうちの少なくとも１つより成る請求項１４の方法。
【請求項１５】
医学的パラメータの頻度特性の予想される変化が特定の患者にとって統計的に意義あるものか否かを判定するステップをさらに含む請求項１４の方法。
【請求項１６】
医学的パラメータの頻度特性の予想される変化が、特定の患者の医学的記録情報に基づき統計的に意義あるものか否かを判定するステップをさらに含む請求項１６の方法。
【請求項１７】
データフォーマットを復号して線グラフ表示データを運ぶデータフィールドを組み込んだ復号済みフォーマットを発生させ、
線グラフ表示データを運ぶデータフィールドにアクセスする処理手順の実行を始動するステップをさらに含み、その処理手順はフォーマット済み及び符号化済みデータへのアクセスを可能にする特注のセキュリティープロトコル設定の確立を除外し、その処理手順は線グラフ表示データを処理して線グラフ表示を表示画像で提示するに適した処理済みデータを与える請求項９の方法。
【請求項１８】
前記発生ステップは、受けたデータをＨＴＭＬアクティブサーバーページフォーマットでタグフィールドにフォーマットするステップより成る請求項９の方法。
【請求項１９】

線グラフ表示をデジタル信号処理に適したデジタルデータへ変換するシステムであって、
（１）線グラフ表示を含む画像を表すデータを受け、
（２）受けたデータを分析して画像表示データ中の線グラフ表示を構成する個々のピクセルを同定し、
（３）カラーを特定するユーザーカラー選択情報を受け、
（４）基準点に関する個々のピクセルの相対的位置を特定する特定カラーの複数のピクセル表示データ要素を所望のデータフォーマットで発生させるようにプログラムされた画像読み取り装置と、
発生させたピクセル表示データ要素をファイルに蓄積するようにプログラムされたデータオブジェクトとより成る線グラフ表示をデジタルデータへ変換するシステム。

【請求項２０】
データオブジェクトは、（ａ）テキストファイルフォーマット、（ｂ）ワードファイルフォーマット、（ｃ）スプレッドシートフォーマット、及び（ｄ）データベース適合フォーマットを含むフォーマットからユーザーが選択するフォーマットのファイルに発生させたピクセル表示データ要素を蓄積するようにプログラムされている請求項２１のシステム。

【請求項２１】
画像読み取り装置はさらに、ファイルヘッダー情報から受けたデータのファイルサイズを突き止め、突き止めたサイズのファイルの２進読み取りを行うことにより前記発生ステップに用いるピクセルデータを抽出するようにプログラムされている請求項２１のシステム。

【請求項２２】
データ要素を復号して線グラフ表示データを運ぶデータフィールドを組み込んだ復号済みフォーマットを発生させるようにプログラムされた信号プロセッサと、
線グラフ表示データを運ぶデータフィールドにアクセスするための処理手順の実行を始動させるようにプログラムされた画像形成オブジェクトとをさらに備え、その処理手順はフォーマット済み及び符号化済みデータへのアクセスを可能にする特注のセキュリティープロトコル設定の確立を除外し、その処理手順は線グラフ表示データを処理して線グラフ表示を表示画像で提示するに適した処理済みデータを与える請求項２１のシステム。

【発明の詳細な説明】
【背景技術】
【０００１】
【技術分野】
【０００２】
本発明は、ハードコピーに含まれる患者の臨床信号を、コンピュータにより分析して蓄積した後、コンピュータネットワークを介して遠隔場所へグラフィック形式で表示できる電子フォーマットへ変換するシステムに係り、さらに詳細には、ハードコピー上の脳電図（ＥＥＧ）及び心電図（ＥＣＧ）のチャートを、ウェブサーバーが受信し、分析し、蓄積した後、リクエストによりウェブブラウザーへ送信できる電子フォーマットへ変換するソフトウェアアプリケーションに係る。
【従来技術の説明】
【０００３】
一次医療に携わる医師の典型的な診療所（または病院の一部の急性医療ユニット）では、患者に対してＥＣＧテストが施される。これらのテストは通常、ペンが紙片上に時々刻々変化するＥＣＧ信号を記録する標準の３または１２リード方式で行われる。これらの装置は、「ペーパーストリップレコーダー」を呼ぶこともある。
【０００４】
紙片は通常、患者の永久記録に添付される（患者のフォルダーにハードコピーの形で入れられる）。この方法は実際、低コストで簡便なものであるが、一次医療の医師から紹介を受けた医師または他のヘルスケア提供者への記録の受け渡しはハードコピーの形でのみ可能である。例えば、患者を紹介された専門医がＥＣＧのコピーを必要とする場合、前もっ

てそのハードコピーを送ってもらうか、または患者がそのハードコピーの複写をもって紹介された医師の元へ行く必要がある。あるいは、一次医療の医師がＥＣＧの記録を受け手となる医師（または、病院もしくは医療センターなど）へファクシミリにより送信することが可能である。これによりハードコピーの持ち運びが不要になる。しかしながら、受け手の所で行われるそのコピーからの情報の読み取りは依然としてハードコピーフォーマットであるに過ぎない。

【０００５】

ＥＣＧ波形の分析は比較的簡単な仕事であるが、波形を分析し、その波形を遠隔場所で見たり、後で見たり、あるいはオフラインで、また遠隔場所でオンラインで分析するために電子保存手段に蓄積する価値のあることが多い。ハードコピーのＥＣＧ記録は自動的に分析できないばかりか、ビットマップまたはＪＰＥＧのような他の簡単な画像ファイル以外の電子的読み取り可能なフォーマットにも容易に変換することができない。これらの電子フォーマットは、容易に転送できるが、遠隔場所からリアルタイムでオンラインの数学的手法により分析することは不可能である。

【０００６】

従って、ＡＳＣＩＩフォーマットのＥＣＧデータをウェブベースで発生させた後、ＥＣＧペーパーストリップを標準のＣ++ビジュアルベーシックのような従来型開発プログラミング言語を用いて開発される電子的手段により自動的に変換且つ分析して、既存のソフトウェアシステムに容易に一体化できるようにするシステムが求められている。従来技術または関連技術を検索すると、画像の走査及び作図の領域においていくつかの既存の機能がオープンマーケットで利用可能であることがわかる。詳述すると、以下の機能に関連する既存の技術が存在する。例えば、ハードコピー上のグラフの電子（または、ＡＳＣＩＩ）データへの変換については、Silk Scientific Software社(www.SilkScientific.com)のUn-Scan-Itは、スキャナーでハードコピーの(x, y)グラフを読み取り、曲線の成分座標を抽出できる代表的な製品である。このSilk Scientific Software社は、そのウェブページ上において、ユーザーはハードコピー上のグラフをスキャナーの最大分解能で(x, y)ＡＳＣＩＩデータへ自動変換できると宣伝している。Un-Scan-Itは、任意のフルページスキャナー、ハンドスキャナーまたは他の画像入力装置と併用すると、ストリップチャート、装置出力、古いグラフ、出版物上のグラフなどをデジタル化できる。多くのデジタル化機能に加えて、このUn-Scan-Itはピーク領域を積分し、データを平滑化し、導関数をとり、グラフのスケールを変更し、他のソフトウェアプログラムに使用するために(x, y)ＡＳＣＩＩデータをエクスポートできる。別の例として、C++のビットマップグラフィックスプログラミングにMarv Luseにより記載されたビットマップ画像の読み取りに関する方法の既存の変種があるが、これはビットマップイメージのＡＳＣＩＩ(x, y)座標を抽出する。画像ファイルを開くこの方法は、この後者の機能を達成するためにLuseによる既存のソフトウェアの一部を改良したものである。また、画像を変化させる目的でJavaコードを用いてビットマップを読み取る方法が存在するが、これはJoseph L. Webber, in Special Edition Using Java 2 Platform Copyright 1999 by QUEに記載されている。しかしながら、従来技術のこれらのシステムにはいくつかの重大な問題がある。可搬性を有し、標準のワードプロセッシングツール（マイクロソフトエクセル及びワードのような）に直接統合可能であり、サーバーのプラットフォーム上で処理を行い最終的な曲線をクライアントに特許請求の範囲に記載されたように配布できるシステムが求められている。

【０００７】

ＡＳＣＩＩ(x, y)データの曲線をJavaScriptまたはVB Scriptのようなクライアントサイドの解釈コードを用いてクライアントプラットフォーム上に表示できることは、１９９６年に発表されたJavaScript PlotterにRonald H. Nicholsonにより記載されている。Nicholsonにより開発された機能は、ＡＳＣＩＩデータを読み取るように特に設計された（Nicholsonのコードに説明されているようにハードコード機能上で作動するのではない）曲線のサーバーサイドバージョンを発生させ、ウェブブラウザーウィンドウ内でＥＣＧ曲線の座標をプロットするために使用される。参考までに、この方法を「サーバーサイド・プロ

ッティング・ユーティリティー」またはＳＳＰＵと呼ぶ。ＡＳＣＩＩ座標データをクライアントに送信し、そこで、Javaアプレットがデータを読み取り、ウェブブラウザーウィンドウ内にプロットする別のユーティリティーを開発することもできる。参考までに、この方法を「アプレット・プロッティング・ユーティリティー」またはＡＰＵと呼ぶ。

【０００８】

上述した必要な利点は２つの方法で得ることができる。即ち、生のＡＳＣＩＩ(x, y)データをサーバー上のアレイに書き込み、ＳＳＰＵを用いるなどしてクライアントのＨＴＭＬページ内の曲線に変換するか、または生のＡＳＣＩＩ(x, y)データをクライアントへ直接書き込み、ＡＰＵを用いるなどしてプロッティングプロセスを実行するクライアントサイド機能によりそれらを解釈することができる。

【発明の概要】

【０００９】

本発明は、収集した患者データの線グラフ表示をデジタル信号処理に適したデジタルデータへ変換するシステムであって、収集した患者データの線グラフ表示を含む画像を表すデータを受け、受けたデータを分析して画像表示データ中の線グラフ表示を構成する個々のピクセルを同定し、基準点に関する個々のピクセルの相対的位置を特定する複数のピクセル表示データ要素を所望のデータフォーマットで発生させ、個々のピクセルの相対的位置を特定し、（ａ）個々のピクセルのカラー及び（ｂ）個々のピクセルの輝度振幅のうち少なくとも１つを特定することにより所望のデータフォーマットを発生させ、発生させたピクセル表示データ要素をファイルに蓄積するステップより成る方法に係る。さらに、受けたデータをフォーマットして、線グラフ表示を表すデータを処理して線グラフ表示を表示画像の形で提示するに適した処理済みデータに変える遠隔場所での処理手順に適合するフォーマットを発生させ、ユーザー命令に応答してフォーマット済みデータを遠隔場所に送り、フォーマット済みデータを用いて遠隔場所で表示画像を発生させる。

【実施例】

【００１０】

本発明は、好ましい実施例を示す添付図面及び以下の詳細な説明からさらに完全に理解されるであろう。しかしながら、添付図面は、本発明を特定の実施例に限定するものとして捉えるべきでなく、説明のため、また理解を助けるためのものにすぎない。

【００１１】

本発明は、ＥＣＧまたはＥＥＧ画像を紙上に再現したようなハードコピー上の患者の臨床信号をパーソナルコンピュータ上の標準のツールを用いて信号処理及び分析ができるようにするためにＡＳＣＩＩデータに変換する統合型ソフトウェアシステムに関わる。本発明はハードコピー上のＥＣＧまたはＥＥＧ画像を、信号及びデータ処理機能を用いて分析可能なソフトコピー上の画像に変換する。この画像は、蓄積した後、ウェブベースのサーバー処理方法を用いて表示することができる。

【００１２】

以下に述べる本発明の好ましい実施例は、ハードコピー上のＥＣＧ画像を、アクティブサーバーページ（ＡＳＰ）アプリケーションを用いるウェブブラウザーにその生のＡＳＣＩＩ座標により表示することにより処理するものである。この実施例は、曲線の形のＡＳＣＩＩ座標をクライアントに送る２つの異なる方法を記載する。その第１のものはＳＳＰＵであり、第２のものはＡＰＵである。これらのＡＳＰ法は、より一般的な製品をサポートするために一般化可能である。この実施例は、マイクロソフトビVisual StudioまたはJDK 1.3.1を用いてC++、Java及びVisual Basicで開発されたものである。これにより、本発明のシステムは他のアプリケーションの一部として容易に一体化することが可能である。しかしながら、当業者は、この実施例が説明の目的のためのものにすぎないことがわかるであろう。

【００１３】

図１－１２は、ＥＣＧ画像情報を読み取り、処理しそしてクライアントのスクリーンに表示することができるＳＳＰＵ及びＡＰＵ成分の能力を示す。図１３－１８は、特定のピク

セルカラー値をビットマップファイルから読み取り、抽出し、そして新しいビットマップファイルを形成する本発明のReadBMP成分の能力を示す。図１９－２６は、ＡＳＣＩＩデータを読み取り、単純なプロットを形成する本発明のＡＰＵ成分の能力をユーティリティーに含まれる自動曲線近似機能と共に示す。

【００１４】

この点からのデータの処理は、図１のデータ流れ図に従って進行する。図１に示すように、スキャニング装置（ファクシミリ装置のような）へのＥＣＧ（ハードコピーフォーマット）の手動による入力を矩形のブロック１０１に示す。ファクシミリでスキャンするとビットマップファイル（.bmp）が形成されるが、他の受け入れ可能なフォーマットとして.gifまたは.jpg(.jpeg)が含まれる。かかるファイルの形成は当該技術分野で周知であるため、ここでは詳説しない。ビットマップファイルは、一旦形成されると、画像リーダー１０１に組み込まれた本発明のBitmapReaderプログラムにより検索される。BitmapReaderの目的は、生の２進 .bmpファイルを開いて全てのエクセル値の位置を求めることである。BitmapReaderは、この実施例では２つの別個のユーティリティーより成り、１つは、ReadImage.classというJavaユーティリティーであり、もう１つはRead.BMP.vbというVisual Basicユーティリティーである。ビットマップは検索され、好ましくは、以下のフォーマット(x, y, color)iを有するＡＳＣＩＩに変換される。ここで、(x)はi番目のピクセルのx座標位置を表し、(y)はi番目のピクセルのy座標位置を表し、colorはi番目のピクセルのＲＧＢカラーを表す。通常、ビットマップファイル及び他のグラフィックファイルのフォーマットでは、１番目のピクセルの座標は１(x=0, y=o)または(0, 0)に相当する。これは、.bmpファイルの一番上の一番左側の隅の位置を表す。最後のピクセルの座標は、位置($x=x_{max}$, $y=y_{max}$)または(x_{max}, y_{max})に相当する。これは、.bmpファイルの一番下で一番右側の隅を表す。

【００１５】

この点までのプロセスを説明するために、図２に示すビットマップファイルのサンプルについて考える。図２のサンプル波形は、元の紙の画像をスキャニングすることにより形成される.bmpファイルの形になっている。本発明の画像リーダーは、例えば、ビットマップファイルからピクセルを従来の方式で読み取ることにより、多数の方法で実現可能である。ヘッダー情報を読み取る構造を決定するプロセスは、Marv Luse, Bitmapped Graphics Programming In C++, Addison-Wesley 1993 and James D. Murray and William van Ryper, Encyclopedia of Graphics File Formats, 2nd Edition, O'Reilly & Associates, Inc., 1996のような幾つかのソースから得られるためここには説明しない。提示されるコードセグメントは、任意のビットマップに読み込む第１の基準の方法を使用する。このコードは、ＡＳＣＩＩ成分(x, y, color)をいかにして識別するかを明示する。

【００１６】

あるいは、Javaアプレットユーティリティープログラムのコードセグメントは画像ファイル（例えば、.jpgファイル）から個々のピクセルを得る。このコンピュータプログラムは、Joseph L. Webber, Special Edition Using Java 2 Platform; Copyright 1999 by QUEに記載されたアプレットに基づくものでよい。しかしながら、ピクセル値の抽出及び出力ファイルへの書き込み方法は、本発明に従って変形されるであろう。

【００１７】

データオブジェクト１０２のアプレットがファイルへの書き込みを行うには、書き込みアクセスを可能にするために使用中のコンピュータシステム上にポリシーファイルをセットする必要がある。図３は、このアプレットに関連するポリシーファイルのサンプルを図３に示す。図３は、アプレットによるファイルへの書き込みを可能にするセキュリティーポリシーファイルのサンプルを示すコンピュータスクリーン画像である。このファイルは、アプレットがビットマップ画像の(x, y)座標をＡＳＣＩＩテキストファイルに書き込むために使用するものであり、その後、このファイルは後処理ツールにより読み取られ、マイクロソフトエクセルに保存される。この画像リーダーのＡＳＰバージョンはサーバー上で実行されるためにセキュリティーポリシーファイルを必要としないことに注意されたい。

この方法のアプレットバージョンは、ＡＳＣＩＩ(x, y)データポイントをビットマップ画像から読み取る能力を示すために説明した。そして、図４は、JDK appletviewerコマンドツールによるアプレットの実行及びoutput.txtという名前の出力ファイルへのアプレットの書き込みを示すコマンドウィンドウのサンプルを示す。

【００１８】

アプレットユーティリティーを実行すると、appletviewerはアプレットに特定されるように元の画像を生成またはエコーする。この画像の生成は必要条件ではなく、ただ、データがアプレットにより正しく読み取られたことを検証する働きがあるに過ぎない。これを図５に示す。

【００１９】

ＡＳＣＩＩテキストデータがここでは、ファイル(output.txt)に存在する。このファイルのコンテンツは、上述した方法（即ち、C++またはJava）若しくは他の任意の方法の何れかを用いてデータオブジェクト１０２により書き込むことができる。データは、一旦テキストの形で利用可能な状態となると、データベースまたはスプレッドシートの何れかに直接インポートすることができる。図６は、エクセルスプレッドシートに存在するこれらのデータの一例を示す。これらのデータは、Javaアプレットを用いて発生されたものである。

【００２０】

データのコンテンツを説明するために、簡単な曲線を、ピクセル値を用いて信号プロセッサ１０３及び画像オブジェクト１０４により発生させたが、この曲線はこれらのデータが実際に元のＥＣＧ画像を表すことを示す。これを図７に示す。図７は、マイクロソフトエクセル内のユーティリティーにより捕捉された生のデータを用いて本発明により描かれたピクセル値の曲線を示すコンピュータスクリーン画像である。これらは図６に示したものと同じデータポイントである。そのデータを読み取ると、y成分の要素が逆になるが、その理由は(0, 0)要素が画像ファイルのスクリーンの一番上の一番左側を表し、点（(xMax, yMax)により画定される画像ファイルの最大領域が一番下の右隅であるため、画像が垂直に裏返して表れるからである。これはＡＳＰ後処理装置により矯正される。

【００２１】

ＥＣＧ波形の再構成を示すために、ピクセル値のサンプルをプロットする。エクセルのスプレッドシートへのデータの直接インポートは、ＣＯＭインターフェイスを介するエクセルへの接続を確立するか、またはＡＤＯインターフェイスを用いてＡＳＰにより達成可能な任意のプログラムの助けにより実行可能である。マイクロソフトのVisual C++プログラムはこの目標にいかにして到達するかを示す一例である。このプログラムを走らせた結果得られるのがエクセルスプレッドファイルであるが、これは２つのコラムのデータ、即ちＥＣＧのx及びyピクセル座標を含む。

【００２２】

エクセルにおいて、ＡＳＰがＯＤＢＣドライバーを介してデータを読み取ることができるように名前付きのレンジを画定する必要がある。データの名前付きレンジの一例を図８に示すが、これはエクセルスプレッドシート内のコラムＡ及びＢ（即ち、Ｘ及びＹピクセルレンジ）に相当する。図８は、本発明により処理されるデータを読み込む準備のためにデータレンジをセットし、名前を付けるコンピュータスクリーン画像を示す。この名前を付けられたレンジにより、アクティブサーバーページ（ＡＳＰ）はこの図に特定される名前付きレンジにより与えられるハンドルを用いてマイクロソフトエクセルファイルから直接読み取ることができる。この名前付きレンジは、オープンデータベースコネクタクティビティ（ＯＤＢＣ）マネジャーがマイクロソフトエクセルファイルのコンテンツをＡＳＰ内で用いるＡＤＯ(ActiveX Data Objects)にリンクしてマイクロソフトエクセルファイルのコンテンツを自動的に読み取れるようにするために用いる。この実施例の出力はコラムのラベルを挿入しなければならないが、これは前述したプログラムコードに示すようにマイクロソフトVisual C++プログラム出力に含まれることがある。

【００２３】

これを図９にわかりやすく示す。ここでは、信号プロセッサ１０３及びＡＳＰのような画像形成オブジェクト１０４のサーバーサイドスクリプトを用いて生のデータを処理することが可能である。図９は、本発明において必要なコラム名を示す更新されたマイクロソフトエクセルスプレッドシートを示すコンピュータスクリーン上の画像である。ＡＳＰ内のＡＤＯを用いてマイクロソフトエクセルのデータを読み取る際、エクセルスプレッドシート内のデータの第１行は関連のスプレッドシートのコラムのラベルとして自動的に解釈される。

【００２４】

ＡＳＰプログラムは、エクセルに組み込まれた数学的及び統計的機能を用いてスプレッドシートのデータを検索し、表示して、データを処理した後、処理結果をウェブブラウザーに書き込む。このようにして、処理結果をデスクトップまたはラップトップコンピュータ、携帯電話またはＰＤＡにこれらの装置に合致するフォーマットで配布することができる。

【００２５】

ＡＳＰプログラムは、ビットマップに関連するピクセル座標の表値を発生する。しかしながら、より興味のあるのは図１０に示す表示であり、これはＡＳＰプログラムにより発生可能である。このプログラムは、前の例のＯＤＢＣリーダーをVBScriptプロッターと組み合わせる。Ronald H. Nicholson, JavaScript Function Plotter, published in 1966に記載されたこれらの方法make_1d_array、make_2d_array及びplot_2dは、本発明を実施するために変形して使用することができる。このユーティリティーはＳＳＰＵにおいて実現するのが好ましい。

【００２６】

ＡＳＰコードのこの利用は説明の目的のためであるに過ぎないことを知るのが重要である。例えば、上述のコードは特定の画像サイズのハード値を含む。しかしながら、このコードの移植性及び一般性を増加させるには、任意サイズの画像を受け入れることができるようにする必要がある。このコードは、実施例の特定サイズの画像のみをサポートするように調整されている。これは、このコードは（現在）８００×８００のピクセル画像をサポートできるにすぎないことを意味するが、本発明はこれに限定されない。当然のこととして、コードそれ自体は、任意サイズのＥＣＧ記録をサポートし、ユーザーが望むデータを表示できるようにするため融通性を増加させる必要がある。図１０を示す目的は、ウェブインターフェイスを介してユーザーは生のデータポイントと画像それ自体の両方とにアクセスすることを示すためである。

【００２７】

ＡＰＵにおいて、クライアントサイドのプロット機能は、サーバーサイドのアプリケーション（ＡＳＰ）により送信されたＡＳＣＩＩデータ値を読み取るのが好ましい。クライアントのユーティリティーは図１１に示すプロットを発生させる。

【００２８】

本発明の別の実施例において、２４ビットのカラー画像ファイルは、分析のために読み取り、データ及び分析結果から発生することができる。本発明のこの局面は、例えば、エクセルスプレッドシート内に形成されるマクロとして実行してもよいが、当業者は本発明がそれに限定されないことがわかるであろう。

【００２９】

上述したマクロを用いて本発明のこの局面を実行するためには、ユーザーはエクセルを立ち上げる必要があるが、その理由はユーティリティーReadMBP.vbが出力データのその処理をエクセルスプレッドシートユーティリティーに一体化するからである。他の実施例において、このユーティリティーはＡＳＣＩＩピクセル値を読み取るその基本的な役割を実行するためにはエクセル内に統合する必要はない。

【００３０】

このユーティリティーを形成するプロセスを、その形成プロセスを明らかにするためにここに要約する。エクセルを一旦立ち上げると、ユーザーはファイルメニューからツールを

選択し、ツール→マクロ→マクロを選択する。ユーザーには図１２に示すようなウィンドウが提示されるが、そこでマクロの名前を打ち込むことができる。図示のマクロはRead.BMPと表示されている。

【００３１】

Visual Basicのマクロそれ自体は、以下に示すように、ステートメント”Open For Binary Access Read”を用いて２４ビットのカラービットマップを開く。”Sub ReadBMP() [Intermediate code removed] Open [filename] For Binary Access Read As #1”。ここで、[filename]はビットマップファイルの完全なパスと名前に相当する。この値がハードコードされている場合、ユーザーは新しいファイルが読み取られる度にそのプログラムを保存し変更する必要がある。従って、本発明の実施例では、ユーザーはこの情報をInput_Sheetと呼ぶワークシートに入力し、これがマクロにより読み取られる。その後、ファイル名が入力ワークシートから動的に読み取られる。このようにして、ユーザーはマクロそれ自体を変化させることなく所望の任意のファイル名を入力することができる。この入力ファイルワークシートを図１３に示す。

【００３２】

図１３に示すように、ファイルパス及びファイル名のセルは特定のビットマップファイルの位置及び名前を特定する。この場合、ファイルパスは”D:/Data/My Pictures/”、またファイル名は”rainbow.bmpであり、このファイルはビットマップファイルから種々のカラーの個々のピクセルにアクセスするためのRead.BMPユーティリティーの融通性及び能力を示すために使用される。従って、ユーティリティーそれ自体では、その一般的なファイル名を特定のワークブックからこれらのセルを読み取ることができる特定のコードで置き換えてもよい。ここでは、String変数であるifpath、ifname及びifilesは、ユーティリティーが読み取るべきビットマップのパス、ファイル名、連結ファイルパス及び名前をそれぞれ特定するにすぎない。同様に、２進出力ファイルへの書き込みのための出力パスは同様な態様で定義される。図示の例において、全てのＡＳＣＩＩデータはpixelCoordsという題のワークシートに書き込まれる。従って、全てのピクセルデータはこのスプレッドシートに保存される。

【００３３】

読み取りアクセスのためにビットマップファイルを一旦開くと、データの個々の要素を読み込む必要がある。ファイルは２進アクセスのために（ビットマップまたは他の画像ファイルを読み取る際に必要であるように）開かれるため、個々のデータ要素はバイトの形でファイルから読み取られる。これは、実施例のプログラムコードでは、変数xをバイトタイプである:Dim x As Byteと宣言することにより行われる。この変数xはファイルから全てのデータ要素を読み込むために使用することができる。そうするためにはまず第１にヘッダー情報をビットマップファイルから読み取る必要がある。ヘッダーはビットマップの高さ、幅及びカラースキーム（即ち、各カラー、赤、緑、青について２４ビットまたは８ビット）に関する情報を提供する。この情報をマクロが使用して、これらのデータを読み取る際ピクセルの各ラインの端部に到達した時を求める。

【００３４】

最初に、ビットマップの幅及び高さを求める必要がある。各バイトはGetコマンドを用いてビットマップファイルから読み取られる。Getコマンドはファイルユニット番号（#1）、レコード番号（オプション）及び実際のデータ（x）を読み取る。ビットマップファイルから読み取られる１９番目の要素はビットマップの高さを表す。ビットマップから読み取られるレコードは２つのカウント変数、即ち、amt及びrecNumberを用いて追跡される。変数amtは、ビットマップのどこにプログラムが存在するかを追跡するために用いるが、変数recNumberはピクセルデータを新しいビットマップファイルに書き込むために使用する。幅を読み取るが、これはその高さから３バイトだけ離れており、その後、ビットマップの深さを読み取る。これは、深さを２４ビット、即ち、赤が１つの８ビットセグメント、緑が１つの８ビットセグメント及び青が１つの８ビットセグメントとして定義する。従って、任意所与のピクセル値のカラーは｛ＲＧＢ｝で与えられ、Ｒは０乃至２５５の間で

変化し、Ｇは０乃至２５５の間で変化し、またＢは０乃至２５５の間で変化する。

【００３５】

次に、ビットマップ内の第１のピクセルの開始点を読み取る。そして、この点でこの態様によりビットマップ全体を読み取る。amt変数は、多くのデータ要素がいかにして読み取られるかを追跡するために１に再び初期化するのが好ましい。それはバイト毎に変数xに読み込まれる。ピクセルの行及び列位置は、ビットマップの高さに関連して読み込まれるピクセルの量に注目して追跡される。行及び列毎のピクセル値については、３つの値、即ちＲ値、Ｇ値及びＢ値がある(0..255, 0..255, 0..255)。ピクセルカラーの選択及び特定は、オペレーターが関連のカラーワード値を用いてビットマップファイル内に所望のカラーを宣言することを可能にするコードセグメントを組み込むことにより自動化されている。これを表１に示す。ユーティリティーは、カラーのワード値を関連の｛ＲＧＢ｝カラーバイト値にマッピングした後、関連のカラーバイト値を用いてビットマップファイル内のマッチするバイトの組み合わせを探索する。

【００３６】

表１：サポートされるカラー

	R	G	B
Red/red	255	0	0
Yellow/yellow	255	255	0
Blue/blue	0	0	255
Green/green	0	255	0
Brown/brown	128	64	0
Purple/purple	128	0	128
Pink/pink	255	0	255
Black/black	0	0	0
White/white	255	255	255
Gray/gray	128	128	128
Aqua/aqua	0	255	255

【００３７】

これよりも多数のカラーを選択できる能力を提供するために（そしてリストが事実上無制限であるようにするために）、三色のバイト値を特定し、適当な論理をVisual Basic Macroに付加する必要がある。当業者はこのプロセスを例えば、ファイルから値を読み込むかまたはユーティリティーをビジュアルカラー発生ユーティリティー（カラーパレットのような）と直接インターフェイスしてユーザーがアナログカラーを選択した後、関連のＲＧＢバイト値に変換するようにして自動化することも可能である。

【００３８】

ユーザーがビットマップから赤いピクセルの値を全て選択したい場合、マクロを｛２５５、０、０｝に等しいバイト値の組み合わせをサーチするように指示するだけでよい。これを行うコードの説明は以下の通りである。ファイルはその全体が読み取られる。バイト値は３つ一組で、即ち、各行及び列のピクセル値について３つのバイト、即ち赤、緑及び青にそれぞれ対応するバイト値で読み込まれている。ピクセルの３つのバイト値が読み取られる度に、それら３つのバイト値を３つの既知の値に対して評価することによりどのカラーか知るためにチェックする。これらの値は、ユーザーが所望のカラーをビットマップから抽出するために特定できるようにするフィールドを介してInput_Sheetにおいて選択される（図１３を参照）。図１３において、第３の列の第２の要素はユーザーが特定するカ

ラー値、即ち選択されたカラーのユーザータイプであり、これはこのシートの右側のチャートに示すカラーの１つに対応しなければならない。変数は、If…thenステートメントの内部に与えられるのが好ましい。

【００３９】

生の２進ファイルを開くプロセスについては上述した。ファイルパス及びファイル名は変数を用いて捕捉される。これらの変数は図１３に示すInput_Sheetにおいて初期化されている。新しいビットマップファイルへの書き込みは簡単である。即ち、入力ファイルから１バイトを読み取る度に出力ファイルへそれを書き込む。しかしながら、出力の場合は、そのファイルに書き込まれた特定のレコード番号を追跡する必要がある。

【００４０】

入力ファイルから各バイトが読み取られると、それらの値はバイト変数として保持され、追跡変数がインクリメントされる。これらのバイト値はまだ出力ファイルに書き込まれない。まず、バイト変数の入力組み合わせが既知のカラー値により定義されるユーザーが特定したカラーにマッチするか否かをチェックする必要がある。所望のカラーにマッチするカラーの組み合わせが見つかると、これらのカラーを正しい順序で出力ファイルに書き込む必要がある。これは、Ｂカラーを(x-2)位置のバイト要素として、Ｇカラーを(x-1)の位置のバイト要素として、またＲカラーをx位置のバイト要素として書き込むことにより開始するのが好ましい。これを一旦行うと、追跡変数はファイルポインターを読み取り出力ファイル内の次の位置に位置決めするための準備としてインクリメントされる。

【００４１】

しかしながら、所望のカラーにマッチするピクセルが見つからない場合、出力ファイルはその位置を空欄にしたままにはできず、他の全てのピクセルに関連するカラーのピクセル位置を維持する必要がある。これは、実施例では、所望のカラーに対応しないピクセルをそれらの位置に書き込むことにより行う（ＲＧＢの組み合わせが｛２５５、２５５、２５５｝に注目）。

【００４２】

ピクセルデータは、ピクセルデータの読み取り前にファイルの実際のサイズがファイルヘッダーから直接得られる２進読み取り法を用いて抽出される。これにより、画像を前もって知ることなしに任意サイズのファイルを読み取ることが可能となる。生の２進データは、各ピクセルのカラーが正確に知られている１ピクセルにつき３つの８ビットセグメント（２４ビットのカラービットマップに相当する）において抽出され、実際の信号をバックグラウンドノイズから判別するために使用することができる。ピクセルは一旦抽出されると、ユーザーが特定する所望のカラーマップと比較され、これによりユーザーが抽出するために所望されるものに対応するピクセルが選択的に抽出される。その後、この方法は抽出されたそれらの値のＡＳＣＩＩピクセル座標を書き出して、ユーザーが特定する画像のカラーの幾何学的コピーだけを含む新しいビットマップファイル（市販のプリントまたは画像操作プログラムに適合するフォーマットの）を新しく形成する。

【００４３】

抽出されたピクセル値は、元のビットマップから抽出されたユーザーが特定するカラーの二次元(x, y)位置を表す。これらの位置がＥＣＧ信号の記録を表すとしてそれらを分析することにより、生のデータから電子的に頻度（即ち、心拍数）を求めることができる。あるいは、ＥＣＧ記録を構成する個々のセグメントを（STセグメント長を求める時にまたはＥＣＧ信号内にアーチファクト（ノイズ）が存在するか否かを判定するために必要であるように）チェックしてもよい。本発明の別の有利な特徴は、データ抽出方法に追跡アルゴリズムを付加した点にある。追跡アルゴリズム（KalmanまたはBatch最小二乗フィルターのような）により、ＥＣＧ頻度の変化を予測し、頻度またはＥＣＧ信号の構成の変化が統計的に意義あるものである（従って臨床家またはヘルスケア提供者にとって重要な情報である）か否か、または同様な生理学的背景（体表面積、年齢、性別、病歴）と比較して普通であるか否かを確かめることができる。このようにピクセルデータを捕捉すると、信号データの保存及び検索のための重要な信号成分の圧縮が容易になる。例えば、信号それ自

体に関連する実際のピクセル（x, y座標値）を蓄積すると（信号を取り囲む白い未使用の空間を無視して）、実際の信号に必要な蓄積量が減少する。さらに、元の信号値の再形成が容易になるが、その理由は信号の(x, y)成分位置が蓄積され、実際の信号を再形成する残りの仕事は図１７及び１８（ａ－ｃ）の例に示すように信号の周りの白い空間を配置するだけであるからである。

【００４４】

迅速な実行を可能にするため、作図ツールを用いてシート上に形成されるAutoShapeはマクロReadBMPに割り当てられる。これにより、ユーザーは入力シートからそのユーティリティーを走らせることができて便利である。その形状へのマクロの割り当てはただ、オブジェクトを右クリックした後、メニューリストからマクロの割り当てを選択するだけで行える。図１４に示すようなメニューウィンドウが表れるが、ユーザーはマクロを選択してＯＫをクリックする。

【００４５】

動作について説明すると、ユーザーが最初に実行することはビットマップの位置を特定することである。この位置はInput_Sheet上の「File Path」セルに入力する。その後、ユーザーは「File Name」セルにビットマップの名前を書き込む。そして、ビットマップから抽出する所望のカラーピクセルを入力する。これがすんだら、ユーザーはマクロボタンをクリックするだけでよい。ビットマップのサイズに応じて実際の抽出動作は数秒から数分かかるが、この性能は装置のプロセッサ及びメモリにより異なる。これが完了すると（カーソルが砂時計から通常の形に変化する）、ユーザーは出力シートをクリックすればよい。上述したように、この例の全ての出力はワークシートのシート５に向けられるが、これは必要条件でないことが明らかである。

【００４６】

図１５に示すrainbow.bmpファイルは、マイクロソフトペイントを用いて作成されたものである。この例の入力スクリーンは図１３に実際示されている。赤のカラーが選択されている。図１６が対応の出力を示す。図１６は、この図のビットマップファイルからの処理済み出力のコンピュータスクリーン画像であり、本発明に用いる図１５に示したビットマップファイルから単一のカラーが抽出された態様を示す。赤色カラーに関連するピクセル座標は、正しい座標が実際に抽出されていることを検証する視覚補助手段として与えられるが、それらの座標のプロットの左側のコラムフォーマットに示す。図１６のコラムＡ及びＢは赤色のピクセル要素全ての行及び列ピクセル位置である。コラムＡ及びＢはチャートツール（（ＸＹスキャター）を選択）を用いてオプションとしてプロットされている。これらのデータのプロットを添付のグラフに示す（ワークシートの右側に示されている）。図１６からわかるように、ピクセル値はビットマップの真の表示である。しかしながら、正確なピクセル位置もわかる。

【００４７】

図１７は、発生したビットマップ、test.bmpを示す。図１７は、元の画像から抽出された選択カラーのビットマップ画像を再び形成するRead.BMPユーティリティーの１つの能力を示すものである。このRead.BMPユーティリティーは、特定のカラーを選択するためにユーザーが照会した後、ＡＳＣＩＩフォーマットの（図６の説明文に記載されている）それらの座標を与え、その後、元のビットマップファイルに表れるのと同じ位置にあるそれらの座標だけを用いて新しいビットマップを形成する。この図は、本発明においてtest.bmpと呼ぶ新しいファイルの内容を示すコンピュータスクリーン画像である。

【００４８】

ファイルは種々のカラーを選択して再び発生することも可能である。Rainbow.bmpから青、緑及びピンクを選択した図１８（ａ）－（ｃ）にこれを示す。

【００４９】

ＡＳＣＩＩデータは、スプレッドシート（またはデータベース）から取り出した後、そのデータをアクティブサーバーページを介してＨＴＭＬページの形でクライアントのコンピュータへ書き込むなどしてユーザーのいる遠隔場所に送信することができる。ＨＴＭＬペ

ージはJavaアプレットへのコールを含み、これはその後、ＡＳＣＩＩデータを読み取り、生のデータをプロットする。アプレットはまた、さらに別の任意の数の(x, y)データセットを同じ軸上にプロットする。以下に述べる例では、アプレットプロッターユーティリティー（ＡＰＵ）がサーバー側のVisual Baxic scriptに供給される生のデータの上に最小二乗法により発生された最良近似回帰曲線の座標をプロットするアプリケーションについて述べる。本発明のこの局面により、ＧＩＦまたはＪＰＥＧ画像のサーバー側でプロット及びインターネットを介する受け渡しを行うことが不要になる。さらに、アクティブサーバーページは、そのデータをプロットのためにクライアントへ書き込む前に回帰曲線の最良の形（線形または二次曲線）を決定する。

【００５０】

インターネットを介する送信は、プロットの画像（サーバー上で作成されクライアントに送信される）の送信よりＡＳＣＩＩ（テキスト）データの送信の点で優れた性能が得られる。処理はサーバーとクライアントの両方で行われるが、サーバー上ではデータがスプレッドシートから取り出され線形または二次曲線に近似され、アプレットパラメータコール内のＨＴＭＬページにフォーマットされるが、クライアント上ではデータが読み取られプロットされる。

【００５１】

これを実現する選択肢には幾つかある。これらには、ＣＧＩまたはサーバー側スクリプトを用いてプロットを形成した後、ＧＩＦまたはＪＰＧ画像をクライアント上に表示し、アプレット内の全てのデータをハードコーディングし、クライアントの処理によりプロットし、画像をスプレッドシートプログラムを用いて作成して、その画像をファイルにコピーした後、プロット画像をアクティブサーバーページまたはクライアント上のＨＴＭＬページを介して表示することを含む（アクティブサーバーページまたはＨＴＭＬページの何れかを介して画像を表示するために調整するオフライン処理が必要）。もちろん当業者は、他の実施例も可能であることがわかるであろう。

【００５２】

好ましい実施例において、ＸＹデータをプロッティングし、多項式の最良近似回帰法を用いて自動的に曲線近似をするためのアクティブサーバーページ及びアプレット駆動ユーティリティーを用いる。ＸＹデータは、エクセルスプレッドシートから抽出されたJavaアプレット及びアクティブサーバーページ（ＡＳＰ）を介してウェブブラウザーに表示することができる。プロットを行うための生のデータは、サーバー上に位置するエクセルスプレッドシートワークブックに維持しても良い。ユーザーはこのファイルを自由に変形した後、ＡＳＰから遠隔操作で実行されるアプレットを用いてそのデータをプロットする。プロットするためのデータはＡＳＰにより読み取られ、アプレットのパラメータタグフィールドに配置される。その後、ＡＳＰは線形及び二次式の最小二乗回帰機能を実行して曲線近似を行い、これが生のデータ上のオーバーレイとしてプロットされるパラメータタグを介してアプレットに供給される。その後、プロットが標準のウェブウィンドウに現れる。回帰機能は、線形または二次近似の何れか（生のデータポイントからＡＳＰにより最小二乗方式で決まる最良の選択）に基づき最良近似オーバーレイ曲線を形成する。

【００５３】

本発明のさらに別の実施例において、アクティブサーバーページ及びアプレット駆動ユーティリティーを用い、最小最良近似回帰法によりＸＹデータをプロットし、自動的に曲線近似を行うことができる。生のＡＳＣＩＩ形式のＸＹプロットデータは、従来の態様で遠隔のサーバー上に蓄積するのが好ましい。これらのデータはテキストファイル、データベースまたはスプレッドシートから取り出すことができる。この例の目的で、この実施例のために、これらのデータはスプレッドシートから得たものである。データは一旦蓄積された後、GenericPlotterアクティブサーバー機能により取り出され、そこで処理されて一般化された最小二乗最良近似機能が決定される。この最良近似機能により回帰方程式の係数が得られるが、これらは生のデータと新しく決定される回帰関数との間の近似の適合性と示すデータのオーバーレイを発生するために使用される。それと共に、生のプロットデー

タと回帰データの両方をアプレットにより使用されるパラメータタグへのコールとそれらのタグの両方を含むＨＴＭＬページに書き込まれるが、それはクライアントの装置に常駐している。これらのデータをＨＴＭＬページに書き込むと、クライアントの装置のＨＴＭＬページ内に固定状態となって常駐する。クライアントの装置によるＨＴＭＬページの取り出しは、自動的にサーバードメインの名前に関連するＵＲＬウィンドウにアクティブサーバーページの名前を特定することにより行われる。

【００５４】

クライアントの装置は、ＨＴＭＬページ内に含まれるパラメータタグを読み取るように設計されたJavaアプレットをプラグインとして維持する。Javaアプレット機能は、元のＡＳＣＩＩデータ及びクライアント装置上に常駐するＨＴＭＬタグ内に含まれる最良近似回帰データを取り出し、これらのデータをアプレットクラスファイル内のアレイに読み込む。アプレットクラスファイルはこれら２つのデータセットを処理し、クライアント装置上に常駐するブラウザーウィンドウ内に生のデータ及び回帰データのオーバーレイの両方を含む二次元のプロットを形成する。そのプロット自体は、クライアントの装置上に常駐するが、クライアントだけにより発生されるため、画像ファイルをサーバーからクライアントへ受け渡すことが不要になる。

【００５５】

本発明のシステムは、幾つかの点で既存の自動プロットプロセスとは異なる。生のＡＳＣＩＩデータがサーバーからクライアントへ直接書き込まれる。サーバーからクライアントへの画像またはプロットは書き込まれない。クライアントは、アプレット内に含まれるプロット機能により実際のプロットの発生を完全に担当する。これは送信帯域幅が狭い場合に特に有利である。また、クライアント装置が連携する必要のあるサーバー装置を識別するＨＴＴＰトンネリングが不要であるため有利である。このようにしてクライアント装置は１つの装置に物理的に連結されない。その代わりアプレットコールを発生するアクティブサーバーページを含む任意のサーバーがデータをクライアント装置に十分に供給する。セキュリティーポリシーは有利なことに、データをサーバーから読み取る際変更または破壊されないが、その理由は、クライアント内のアプレットがＨＴＭＬページのアプレットコール内に含まれるパラメータタグから固定した値を直接読み取るからである。

【００５６】

動作について説明すると、実行はサーバーとクライアントの両方で行われる。プロットすべきデータを含む生のＡＳＣＩＩデータファイルが存在するかまたは外部プロセスもしくは個人により形成される。現在のこの実施例では、このファイルがエクセルにより読み取られ表に配置された後、特定の名前（この例ではrawData.xls）で保存される。本発明のこの局面と前のものとの関係は、Read.BMPユーティリティーにより作成される生のＡＳＣＩＩデータが既にＡＳＰによる直接アクセスに必要な形式でデータを提供する点である。一旦保存されると、アクティブサーバーページはこのファイルを読み取り、クライアント上に表示されるＨＴＭＬページ内でのコールに適したフォーマットでデータを配置する。ところで、クライアント上では、そのブラウザーへのＨＴＭＬの送信によりクライアント上でのアプレットの検索が開始され、クライアントブラウザーで実行される。アプレットはクライアント上のＨＴＭＬページに書き込まれたパラメータを検索し、最良近似方程式を支配するだけでなく生のデータ上にオーバーレイされる最良近似曲線を変換する線形最小二乗法を用いてブラウザーウィンドウ内にプロットを表示する。

【００５７】

ＡＳＰによるマイクロソフトエクセルの名前付きデータレンジを読み取るプロセスは、公開された文献に記載されているため、この実施例から正確なステップを省略する。しかしながら、この名前付きレンジによるアクティブサーバーページ（ＡＳＰ）は、その図に特定された名前付きレンジにより与えられるハンドルを用いてマイクロソフトエクセルファイルから直接読み取ることができる。名前付きレンジは、オープンデータベースコネクティビティ（ＯＤＢＣ）マネジャーが使用して、マイクロソフトエクセルファイルのコンテンツをＡＳＰ内で使用されるＡＤＯにリンクさせてマイクロソフトエクセルファイルのコ

ンテンツを自動的に読み取ることができる。以下のステップは、エクセルスプレッドシートを立ち上げるプロセスを説明する。これらのステップはスプレッドシートが開かれているという前提で開始される。以下のステップ１乃至３は、エクセルスプレッドシート内のセルの名前付きレンジの作成について詳述するものである。

【００５８】

１）(x, y)フォーマットでデータを入力する、即ち１つのセルにつき１つの値を入力する。ワークシートの第１行はそれぞれx及びyのデータコラムの名前でなければならない。

【００５９】

２）ウェブページがデータを取り出すセルのレンジの名前を作成する。セルの上でマウスをクリックし、ドラッグして、このレンジ内に含める。

【００６０】

３）選択挿入／名前／定義。選択した領域の名前をワークシートに入力する。選択した下記の領域の名前はdataFieldsであり、これはこの実施例内のＡＰＵの働きを説明するために用いるレンジの名前の一例である。

【００６１】

スプレッドシートを一旦作成し、セルの選択した範囲を画定すると、スプレッドシートの範囲をＡＤＯとしてアクセス可能にする必要がある。この処理手順により、アクティブサーバーページが開いてエクセルスプレッドシート内のデータを取り出すことが可能になる。これは、スタートボタンに行って、以下のように設定－制御ファイル－管理ツール－データソース（ＯＤＢＣ）へ進むことにより、「データベース管理」を選択して、開始されるデータソース（ＯＤＢＣ）のアイコンをダブルクリックすると、図４に示すウィンドウが現れる。システムＤＳＮのタブを必ずクリックする。

【００６２】

追加ボタンをクリックすると、システムデータソース領域に新しいデータソースが追加される。マイクロソフトエクセルドライバー(.xls)を選択し、終了をクリックする。そうするとデータソース名（この場合はrawData）が追加される。このようにすると、ワークブックが選択される。ワークブックの選択を押すとナビゲーションウィンドウが現れる。ユーザーはシステムの保存ドライブ上に位置する特定のエクセルファイルへナビゲートして、そのファイルを選択し、ＯＤＢＣ管理ツールを閉じなければならない。これが完了すると、特定のエクセルスプレッドシート、名前付きレンジ及びワークブックがＡＤＯに永久にリンクされ、ＡＳＰコードの本体内でアクセス可能となる。

【００６３】

アプレットは作図エンジンである。アクティブサーバーレンジにより読み取られる生のデータをＡＤＯインターフェイスを介してエクセルファイルからパラメータとして受け取り、それらの生のデータポイントをウェブブラウザーウィンドウに表示される図形に変換する。アプレットは、データを文字列アレイに読み込んだ後、それらを浮動ポイント値に変換する。データはその後、最大及び最小のx及びy座標限界値を求めるために評価される。ウェブブラウザーウィンドウには軸線が描かれ、データポイントが軸線のサイズにスケーリングされた(x, y)対に変換される。ペイント法は各(x, y)対を読み込んだデータポイントの現在セットと次のセットとの間のラインセグメントに変換するためにdrawLine機能を使用する。２つの図形、即ち生のデータと最良近似関数（アクティブサーバーページ内の最小二乗法を用いて求めた）が描かれる。これらは共にアプレット内のパラメータアレイを介して入力される。

【００６４】

アプレットの重要な特徴を示して読者にそれらの重要性を喚起するために、アプレットの好ましい実施例を以下に説明する。アプレットは、アクティブサーバーページを介して受け渡されるパラメータによりデータをアレイに読み込む。これらのアレイは最初はテキスト列として充填され、その後、浮動ポイント値に変換される。２セットの浮動ポイントアレイ、即ち生のデータのための１つのセットと最良近似モデルデータの別のセットが発生する。SloadData列とSloadModel列とを用いてデータをアプレットパラメータリストから

読み込む。これらの列はＸ及びＹデータの両方を指示する。その後、これらのデータを浮動ポイント値に変換してData及びModelアレイにそれぞれ蓄積する。その後、Ｘ及びＹ成分を分離してxData、yData、xModel及びyModelアレイに蓄積するこれらのアレイのサイズは１５００個の要素に任意に設定されるが、これは３０００個の生のデータ値と３０００個の最良近似値とを意味する。当業者はベクトルを用いることによるこれらのアレイをアプレットパラメータリストに含まれるデータの量により必要とされる寸法に動的に設定することが可能である。他のアプリケーションのためのこの方法の他の実施例では、ベクトルを使用した。この実施例は言語構築のもっとも効率のよい方式を表すものでないのを知ることは重要である。動作効率の改善を主眼とした開発努力により、これらの方法の性能が改善されメモリの使用量が減少することに疑いはない。しかしながら、動作が物理的なメモリ及びデータの特徴的なサイズにより制限される特殊な状況では、これらを固定された量として特定すると最も効率的な実施例となる場合が多いことを知るのは重要である。

【００６５】

データは以下の２つのコードライン、即ちSloadData=getParameter(“loadData”)及びloadModel=getParameter(“loadModel”)を介してusePageParams()法により読み取られる。その後、extractData()法を用いてテキストから数値への変換を行う。まず第１に、データアレイの長さを求める必要がある。次に、索引変数を用いてこのアレイをループすることにより列の端部に到達したか否かを判定しなければならない。コンマの例をアプレットパラメータリストにおいて区切り記号として用いる。１つが見つかると、プログラムはそれが列のデータ値の端部に到達したことを知る。そこで、Dataアレイにデータ要素の全てを充填し、コンマを取り除き、データをＸ及びＹ成分に分離する。上述したものと同じステップをモデルデータについて実行する。

【００６６】

データを読み取った後、スケーリングを施しグリッドに描く必要がある。グリッド領域はdrawGrid()法を用いて描くが、これはアプレットパラメータリストに設定された量により印を測定する。この方法はまた、軸の大きな印の次の数値を引き出す。現在、Ｘ及びＹ座標方向における最小及び最大値の選択は、生のデータ及びモデルデータ（それぞれ最小または最大の何れか）に基づくものである。生のデータ及びモデルデータの両方の値を局部的な最小及び最大値と比較し、小さいか大きいかに応じて局部的な最小及び最大の新しい値に割り当てると、それぞれ帯域的な最小及び最大値となる。

【００６７】

この例では、データのスケーリング及び作図を１つのステップで行う。まず第１に、Ｘ及びＹ成分方向のスケールファクターをブラウザー作図ウィンドウのサイズ及びＸ及びＹ成分方向における最大及び最小値のレンジに関して決定する。次に、これらのスケールファクターをデータポイントの現在の値とそれらの次の値との間のスプラインを描くFORループの内部の各データポイントに適用する。その後、同じ方法を上述した生のデータに適用したようにモデルデータのスケーリングに適用する。

【００６８】

この例のアクティブサーバーページがVBScriptに書き込まれる。しかしながら、当業者は、そのページをJavaScriptのような他の言語で書けることもわかるであろう。何れのスクリプト言語を用いてもよく、開発者の好みに応じて同一の結果を得ることができる。VB ScriptはASP（しばしばマイクロソフトビジュアルスタジオ開発の状況内で用いるネイティブ言語）を開発する者によってしばしば使用される。第１のステップは読み取りアクセスのためにＡＤＯオブジェクトを開くことである。名前付きレンジdataFields内に定義されたフィールドがエクセルスプレッドシートから選択される。図１９は、名前付きレンジに関連する(x, y)データポイントを含む入力ファイルのサンプルを示す。このファイルを用いて図２０及び２１において作図ユーティリティーを後で説明する。

【００６９】

その後、読み取りのためにＡＤＯ接続を開く。データ要素アレイに読み込む。そして、ＷＨＩＬＥループを用いてファイルを読み取るが、名前付きレンジの端部まで読み取る。ア

プレットそれ自体をコードのＢＯＤＹテキストセグメント内でコールする。読み取ったばかりのデータをアプレットの範囲内においてパラメータとして書き込む。

【００７０】

この例では、生のデータとモデルデータが共にアクティブサーバーページからのパラメータとして書き込まれる。従って、アプレットがクライアントにロードされ、アクティブサーバーページが実行される結果として書き込まれたＨＴＭＬページからそのデータを受ける。このように、ＪＰＥＧまたはＧＩＦデータがクライアントに書き込まれるサーバーベースのプロッターとは対照的にＡＳＣＩＩテキストだけがサーバーからクライアントへ受け渡される。これはクライアントーサーバー帯域幅についての懸念が存在する時に重要な利点であるが、その訳はこれら２つの間にテキストベースのデータだけが受け渡され、テキストベースのデータ量は比較的小さい（即ち、数百個未満のデータポイントを表すため）からである。

【００７１】

モデルデータは、そのデータの最小二乗線形または二次解を見つけるサブルーチン、cFitLine()を用いて形成される。線形または二次モデルの選択は、生の各データポイントと最良近似モデル方程式との間の距離を$y=mx+b$（線形）または$y=c1+c2x+c3x^2$に基づき比較する重み付け距離パラメータ（lineChiSquire、quadChiSquare）に基づき行われる。モデル（線形、二次）は線形モデルと二次モデルとから重み付け距離測度を求めることにより選択する。２つの距離測度のうち小さい方が採用される使用されるモデルとなる。回帰データは生のデータの上に重ね書きするものとして書き込むのが好ましく、オプションとしての分散及び標準偏差も回帰曲線が生のデータにマッチする度合いを示すx^2二乗近似パラメータと共に書き込むことが可能である。

【００７２】

アクティブサーバーページはx^2二乗最良近似（Mahalanobis距離）比較機能を用いて最小二乗回帰曲線の最良の形態を自動的に求め、ＨＴＭＬページ内のアプレットタグフィールドにスプレッドシートから読み込んだ生のデータと最小二乗回帰曲線の両方とに関連するデータとを充填する。これらの数値をＡＳＣＩＩ値のリストの形でＨＴＭＬページに書き込む。生のデータ及び回帰曲線の重ね書きに関連するＡＳＣＩＩ値のこのリストは、浮動ポイント数アレイとしてクライアント装置上のアプレットにより読み取られる。浮動ポイント数のアレイはアプレットにより自動的にスケーリングされ、クライアント装置のＨＴＭＬウィンドウ内にプロットされる。

【００７３】

ＡＰＵ法は、ＵＲＬパラメータリスト内に選択されるプロットタイプとプロットすべきデータポイントの量とを特定することによりクライアント上に形成されるプロットのタイプを選択できるようにする。ＡＰＵ法の１つの利点は、Read.BMP VB Scriptユーティリティー、C++またはJavaの等価物により直接抽出されるＥＣＧまたはＥＥＧを共にプロットできることである。図２０は、ＡＰＵの実行を説明するための、ユーティリティーへのＵＲＬコール内に含まれるパラメータに充填をおいて示すコンピュータスクリーン上の画像である。パラメータリストはプロットタイプとプロット内に含まれるデータポイントの数をリクエストする。プロットタイプがecgだとＥＣＧプロットであることを示す。プロットタイプがxyだと簡単なグラフを示す。図２０はプロットタイプとクライアントによりプロットされる特定のデータポイントの行とを選択するためにGenericPlotter.aspページをいかに用いるかを示す。図２０の例のＵＲＬアドレスバーは下記の通りである。

http://ml1jrz0c/GenericPlotter.asp?ptype=ecg&np=300

【００７４】

ＡＳＰページは２つのパラメータ、即ち、発生すべきプロットのタイプがＥＣＧ記録であるか否かを特定するために用いると、データポイント対の数（ここでは、np=300は(x, y)対の数に相当する）を受ける。ptype=ecgを選択するとこの実施例では曲線近似機能がオフになり、クライアント上の単一セットの生のデータの作図が可能になる。図１１はこの実施例を示す。

【００７５】
ＵＲＬにおいて以下のことを特定すると、標準の(x, y)プロットが発生され、曲線近似が可能になる。従って、図９のデータをプロットするには、以下のＵＲＬをリクエストする。
http://ml1jrz0c/GenericPlotter.asp?ptype=ex&np=9
【００７６】
ptype=xy及びnp=9とＡＳＰページに特定すると、標準のx, yプロットが発生され、最初の９個のデータポイントがプロットされる。従って、曲線の近似が行われクライアントに送られる。この動作の結果は図２１のスクリーンに表れる。図２１は、データの最良近似重ね書きをした生のデータプロットと最良近似式を示す。ｘｙプロットが選択されると、ＡＳＰにより曲線近似ルーチンが自動的に作動し、元のプロット上に重ね書きされるプロットデータの最小二乗近似が発生する。この例では、図１９の示すファイルから９個のデータポイントがプロットするために選択される。
【００７７】
本発明は従来技術と比べて有意な利点を有する。ウェブインターフェイスを介してデータのＡＳＣＩＩ形式にアクセスすると、ＡＳＰコードに書き込まれた方法かまたは例えば、信号分析（例えば、周波数分析、信号セグメント長（ＳＴセグメント）分析、信号の平均化など）を行う能力を増加させる外部の.dllファイルの何れかを介して分析ツールの重ね書きが可能になる。これはエクセルスプレッドシートツールそれ自体（即ち、フーリエ変換、例えばパワースペクトル分析、Lombピリオドグラム及びウェーブレット分析に含まれるマクロを用いることにより行うことができる。あるいは、これらの方法をサーバーアプリケーションの内部の生のデータのこの分析を行うことができるVisual BasicまたはJscript（またはその言語）で書くことができる。その結果はウェブにより作動される装置（パーソナルデジタルアシスタント、ラップトップ、携帯電話）に送信可能である。
【００７８】
本発明はピクセルのＡＳＣＩＩ値にアクセスするため、スクリーン情報でビットマップ画像全体をペーストする必要はない。サーバーからウェブのクライアントへ生の.jpgまたは.bmp画像を送らなくても部分的な画像をスクリーン（または画像の一部）上に与えることが可能である。本発明はサーバーとクライアントとの間でＡＳＣＩＩデータの受け渡しを可能にするため相互性能が増加する。また、これらの方法はVisual Basicで書かれるため、既に存在する製品内部においてバックエンド法としてあげることができる。
【００７９】
回帰曲線の数学的処理及び生のデータの抽出はサーバー装置（クライアントでない）上で起こるため、アプレットが局部的な装置にアクセスできないことに関連するセキュリティ制限（即ち、アプレットに対する特別なセキュリティ制限を非作動にすることなく）満足する。本発明により、ウェブブラウザーまたはJavaプラグインサポートする任意の装置上にデータをプロットすることが可能である。サーバー上の処理及びＨＴＭＬを介するクライアントへのＡＳＣＩＩデータの送信により、サーバーとクライアントとの間の低帯域幅に対する解決法が得られるが、その理由はクライアントのプロセッサを用いてクライアント上にプロットが発生されるからである。
【００８０】
本発明の特定の実施例に関して説明したが、特許請求の範囲に記載される本発明の思想及び範囲から逸脱することなく多数の変形例及び設計変更が可能であることがわかるであろう。例えば、本願に記載した実施例は単一のアプリケーションサーバーを組み込んでいるが、当業者は任意の数のコンピュータシステムアプリケーションサーバー、データサーバー及びウェブサーバーにより本発明を実現できることがわかるであろう。同様に、別個の画像リーダーに加えて、別個の画像リーダー、データオブジェクト、信号プロセッサ及び画像形成オブジェクトに加えて、本発明のソフトウェアは個々の成分またはアプリケーションの組を備えた単一のアプリケーションを構成することが可能であり、その形態は特に限定されない。

【図面の簡単な説明】
【００８１】
【図１】本発明の好ましい実施例の動作を示すフローチャートである。
【図２】クライアントコンピュータ上の遠隔場所で表示されるＥＣＧビットマップ画像のサンプルを示すコンピュータスクリーン画像である。
【図３】アプレットがファイルに書き込めるようにするためのセキュリティーポリシーファイルのコンテンツのサンプルを示すコンピュータスクリーン画像である。
【図４】本発明のアプレットReadImageのappletviewerによる実行を示すコンピュータスクリーン画像である。
【図５】本発明のＨＴＭＬファイルの内部にappletviewerによりReadImageを生成させるコンピュータスクリーン画像である。
【図６】本発明においてテキストに書き込まれマイクロソフトエクセルにインポートされたＥＣＧ画像を含むファイルからのデータを示すコンピュータスクリーン画像である。
【図７】マイクロソフトエクセル内のユーティリティーRead.BMPにより捕捉された生のデータを用いて描かれる本発明のピクセル値のプロットを示すコンピュータスクリーン画像である。
【図８】本発明により処理されるデータを読み込むための準備としてデータ範囲を設定し名前を定義するコンピュータスクリーン画像である。
【図９】本発明における必要なコラム名を示す更新されたマイクロソフトエクセルのスプレッドシートを示すコンピュータスクリーン画像である。
【図１０】ＳＳＰＵを用いた本発明の元のビットマップファイルからの一部のデータのウェブブラウザー曲線を示すコンピュータスクリーン画像である。
【図１１】ＡＰＵを用いた本発明の元のビットマップファイルからの一部のデータのウェブブラウザー曲線を示すコンピュータスクリーン画像である。
【図１２】ビットマップまたはＪＰＥＧファイルからＡＳＣＩＩ(x, y)データを抽出する本発明の新しいマクロを発生させるコンピュータスクリーン画像である。
【図１３】ビットマップファイルからピクセルデータを抽出するために本発明においてユーティリティーRead.BMPにより使用されるInput_Sheetデータ入力ウィンドウを示すコンピュータスクリーン画像である。
【図１４】本発明に使用するマクロ割り当てウィンドウを示すコンピュータスクリーン画像である。
【図１５】元の画像から特定のカラースキームを抽出するユーティリティーRead.BMPの能力を示すよう本発明において使用されるビットマップrainbow..bmpを示すコンピュータスクリーン画像である。
【図１６】図１５のビットマップファイルからの処理済出力を示すコンピュータスクリーン画像である。
【図１７】本発明においてtestmap..bmpと呼ぶ画像ファイルの内容を示すコンピュータスクリーン画像である。
【図１８（ａ）】本発明に用いるrainbow..bmpから作成される青色を示すコンピュータスクリーン画像である。
【図１８（ｂ）】本発明に用いるrainbow..bmpから作成される緑色を示すコンピュータスクリーン画像である。
【図１８（ｃ）】本発明に用いるrainbow..bmpから作成されるピンク色を示すコンピュータスクリーン画像である。
【図１９】本発明において非ＥＣＧ特定データが入力され、セル領域が強調されたエクセルスプレッドシートウィンドウを示すコンピュータスクリーン画像である。
【図２０】ＡＰＵの実行を、ユーティリティーへのＵＲＬコールに含まれるパラメータを強調して示すコンピュータスクリーン画像である。
【図２１】本発明に用いるアプレットベースのＡＰＵのx-y作図出力を示すコンピュータスクリーン画像である。

【国際公開パンフレット】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 March 2003 (13.03.2003)

PCT

(10) International Publication Number
WO 03/021517 A2

(51) International Patent Classification[7]: G06F 19/00

(21) International Application Number: PCT/US02/20651

(22) International Filing Date: 1 July 2002 (01.07.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/316,602 31 August 2001 (31.08.2001) US
10/107,229 27 March 2002 (27.03.2002) US

(71) Applicant: SIEMENS MEDICAL SOLUTIONS HEALTH SERVICES CORPORATION [US/US]; 51 Valley Stream Parkway, Malvern, PA 19355 (US).

(72) Inventor: ZALESKI, John, R.; 219 Elmwood Lane, West Brandywine, PA 19320 (US).

(74) Agents: BURKE, Alexander, J. et al.; Siemens Corporation - Intellectual Property Dept., 186 Wood Ave. South, Iselin, NJ 08830 (US).

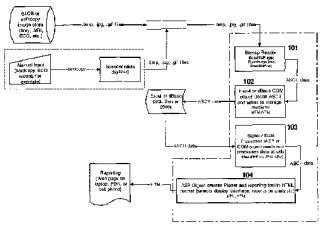
(81) Designated States (national): CA, JP.

(84) Designated States (regional): European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR).

Published:
without international search report and to be republished upon receipt of that report

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: A SYSTEM FOR PROCESSING IMAGE REPRESENTATIVE DATA



(57) Abstract: The present invention is directed to a system for converting a graphical trace of acquired patient data into digital data suitable for digital signal processing, which includes the steps of receiving data representative of an image including the graphical trace of acquired patient data; analyzing the received data to identify individual pixels comprising the graphical trace in the image representative data; generating a plurality of pixel representative data elements in a desired data format identifying the relative location of the individual pixels in relation to a reference point; generating the desired data format by identifying the relative location of the individual pixels and identifying at least one of, (a) color of said individual pixels and (b) luminance amplitude of said individual pixels; and storing the generated pixel representative data elements in a file. In addition, the received data may be formatted into a format compatible with a procedure located at a remote destination, wherein the procedure is for processing the data representative of the graphical trace to provide processed data suitable for use in presenting the graphical trace in a displayed image; and communicating the formatted data to the remote destination in response to a user command; and generating the displayed image at the remote destination using the formatted data.

WO 03/021517 A2

A System for Processing Image Representative Data

BACKGROUND OF THE INVENTION

The present application is a non-provisional application of provisional application having serial number 60/316,602 filed by John R. Zaleski on August 31, 2001 and U. S. Provisional Application Serial No. 60/348,602, entitled "System for Remote Plotting of a Graphical Trace," filed January 15, 2002.

Field of the invention

The present invention is directed to a system for translating hardcopy clinical patient signals into an electronic format that can be analyzed by a computer, stored, and later displayed in graphical form remotely over a computer network. More particularly, the present invention is directed to a software application that enables hardcopy Electroencephalogram (EEG) and Electrocardiogram (ECG) charts to be converted into an electronic format that can be received, analyzed, and stored by a Web server, and then transmitted to a Web browser upon request.

Description of the Prior Art

In a typical primary care physician's office (or even in some acute care units at hospitals) ECG tests are performed on patients. These tests are typically performed using the standard 3- or 12-lead methods in which a pen records the time-varying ECG signal on a paper strip. These devices are sometimes called "paper strip recorders."

The paper strip is normally attached to the patient's permanent record (included in hardcopy format in the patient's folder). While this method is indeed cost effective and simple, it is transportable only in hardcopy format from the primary care physician to any referred physician or other health care provider. For instance, if a patient is referred to a specialist who will require a copy of the ECG, the hardcopy will either need to be sent beforehand or the patient will need to carry a copy of the hardcopy record with him or her to the referred physician's location. Alternately, the primary care physician can send an electronic facsimile of the ECG record to the recipient physician (or hospital,

or surgical center, etc.). This obviates the need to carry hardcopy. However, the copy is still retrieved on the receiving end in hardcopy format only.

While analyzing an ECG waveform is a relatively simple task, it is often worthwhile to perform analysis on the waveform or to store the waveform in an electronic repository for remote viewing, later viewing, or off-line and remote on-line analysis. Hardcopy ECG records cannot be analyzed automatically, nor can they be transformed easily into other electronically readable formats other than simple images files, such as bit maps or in JPEG (***J**oint **P**hotographic **E**xperts **G**roup*) format. These electronic formats, while easily transferable cannot be analyzed using on-line mathematical methods, in real time from a remote location.

Accordingly, a system is needed that is capable of producing Web-based rendering of ASCII (***A**merican **S**tandard **C**ode for **I**nformation Interchange)* formatted ECG data, and then transforming and analyzing ECG paper strips automatically via electronic means, which is developed using conventional development programming languages, such as standard C++ and Visual Basic, so that it can be easily integrated into existing software systems. Research into prior or related art reveals that some existing functionality in the area of image scanning and plotting is available on the open market. Specifically, existing art relating to the following functionality is available. For example, relative to transforming hardcopy graphs into electronic (or ASCII) data, *Un-Scan-It* by Silk Scientific Software (www.SilkScientific.com) is a representative product that enables reading hardcopy (x, y) graphs from scanners and extracting the component coordinates of the plot curve. Silk Scientific advertises on their Web page that allows a user to automatically convert hard copy graphs to (x,y) ASCII data at Full Scanner Resolution. UN-SCAN-IT works with any full page scanner, hand scanner, or other image input device to digitize strip charts, instrumental output, old graphs, published graphs, etc. In addition to the many digitizing features, UN-SCAN-IT also integrates peak areas, smooths data, takes derivatives, re-scales graphs, and exports (x,y) ASCII data for use in other software programs. Another example is an existing modification to a method described by Marv Luse in Bitmapped

Graphics Programming in C++ relative to reading bitmap images) extracts the ASCII (x,y) coordinates of a bitmap image. The method for opening the image file was adapted from an existing piece of software by Luse in order to accomplish this latter function. A method also exists for reading bitmap images using Java code for the purpose of altering the images and is described by Joseph L. Webber in *Special Edition Using Java 2 Platform* Copyright 1999 by QUE. However, these systems of the prior art have several significant disadvantages. A system is needed that is portable, and integrates directly with standard word processing tools (such as Microsoft Excel and Word); and that performs processing on a server platform and delivers the finished plot to a client, in accord with the details specified in the Claims.

The ability to display plots of ASCII (x, y) data on client platforms using client-side interpreted code, such as JavaScript or VB Script, is described by Ronald H. Nicholson in a JavaScript Plotter published in 1996. The functions developed by Nicholson may be used to create a server-side version of the plot specifically designed to read ASCII data (as opposed to operating on hard-coded functions, as is described in Nicholson's code) and to plot the coordinates of an ECG trace within a Web browser window. For reference purposes, this method will be termed the ***Server-Side Plotting Utility***, or ***SSPU***. An additional utility may also be developed that transmits ASCII coordinate data to a client, where a Java applet then reads the data and plots them within the Web browser window. For reference purposes, this method will be termed the ***applet Plotting Utility***, or ***APU***.

The needed advantages described above can then be accomplished in two ways: raw ASCII (x, y) data can be written to an array on the Server and are transformed into a plot within the client's HTML (***Hyper Text Markup Language***) page, such as by using an SSPU; or, raw ASCII (x, y) data can be written directly to the client where they are interpreted by a client-side function that performs the process of plotting, such as by using an APU

SUMMARY OF THE INVENTION

The present invention is directed to a system for converting a graphical trace of acquired patient data into digital data suitable for digital signal

processing, which includes the steps of receiving data representative of an image including the graphical trace of acquired patient data; analyzing the received data to identify individual pixels comprising the graphical trace in the image representative data; generating a plurality of pixel representative data elements in a desired data format identifying the relative location of the individual pixels in relation to a reference point; generating the desired data format by identifying the relative location of the individual pixels and identifying at least one of, (a) color of said individual pixels and (b) luminance amplitude of said individual pixels; and storing the generated pixel representative data elements in a file. In addition, the received data may be formatted into a format compatible with a procedure located at a remote destination, wherein the procedure is for processing the data representative of the graphical trace to provide processed data suitable for use in presenting the graphical trace in a displayed image; and communicating the formatted data to the remote destination in response to a user command; and generating the displayed image at the remote destination using the formatted data.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a flow chart of the operation of the preferred embodiment of the present invention.

Figure 2 is a computer screen shot of a sample ECG bitmap image displayed remotely on a client computer.

Figure 3 is a computer screen shot of a sample security policy file contents to enable an applet to write to a file.

Figure 4 is a computer screen shot of an *appletviewer* execution of the *ReadImage* applet of the present invention.

Figure 5 is a computer screen shot of an *appletviewer* rendering of *ReadImage* inside of an HTML file of the present invention.

Figure 6 is a computer screen shot of data from a file containing an image rendering of an ECG written to text and imported into MS Excel in the present invention.

Figure 7 is a computer screen shot of a plot of pixel values in the present invention drawn using the raw data captured by the *readBMP* utility within MS Excel.

Figure 8 is a computer screen shot of setting the data range and defining a name in preparation for reading in the data to be processed by the present invention.

Figure 9 is a computer screen shot of an updated MS Excel spreadsheet illustrating required column names in the present invention.

Figure 10 is a computer screen shot of a Web browser plot of a subset of data from the original bitmap file in the present invention using the SSPU.

Figure 11 is a computer screen shot of a Web Browser plot of a subset of data from the original bit map file in the present invention using the APU.

Figure 12 is a computer screen shot of creating a new macro in the present invention that extracts the ASCII (x, y) data from the bitmap or JPEG file.

Figure 13 is a computer screen shot of an "Input_Sheet" data input window used in the present invention by the *ReadBMP* utility to extract the pixel data from the bit map file.

Figure 14 is a computer screen shot of a macro assignment window used in the present invention.

Figure 15 is a computer screen shot of a bitmap, rainbow.bmp, used in the present invention to illustrate the ability of the *ReadBMP* utility to extract specific color schemes from the original image.

Figure 16 is a computer screen shot of a processed output from the bitmap file of Figure 15.

Figure 17 is a computer screen shot of the content of an image file, called test.bmp in the present invention.

Figures 18(a)-(c) are computer screen shots of blue, green, and pink color bitmaps created from rainbow.bmp used in the present invention.

Figure 19 is a computer screen shot of an Excel spreadsheet window with non-ECG-specific data entered and cell region highlighted used in the present invention.

Figure 20 is a computer screen shot illustrating the execution of the *APU*, with emphasis on the parameters contained within the URL call to the utility.

Figure 21 is a computer screen shot of the "x-y" plotting output of the applet-based APU used in the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention will be understood more fully from the detailed description given below and from the accompanying drawings of preferred embodiments of the invention; which, however, should not be taken to limit the invention to a specific embodiment but are for explanation and understanding only.

The present invention is directed to an integrated software system for translating hardcopy clinical patient signals, such as ECG or EEG paper image renderings, into ASCII data for the purpose of enabling signal processing analysis using standard PC-based tools. The invention translates the hardcopy ECG or EEG imagery into softcopy by which the softcopy image can be analyzed using signal and data processing functions. The image can be stored and then be displayed using Web-based server processing methods.

The preferred embodiment of the invention described below involves the processing of a hardcopy ECG image through display of that image via its raw ASCII coordinates in a Web browser using an Active Server Page (ASP) application. The embodiment describes two distinct methods for delivering the ASCII coordinates in the form of plots to the client. The first of these is the SSPU and the second is the APU. These ASP methods may be generalized to support a more general product. The embodiment was developed in C++, Java, and Visual Basic using MS Visual Studio and JDK1.3.1. This allows the system of the present invention to be easily integrated as part of other applications. However, one of ordinary skill in the art will appreciate that this embodiment is for purposes of illustration only.

Figures 1-12 illustrate the ability of the *SSPU* and *APU* components of the invention to read, process, and display ECG image information to a client screen. Figures 13-18 illustrate the capability of the *ReadBMP* component of

the invention to read, extract specific pixel color values from, and to create new bit map files. Figures 19-26 illustrate the capability of the *APU* component of the invention to read ASCII data and create simple plots together with an automatic curve fitting function included in the utility.

Processing the data from this point will proceed in accord with the data flow diagram provided in Figure 1. As shown in Figure 1, the manual input of the ECG (in hardcopy format) to a scanning device (such as a facsimile machine) is shown in the rectangular outline 101 in Figure 1. The product of the facsimile scan is a bitmap file (.bmp), although other acceptable formats could include .gif or .jpg (.jpeg). The creation of such a file is well known in the art and will not be elaborated upon further here. Once the bitmap file is created, it is retrieved by a *BitmapReader* program of the present invention, embodied in Image Reader 101. The purpose of BitmapReader is to open the raw binary .bmp file and to determine the location of all pixel values. The BitMap Reader is defined in the current embodiment in the form of two separate utilities: one Java utility entitled *ReadImage.class* and one Visual Basic utility entitle *ReadBMP.vb*. The bitmap is retrieved and translated into ASCII, preferably having the following format: $(x, y, color)_i$, where (x) represents the x-coordinate position of the i-th pixel; (y) represents the y-coordinate position of the i-th pixel, and color represents the RGB color of the i-th pixel. Normally in bitmap files and other graphic file formats, the coordinates of the 1st pixel correspond to the position (x=0, y=0), or (0,0). This represents the position in the top left-hand most corner of a .bmp file. The coordinates of the last pixel correspond to the position ($x=x_{max}$, $y=y_{max}$), or (x_{max}, y_{max}). This represents the position in the bottom right-hand most corner of a .bmp file.

To illustrate the process to this point, consider the sample bitmap file, shown in Figure 2. The sample waveform of Figure 2 is rendered in the form of a .bmp file, created by scanning the original paper image. The image reader of the present invention may be implemented in a number of ways, such as by reading the pixels from a bitmap file in a conventional manner. The process for setting up a structure for reading the header information is not included as it is available in several sources, such as Marv Luse, Bitmapped Graphics

Programming In C++, Addison-Wesley 1993 and James D. Murray and William van Ryper, Encyclopedia of Graphics File Formats, 2nd Edition; O'Reilly & Associates, Inc. 1996. The code segments presented employ the method of the first reference to read in an arbitrary bitmap. The code shows explicitly how the ASCII components (x,y, color) are identified.

Alternatively, a Java applet utility program code segment may render individual pixels from an image file (such as a .jpg file). This computer program may be based on an applet described in Joseph L. Webber, Special Edition Using Java 2 Platform; Copyright 1999 by QUE. However, the extraction of the pixel values and the method for writing to an output file would be modified in accordance with aspects of the present invention.

In order for the applet of Data Object 102 to write to a file, it is necessary to set the policy file on the computer system being used to allow write access. A sample policy file associated with this applet is illustrated in Figure 3. Figure 3 is a computer screen shot of a sample security policy file contents to enable an applet to write to a file. This file is used by an applet to write the (x, y) coordinates of the bitmap image to an ASCII text file, which is then read by a post-processing tool and stored in MS (*Microsoft*) Excel. Note that the ASP (***A**ctive **S**erver **P**age*) version of this image reader does not require a security policy file as the ASP version executes on a server. An applet version of this method was illustrated here to demonstrate the capability of reading the ASCII (x, y) data points from a bitmap image. And, Figure 4 illustrates a sample command window showing execution of the applet via the JDK *appletviewer* command tool and the writing of the applet to an output file named output.txt.

Upon execution of the applet utility, the *appletviewer* renders, or echoes, the original image applet) as specified in the applet. This image rendering is not required, but merely serves to verify that data were correctly read by the applet. This is illustrated in Figure 5.

The ASCII text data now exist in a file (output.txt). The contents of this file can be written by Data Object 102 using either of the methods defined above (that is, in C++ or Java), or any other method. Once available in text form, the data can be imported directly into either a database or a spreadsheet.

An example of these data as they exist in an Excel spreadsheet is as shown in Figure 6. These data were generated using a java applet.

To illustrate the content of the data, a simple plot was generated by Signal Processor 103 and Imaging Object 104 using the pixel values to illustrate that indeed the data are representative of the original ECG image. This is shown in Figure 7. Figure 7 is a computer screen shot of a plot of pixel values in the present invention drawn using the raw data captured by a utility within MS Excel. These are the same data points as shown in Figure 6. Upon reading the data, y-component elements are reversed (because the (0,0) element represents the top-left-hand of the screen in the image file, and the maximum extent of the image file, defined by the point (xMax, yMax) is the bottom right-hand corner) so the image appears flipped vertically. This is rectified by the ASP post processor.

Sample pixel values are plotted to illustrate reconstruction of ECG waveform. Importing the data directly into an Excel spreadsheet can be accomplished with the aid of any program that can establish a connection via a COM interface to Excel, or can be accomplished in ASP using an ADO interface. An MS Visual C++ program is one example of how to accomplish this goal. The result of running this program is an Excel spreadsheet file, which contains two columns of data: the x and y pixel coordinates of the ECG.

In Excel, a named range must be defined so that an ASP can read the data via an ODBC driver. An example of the named range of the data is shown in Figure 8, and corresponds to columns A and B within the Excel spreadsheet (that is, the 'X' and 'Y' pixel ranges). Figure 8 is a computer screen shot of setting the data range and defining a name in preparation for reading in the data to be processed by the present invention. This defined named range enables an Active Server Page (ASP) to read directly from the MS Excel file using the handle provided by the named range specified in the figure. The named range is employed by the Open Database Connectivity (ODBC) manager to link the contents of the MS Excel file to the ADO (***ActiveX Data Objects)*** used within the ASP so that the contents of the MS Excel file can be read automatically. The output in this embodiment must have the column

labels inserted, which may be included in the MS Visual C++ program output for titles, as shown in previously mentioned program code.

This is shown visually in Figure 9. It is now possible to process the raw data using a server-side script of Signal Processor 103 and Imaging Object 104, such as an Active Server Page (ASP). Figure 9 is a computer screen shot of an updated MS Excel spreadsheet illustrating required column names in the present invention. When reading MS Excel data using an ADO within an ASP, the first row of data within the Excel spreadsheet is automatically interpreted as the labels of the associated spreadsheet columns.

The ASP program retrieves the spreadsheet data and displays using the intrinsic Excel mathematical and statistical functions to process the data and then to write the processed result to the Web browser. In this way, processed results may be disseminated to a desktop or laptop computer, Cell Phone, or PDA in formats consistent with these devices.

The ASP program produces tabular values of the pixel coordinates associated with the bitmap. However, more interesting is the display of Figure 10, that may be produced by the ASP program. . This program combines the ODBC reader of the previous example with a VBScript plotter. The methods make_1d_array, make_2d_array, and plot_2d disclosed in Ronald H. Nicholson, Javascript Function Plotter, published in 1996, and may be modified and used to achieve the present invention. This utility is preferably embodied in the SSPU.

It is important to note that this use of ASP code is for purposes of illustration only. For example, the code described above contains hard values for specific image sizes. However, to make this code more portable and general, it must be able to accept an arbitrary size image. This code is tailored to support only the specific size image for the worked example. The implication is that the code can only (presently) support an 800 x 800 pixel image, but the present invention is not limited thereto. Naturally, the code itself must be made more flexible to be able to support any ECG trace of any size and to display the data desired by a user. The purpose in showing Figure 10 is to illustrate that,

via a Web interface, the user has access both to the raw data points and to the image itself.

In the APU, a client-side plotting function preferably reads the ASCII data values transmitted to it by a server-side application (ASP). The client utility produces the plot shown in Figure 11.

In a further embodiment of the present invention, 24-bit color image files can be read for purposes of analysis, and generated from data and analyzed results. This aspect of the present invention may be accomplished, for example, as a macro built within an Excel spreadsheet, but those of ordinary skill in the art will appreciate that the present invention is not limited thereto.

To carry out this aspect of the invention using the aforementioned macro, the user needs to launch Excel, although the reason for this is that the utility, *ReadBMP.vb*, integrates its processing of the output data with Excel spreadsheet utilities. In other embodiments, this utility need not be integrated within Excel in order to function in its basic role of reading the ASCII pixel values.

The process of creating this utility is summarized here to clarify the process of its creation. Once Excel is launched, the user would select Tools from the file menu and select Tools→Macro→Macros. The user will be presented with a window, as in Figure 12, in which he/she may type the name of the macro. The macro illustrated herein has been designated as *ReadBMP*.

The visual basic macro itself opens a 24-bit color bitmap using the "Open For Binary Access Read" statement, as follows: "Sub ReadBMP() [Intermediate code removed] Open [filename] For Binary Access Read As #1". Here, [filename] corresponds to the fully qualified path and name of the bitmap file. If this value is hard-coded, then the user will have to save the program and make changes every time a new file is to be read. Therefore, in the embodiment of the present invention, the user may enter this information in a worksheet—called Input_Sheet—that is read by the macro. The file name is then read from the input worksheet dynamically. In this way, the user can enter any file name desired without altering the macro itself. This input file worksheet is shown in Figure 13.

As shown in Figure 13, the File Path and File Name cells identify the location and name of a particular bitmap file. In this case, the file path is "D:\Data\My Pictures\" and the file name is "rainbow.bmp" (this file is used to illustrate the flexibility and capability of the ReadBMP utility to access individual pixels of various colors from the bitmap file). Thus, in the utility itself the generic [filename] may be replaced with specific code that can read these cells from the particular workbook. Here, the String variables ifpath, ifname, and ifiles merely identify the path, file name, and concatenated file path and name, respectively, of the bitmap to be read by the utility. Similarly, the output path (for writing to the binary output file) is defined in a similar manner. In the example shown, all ASCII data are written to a worksheet entitled "pixelCoords". Therefore, all pixel data will be saved to this spreadsheet.

Once the bitmap file is opened for read access, it is necessary to read in the individual elements of data. Because the file is opened for binary access (as is necessary when reading bitmap or other image files), the individual elements of data will be read from the file in the form of bytes. This is accomplished in the example program code by declaring a variable, x, as being of type Byte: Dim x As Byte. The variable, x, may now be used to read in all elements of data from the file. In order to do so, the header information must first be read from the bitmap file. The header provides information regarding the bitmap's height, width, and color scheme (that is, 24-bit, or 8-bits per each color—Red, Green, Blue). This information is used by the macro to determine when, in reading these data, the end of each line of pixels has been reached.

First, the width and height of the bitmap must be determined. Each byte is read from the bitmap file using the Get command. The Get command reads the file unit number (defined as #1), the record number (optional), and the actual data (defined as x). The 19th element read from the bitmap file represents the height of the bitmap. The records read from the bitmap are tracked using two counting variables: amt and recNumber. The variable 'amt' is used to keep track of where in the bitmap the program is, while the variable 'recNumber' is used to write the pixel data to the new bitmap file. The width is read, which is separated from the height by three bytes, and then the depth of

the bitmap is read. This defines the depth as being 24 bits: one 8-bit segment for Red, one 8-bit segment for Green, and one 8-bit segment for Blue. Thus, the color of any given pixel value is given by {RGB}, where R varies between [0..255], G varies between [0..255] and B varies between [0..255].

Next, the start of the first pixel within the bitmap is read. And, the entire bitmap is read in this manner at this point. The 'amt' variable is preferably re-initialized to 1 to keep track of how many data elements are being read. It is read byte by byte into the variable x. The row and column location of the pixel is tracked by noting the quantity of pixels read in relation to the height of the bitmap. For every row and column pixel value, there correspond three values: an R-value, a G-value and a B-value (0..255,0..255,0..255). Pixel color selection and identification has been automated by incorporating a code segment that enables the operator to state (using the associated color word value) a desired color to be sought within the bitmap file. This is illustrated in Table 1. The utility maps the word value for color to an associated {RGB} color byte value, and then uses the associated color byte value to seek for the matching byte combinations within the bitmap file.

Table 1: Supported Colors

	R	G	B
Red/red	255	0	0
Yellow/yellow	255	255	0
Blue/blue	0	0	255
Green/green	0	255	0
Brown/brown	128	64	0
Purple/purple	128	0	128
Pink/pink	255	0	255
Black/black	0	0	0
White/white	255	255	255
Gray/gray	128	128	128
Aqua/aqua	0	255	255

To provide the capability to select more colors than these (and the list is effectively unlimited), then it is necessary to identify the three-color byte values and add the appropriate logic to the Visual Basic Macro. One of ordinary skill in the art will appreciate that this process could also be automated too, for example, by reading in the values from a file or by interfacing the utility directly with a visual color generation utility (such as a color palette) in which the user selects an analog color which is then automatically translated into the associated {RGB} byte values.

If the user wishes to select all red pixel values from a bitmap, then the macro can simply be directed to search for a combination of byte values that equal {255, 0, 0}. A description of the code that accomplishes this follows. The file is read in its entirety. The byte values have been read in triples—that is, for each {row, column} pixel value, there are three bytes: one corresponding to red, green, and blue, respectively. Each time a pixel "triple" is read, it is checked to see what "color" it is by evaluating the three byte values against three known values. These values are selected in the Input_Sheet via a field that enables the user to specify the desired color to extract from the bitmap (refer back to Figure 13). In Figure 13, the second element of the third column is a user-specified color value: the user types in the selected color, which must correspond to one of the colors shown in the chart on the right-hand side of this sheet. The variables are preferably assigned inside *If...then* statements.

The process by which a raw binary file is opened was described above. The file path and file name are captured using variables. These variables were initialized in the Input_Sheet as shown in Figure 13. Writing to the new bitmap file is simple: every time a byte is read from the input file, write it to the output file. However, in the case of the output, the specific record number written to the file must be tracked.

As each byte is read from the input file, the values are held in byte variables and the tracking variable is incremented. These byte values are not yet written to the output file. First it is necessary to determine whether the input combination of byte variables matches with the user-specified color, defined by the known color values. Once a color combination has been found that

matches with the desired color, these colors must be written to the output file in the correct order. This is preferably begun by writing the B color as the byte element in the (x – 2) position, the G color as the byte element in the (x – 1) position, and the R color as the byte element in the x position. Once this is done, the tracking variable may be incremented in preparation for reading and positioning the file pointer to the next location within the output file.

However, if a pixel that matches with the desired color is not found, the output file cannot be left blank in that location; the position of the colored pixel location in relation to all other pixels must be maintained. This has been achieved in the example by writing white pixels to those locations that don't correspond to the desired color (note RGB combination: {255,255,255}).

The pixel data is extracted using a binary read method in which the actual size of the file is determined directly from the file header prior to reading the pixel data. This allows for reading of arbitrarily sized files without having *a priori* knowledge of the image. The raw binary data are extracted in three 8-bit segments per pixel (corresponding to a 24-bit color bitmap) in which the exact knowledge of each pixel color is known and may be used to discriminate actual signals from background noise. Once extracted, the pixels are compared with a user-specified desired color map, which will selectively extract those pixels corresponding to those desired for extraction by the user. The method will then write out the ASCII pixel coordinates of those extracted values and re-create a new bitmap file (in a format compatible with commercial paint or image manipulation programs) containing only the color and geometric reproduction of the user-specified image.

The extracted pixels values represent the two-dimensional (x,y) locations of the user-specified colors extracted from the original bitmap. Given these locations represent the trace of an ECG signal, they may be analyzed to determine frequency (that is, heart rate) electronically from the raw data. Alternatively, individual segments comprising the ECG trace may be studied (such as is necessary when determining ST segment length or for determining whether artifact (noise) exists within an ECG signal). Another advantageous feature of the present invention is the addition of a tracking algorithm to the

data extraction method. A tracking algorithm (such as a Kalman or Batch-least-squares filter) allows predicting changes in ECG frequency, ascertaining whether changes in frequency or ECG signal composition are statistically significant (and, thereby, important information for the clinician or care provider), or are normal in comparison with other patients of similar physiological background (body surface area, age, gender, clinical presentation).Capturing the pixel data in this way also facilitates compression of the key signal components for archiving and retrieval of the signal data. For example, by storing only the actual pixel (x,y) coordinate values associated with the signal itself (ignoring unused white space surrounding the signal) reduces the amount of storage required for the actual signal. Furthermore, recreation of the original signal value is accomplished easily since the (x,y) component positions of the signal are stored, and the only remaining task in recreating the actual signal is the placement of filler white space around the signal (as is illustrated in the example of Figures 17 & 18(a-c)).

To enable rapid execution, an AutoShape figure created on the sheet using the Drawing Tool is assigned to the macro *ReadBMP*. This enables the user to run the utility conveniently from the input sheet. The assignment of the macro to the shape is accomplished simply by right-clicking on the object and then selecting "Assign Macro" from the menu list. A menu window appears, as shown in Figure 14, and the user then selects the macro and clicks "OK".

In operation, the first thing a user does is to identify the location of a bitmap. This location is entered into the "File Path" cell on the Input_Sheet. The user then enters the name of the bitmap in the "File Name" cell. The user then types in the desired color pixels to extract from the bitmap. Once accomplished, the user simply clicks on the macro button. Depending on the size of the bitmap, the actual extraction process takes anywhere from a few seconds to a few minutes (this performance is also machine processor and memory dependent). Once complete (the cursor will change back from an hour glass to its normal appearance), the user may click on the output sheet. As stated earlier, all output in this example is directed to worksheet sheet5, although this is clearly not required.

The rainbow.bmp file shown in Figure 15 was created using MS Paint. The input screen for this example was actually shown in Figure 13. The color "red" is selected. The corresponding output is shown in Figure 16. Figure 16 is a computer screen shot of a processed output from the bitmap file of Figure 16., illustrating the extraction of a single color from the bitmap file described by Figure15 used in the present invention The pixel coordinates associated with the color RED are shown in column format to the left of the plot of those coordinates, provided as a visual aid to verify the correct coordinates have indeed been extracted. Columns A and B in Figure 16 are the row and column pixel locations for all red pixel elements. Columns A and B have optionally been plotted using the chart tool (selecting XY (scatter)). The plot of these data is shown on the accompanying graph (shown on the right of the worksheet). As can be seen from Figure 16, the pixel values are a true representation of the bitmap. However, the exact pixel locations are also revealed.

The generated bitmap, test.bmp, is shown in Figure 17. Figure 17 illustrates one capability of the *ReadBMP* utility to re-create a bitmap image of the selected color extracted from the original image. The ReadBMP utility, after querying the user for a particular color choice, provides those coordinates in ASCII format (described in the narrative text of Figure 16) and then creates a new bit map file using only those coordinates placed at the same location as they appeared in the original bitmap file. This figure is a computer screen shot of the content of this new file, called test.bmp, in the present invention.

The file may also be regenerated with different colors selected. This is shown in Figures 18(a)-(c) with the selection of blue, green, and pink from rainbow.bmp.

The ASCII data may be retrieved from a spreadsheet (or a database) and transmitted remotely to a user, such as by writing the data via an active server page to a client computer in the form of an HTML page. The HTML page would contain a call to a Java applet that then reads the ASCII data and plot the raw data. The applet may also plot overlays of any number of additional (x,y) data sets on the same axes. In the example described below, an application is described in which the applet Plotter Utility (APU) plots the

coordinates of a least-squares generated best-fit regression curve on top of raw data supplied by a server side Visual Basic script. This aspect of the present invention obviates the need to perform server-side plotting and passing of either GIF or JPEG images over the Internet. In addition, the active server page determines the best form of the regression curve (linear or quadratic) before writing the data to the client for plotting.

Superior performance may be achieved in terms of transmission of ASCII data (text) over the Internet versus transmitting an image of a plot (created on the server and transmitted to a thin client). Processing takes place on both the server and the client: on the server, data are retrieved from the spreadsheet, are fit to a linear or quadratic equation, and are formatted into an HTML page within applet parameter calls; on the client, data are read and plotted.

Several alternatives for implementing this exist. These include creating a plot using CGI or server-side script and then displaying the image as a GIF or JPG on the client; and hard-coding all data within an applet and plotting entirely using client processing; creating an image using a spreadsheet program and copying the image to a file and then displaying the plot image via an active server page or an HTML page on a client (requires off-line processing to prepare image for display either via active server page or HTML page). Of course, those of ordinary skill in the art will appreciate that other embodiments are also possible.

In the preferred embodiment, an active-server-page and applet-driven utility for plotting and automatically curve-fitting XY data using a polynomial best-fit regression technique are used. XY data may be displayed in a Web browser via a Java applet and an Active Server Page (ASP) extracted from the Excel spreadsheet. The raw data for plotting may be maintained in the Excel spreadsheet Workbook, located on the server. The user modifies this file at will and then plots the data using the applet executed remotely from the ASP. The data for plotting is read by the ASP and placed in the parameter tag fields of the applet. The ASP then executes a linear and quadratic least squares regression function, resulting in a curve-fit that is also supplied to the applet via

parameter tags for plotting as an overlay on the raw data. The plot then appears in a standard Web window. The regression function creates a best-fit overlay curve based on either a linear or quadratic fit (best choice defined in the least-squares sense by the ASP from the raw data points).

In a further embodiment of the invention, an active-server-page and applet-driven utility may be used for plotting and automatically curve-fitting XY data using a minimum best-fit regression technique. XY plotting data in a raw ASCII form are preferably stored on a remote server in a conventional manner. These data can originate from a text file, a database, or a spreadsheet. For the purpose of this example, these data originate from a spreadsheet. The data, once stored, are retrieved by the GenericPlotter active server page function where they are processed to determine a generalized least squares best-fit function. This best fit function results in the production of coefficients of a regression equation that are used to produce an overlay of data illustrating the goodness of fit between the raw data and the newly-determined regression function. Together, both the raw plotting data and the regression data are written to an HTML page containing both a call to and parameter tags for use by an applet, which remains resident on a client machine. Upon writing these data to the HTML page, these data become fixed and resident within the HTML page of a client machine. The retrieval of the HTML page by the client machine is accomplished automatically by specifying the name of the active server page in the URL window associated with the name of the server's domain.

The client machine maintains as a plug-in a Java applet which is designed to read the parameter tags contained within the HTML page. The Java applet function retrieves the original ASCII data and the best-fit regression data contained within the parameter tags of the HTML form now resident on the client machine and reads these data into arrays within the applet class file. The applet class file processes these two data sets and creates within the browser window resident on the client's machine a two-dimensional plot containing both the raw data and an overlay of regression data. The plot itself, while resident on the client machine, is generated solely by the client, thereby obviating the need to pass an image file from server to client.

The system of the present invention differs from existing automatic plotting processes in several ways. The raw ASCII data are written directly from the server to the client. No image or plot is written from server to client. The client is fully responsible for generating the actual plot via the plotting functions contained within the applet. This is highly advantageous where transmission bandwidth is limited. Also, HTTP tunneling is advantageously not required, which identifies the server machine with which the client machine must be associated. In this way, the client machine is not tied physically to one server. Rather, any server containing the active server page that generates the applet call is sufficient to supply the data for the client machine. applet Java Security policies are advantageously not altered or breached in the process of reading the data from the server because the applet within the client is reading fixed values directly from the parameter tags contained within the applet call in the HTML page.

In operation, the execution takes place on both the server and the client. A Raw ASCII data file containing the data to be plotted either exists or is created by an external process or an individual. In the current embodiment, this file is read by Excel and is placed in a table, which is then saved under a specific name ("rawData.xls", in this example). The relationship between this aspect of the invention and the previous is that the raw ASCII data created by the ReadBMP utility already provide the data in the form necessary for direct access by the ASP. Once stored, an active server page reads this file and places the data in the appropriate format for calling within the resulting HTML page to be displayed on the client. Meanwhile, on the client, the sending of the HTML to the client's browser triggers the retrieval of the applet on the client, now executing in the client browser. The applet retrieves the parameters written to the HTML page on the client and displays a plot within the browser window, together using linear least squares which renders the best-fit curve to be overlaid on the raw data in addition to the governing best-fit equation.

The process of causing an ASP to read an MS Excel named data range is described in the open literature, so the exact steps will be omitted from this embodiment. However, this named range enables an Active Server Page

(ASP) to read directly from the MS Excel file using the handle provided by the named range specified in the Figure. The named range is employed by the Open Database Connectivity (ODBC) manager to link the contents of the MS Excel file to the ADO (*ActiveX Data Objects)* used within the ASP so that the contents of the MS Excel file can be read automatically. The following steps describe the process of setting up the Excel spreadsheet. The steps begin with the assumption that the spreadsheet has been open. Steps 1 through 3 below detail the creation of a named range of cells within the Excel spreadsheet.

1) Enter data in (x,y) format: one value per cell. The first row of the worksheet must be the name of the x and y data columns, respectively

2) Create a name for the range of cells from which the Web page will retrieve the data. Click and drag the mouse across the cells to include within the range.

3) Select Insert->Name->Define-> Enter the name of the selected region in the worksheet. Name of selected region below is "dataFields" (where "dataFields" represents an example name for the range used to illustrate the functioning of the APU within the embodiment).

Once the spreadsheet has been created and a selected range of cells has been defined, it is necessary to make the spreadsheet range accessible as a ActiveX Data Object (ADO). This procedure will permit the active server page to open and retrieve the data within the Excel spreadsheet. This is begun by selecting "database administration," found by going to the Start button and proceeding as follows: Settings->Control Panel->Administrative Tools->Data Sources (ODBC). Double-clicking on the Data Sources (ODBC) icon will bring up the window shown in Figure 4. Be sure to click on the System DSN (*Data Source Name*) tab.

Clicking on the Add button adds a new data source to the System Data Source area. Select "Microsoft Excel Driver (*.xls)" and click on Finish. The Data Source Name ("rawData" in this case) is then added. Then, Workbook is selected. Pressing Select Workbook will create a navigation window. The user must then navigate to the specific Excel file located on the system's storage drive, select the file, and close out the ODBC Administrator tool. Once

completed, this permanently links the specified Excel spreadsheet, named range, and workbook to an ADO which is then accessible within the body of the ASP code.

The applet is the plotting engine. It receives as parameters the raw data read by the active server page from the Excel file via the ADO interface and converts those raw data points into a plot, displayed in the Web browser window. The applet reads the data into character string arrays and then converts them to floating point values. The data are then evaluated to determine maximum and minimum X and Y coordinate limits. Axes are drawn in the Web Browser window, and the data points are converted into (X,Y) pairs scaled to the size of the axes. The paint method uses a drawLine() function to convert each (X,Y) pair into a line segment between the current and next set of data points read in. Two plots are drawn: the raw data and the best-fit function (determined using least squares within the active server page). Both are input via a parameter array within the applet.

A preferred embodiment of the applet is discussed below to identify key features of the applet and raise their importance to the attention of the reader. Data are read by the applet through parameters passed in via the active server page into arrays. These arrays are initially filled as text strings and then translated into floating point values. Two sets of floating point arrays are created: one set for the raw and another set for the best-fit model data. A SloadData[] string and the SloadModel[] string are used to read in the data from the applet parameter list. These strings feature both the X and Y data. These data are then transformed into floating point values and stored in a Data[] and Model[] arrays, respectively. The X and Y components are then separated and stored in an xData[], yData[], xModel[], and yModel[] arrays. The size of these arrays is arbitrarily set to 1500 elements (implies 3000 raw data values and 3000 best-fit values). Those of ordinary skill in the art will appreciate that by using vectors, it would be possible to dynamically dimension these arrays to be as large or small as required by the quantity of data contained in the applet parameter list. Other embodiments of this method for other applications have employed the use of vectors. It is important to note that the present

embodiment does not represent the most efficient implementation of the language constructs. Development efforts focused on improving operational efficiency can no doubt increase the performance and reduce memory consumption of the methods. It is important to note, however, that in specialized circumstances in which operation is limited by physical memory and the characteristic sizes of data are known in advance, specifying these as fixed quantities frequently represents the most efficient embodiment.

The data are read in a usePageParams() method via the following two lines of code: SloadData = getParameter("loadData"); and loadModel = getParameter("loadModel"). An extractData() method is then used to translate from text to numerical values. First, the length of the data array must be determined. Next, it is necessary to loop through this array using an index variable to determine whether the end of the string has been reached. Instances of commas (",") are used as delimiters in the applet parameter list. Whenever one is found, the program knows that it has reached the end of another data value. Now, the Data[] array is filled with all of the data elements, commas removed, and the data are separated into X and Y components. The same steps as above are then performed for the model data.

Once the data are read, they need to be scaled and drawn in a grid. The grid area is drawn using a drawGrid() method, which also measures ticks by the quantity specified in the applet parameter list. The method also draws the numerical values next to the axis major tick marks. Presently, the selection of minimum and maximum values in the X and Y coordinate directions are based on the minimum and maximum of the raw data and model data (whichever are smallest and largest, respectively). The values of both the raw data and the model are compared with the local minima and maxima, and if smaller or larger, are assigned to the new values of the local minima and maxima, ultimately becoming the global minima and maxima, respectively.

The data scaling and plotting are accomplished in the example in one step. First, the scale factors for the X and Y component directions are defined with respect to the size of the browser plotting window and the range of the maximum and minimum values in the X and Y component directions. Next,

these scale factors are applied to each data point inside of a FOR loop that draws a spline between the current value of the data points and their next value. The same method is then applied to scaling the model data as was applied to the raw data above.

The active server page in this example is written in VBScript. However, those of ordinary skill in the art will appreciate that the page could also be written in other languages, such as JavaScript. Either scripting language can be employed and will achieve identical results, depending on the preference of the developer. VB Script is frequently employed by those developing ASP (often the native language used within the MS Visual Studio development environment).The first step involved is opening the ADO object for read access. The fields defined within the named range "dataFields" are selected from the Excel spreadsheet. Figure 19 illustrates a sample input file containing (x,y) data points associated with the "dataFields" named range. This file will be used to illustrate the plotting utility later in Figures 20 and 21.

The ADO connection is then opened for reading. The data elements are read into arrays. Then, the file is read using a WHILE loop, which reads to the end of the named range. The applet itself is called within the BODY text segment of the code. The data just read is written as parameters within the scope of the applet.

In this example, the raw data and the model data are both written as parameters from the active server page. Thus, the applet is loaded on the client and receives its data from the HTML page written as a result of the execution of the active server page. In this way, only ASCII text is passed from the server to the client, as opposed to server-based plotters, in which JPEG or GIF data are written to the client. This provides the significant advantage when there is a concern over client-server bandwidth, as only text-based data is passed between the two, and the quantity of text-based data is relatively small (that is, representative of fewer than several hundred data points).

The model data is created using a subroutine, cFitLine(), that finds a least-squares linear or quadratic solution for the data. The selection of linear or quadratic model is made on the basis of a weighted distance parameter

("lineChiSquare", "quadChiSquare") which compares the distance between each raw data point and the best-fit model equation based on y=mx+b (linear) or $y=c1 + c2x + c3x^2$. The model (linear, quadratic) is selected by determining the weighted distance measure from the linear model with the quadratic model. The smaller of the two distance measures results in the winning model. The regression data is preferably written as an overlay on top of the raw data; and optional variance and standard deviation can also be written, together with a chi-square fit parameter indicating the degree to which the regression curve matches the raw data.

The active server page automatically determines the best form of the least squares regression curve using a chi-squared best fit (Mahalanobis Distance) comparator function, and fills the applet parameter tag fields within the HTML page with the data associated with both the raw data read from the spreadsheet and the least squares regression curve. These numeric values are written to the HTML page in the form of a list of ASCII values. This list of ASCII values associated with the raw data and the regression curve overlay are read by the applet on the client machine as arrays of floating point numbers. The arrays of floating point numbers are automatically scaled by the applet and are plotted within the HTML window on the client machine.

The APU method provides the capability to select the type of plot to be created on the client by specifying within the URL parameter list a choice of plot type and the quantity of data points to be plotted. One of the benefits of the APU method is in its ability to plot both the ECG or EEG data directly extracted by the ReadBMP VB Script utility, the C++, or Java equivalent. Figure 20 is a computer screen shot illustrating the execution of the *APU*, with emphasis on the parameters contained within the URL call to the utility. The parameter list requests the plot type and the number of data points contained within the plot. A plot type of "ecg" indicates an ECG plot. A plot type of "xy" indicates a simple graph. Figure 20 illustrates how the GenericPlotter.asp page can be used to select the type of plot and the quantity of data points specified to be plotted by the client. The URL address bar in the instance of Figure 20 reads as follows:

http://ml1jrz0c/GenericPlotter.asp?ptype=ecg&np=300

The ASP page receives two parameters: ptype, which is used to specify whether the type of plot to be generated is an ECG trace, and the number of data point pairings (here, np=300 corresponds to the quantity of (x,y) pairs). Selecting ptype=ecg turns off the curve-fitting capability in the current embodiment and enables the plotting of the single set of raw data on the client. An example implementation of this was shown in Figure 11.

Specifying the following in the URL causes a standard {x,y} plot to be generated, with curve-fitting enabled. Thus, to plot the data of Figure 19, the following URL would be invoked:

http://ml1jrz0c/GenericPlotter.asp?ptype=xy&np=9

Specifying ptype = xy and np=9 tells the ASP page that a standard {x,y} plot is to be generated, and to plot the first 9 data points. Thus, a curve-fit will also be generated and sent to the client. The result of this action is demonstrated in the screen shot of Figure 21. Figure 21 shows the raw data plot with the best-fit overlay of the data, and the best-fit equation. When "xy" plotting is selected, a curve-fitting routine is automatically invoked by the ASP that generates a least-squares fit of the plot data which is overlaid on the original plot. In this example, 9 data points are selected for plotting, from the file depicted in Figure 19.

The present invention has significant benefits over the prior art. Access to the ASCII form of the data through a Web interface enables overlaying analytical tools either through methods written in the ASP code or through external .dll files that promote, for example, the ability to perform signal analysis (e.g., frequency analysis, signal segment length (ST segment) analysis, averaging of the signal, etc.). This may be accomplished through the use of macros contained within the Excel spreadsheet tool itself (that is, Visual Basic macros to support Fourier Transforms, Power Spectral Analysis, Lomb Periodogram and Wavelet analysis, for instance). Alternatively, these methods can be written in Visual Basic or Jscript (or other languages) that can perform this analysis on the raw data inside of the server application. The results can be posted to any Web-enabled device (Personal Digital Assistant, Laptop, Cell Phone).

Because the present invention provides access to the ASCII values of the pixels, it is not necessary to paste the entire bitmap image on the screen. A partial image may be provided on the screen (or segments of an image) without having to pass the raw .jpg or .bmp image from the server to the Web client. The present invention passes ASCII data between the server and the client, which enhances overall performance. Also, because the methods are written in Visual Basic, they can be instantiated as back-end methods inside of already-existing products.

Mathematical processing of regression curve and extraction of raw data occurs on the server machine (not the client), thereby satisfying security restriction associated with the (standard) inability of applets to access data files on a local machine (that is, without disabling special security restrictions on the applet). The present invention ensures that data can be plotted on any machine that can support a Web Browser and a Java Plug-in. Processing on the server and transmission of ASCII data via HTML to client provides a solution for low-bandwidth communications between any server and client since plot is generated on the client, using the client's processor.

Although this invention has been described with reference to particular embodiments, it will be appreciated that many variations may be resorted to without departing from the spirit and scope of this invention as set forth in the appended claims. For example, the embodiments disclosed herein incorporate a single application server, while one of ordinary skill in the art will appreciate that any number of computer system application servers, data servers, and Web servers may achieve the invention. Similarly, in addition to the a separate image reader, data object, signal processor, and imaging object illustrated herein, the software of the present invention can comprise a single application having individual components or a suite of applications, and its form is not particularly limited.

CLAIMS

What is claimed is:

1. A method for converting a graphical trace of acquired patient data into digital data suitable for digital signal processing, comprising the steps of:

receiving data representative of an image including said graphical trace of acquired patient data;

analyzing said received data to identify individual pixels comprising said graphical trace in said image representative data;

generating a plurality of pixel representative data elements in a desired data format identifying the relative location of said individual pixels in relation to a reference point;

generating said desired data format by identifying the relative location of said individual pixels and identifying at least one of, (a) color of said individual pixels and (b) luminance amplitude of said individual pixels; and

storing said generated pixel representative data elements in a file.

2. A method according to Claim 1, including the step of scanning a paper reproduction of said graphical trace to provide said data representative of said image including said graphical trace.

3. A method according to Claim 1, wherein said data representative of said image including said graphical trace is in at least one data format of, (a) a bitmap file format, (b) a graphical interchange format (GIF format). (c) a (Joint Photographic Expert Group) JPEG format and (d) a Tagged Image File Format (TIFF).

4. A method according to Claim 1, wherein said desired data format is an ASCII format comprising:

(x, y, color)i

where (x) represents the x-coordinate position of the i-th pixel; (y) represents the y-coordinate position of the i-th pixel, and color represents the RGB color of the i-th pixel.

5. A method according to Claim 1, wherein said graphical trace of a medical parameter comprises at least one of, (a) an electro-cardiogram trace, (b) an electro-encephalogram trace and (c) a ventilator parameter trace comprising at least one selected from the group consisting of respiratory rate, tidal volume, minute volume, pulse oximetry, etc.), and (d) a blood parameter trace.

6. A method according to Claim 1, further comprising the steps of:

formatting said received data into a format compatible with a procedure located at a remote destination, said procedure being for processing said data representative of said graphical trace to provide processed data suitable for use in presenting said graphical trace in a displayed image; and

communicating said formatted data to said remote destination in response to a user command; and

generating said displayed image at said remote destination using said formatted data.

7. A method according to Claim 1, further comprising the steps of:

decoding said data format to provide a decoded format incorporating data fields conveying said data representative of said graphical trace; and

initiating execution of a procedure for accessing said data fields conveying said data representative of said graphical trace and exclusive of establishing customized security protocol settings enabling access by said procedure to said formatted and encoded received data, said procedure processing said data representative of said graphical trace to provide processed data suitable for use in presenting said graphical trace in a displayed image.

8. A method for converting a graphical trace into digital data suitable for digital signal processing, comprising the steps of:

receiving data representative of an image including a graphical trace;

analyzing said received data to identify individual pixels comprising said graphical trace in said image representative data;

receiving user color selection information specifying a color;

generating a plurality of pixel representative data elements of pixels of said specified color in a desired data format identifying the relative location of said individual pixels in relation to a reference point; and

storing said generated pixel representative data elements in a file.

9. A method according to Claim 9, wherein said storing step comprises storing said generated pixel representative data elements in a file of format selected by a user from formats including, (a) a text file format, (b) a word file format, (c) a spreadsheet format and (d) a database compatible format.

10. A method according to Claim 9, further comprising the steps of:

determining file size of said received data from file header information; and

extracting pixel data for use in said generating step by performing a binary read of said file of determined size.

11. A method according to Claim 9, further comprising the steps of:

extracting raw binary data representing pixel data in three 8-bit segments per pixel;

comparing extracted pixel data against a user-specified desired colormap to selectively filter those pixels corresponding to those of a desired color; and

generating coordinates of the filtered extracted pixel data in a desired format for use in said step of generating a plurality of pixel representative data elements.

12. A method according to Claim 9, further comprising the step of analyzing generated pixel representative data elements representing a segment of said graphical trace to determine frequency of features embodied in said graphical trace segment.

13. A method according to Claim 9, wherein said graphical trace represents a medical parameter and including the step of predicting changes in medical parameter frequency characteristics.

14. A method according to Claim 14, wherein said graphical trace of a medical parameter comprises at least one of, (a) an electro-cardiogram trace, (b) an electro-encephalogram trace and (c) a ventilator parameter trace, and (d) a blood parameter trace.

15. A method according to Claim 14, further comprising the step of determining whether said predicted changes in medical parameter frequency characteristics are statistically significant for a specific patient.

16. A method according to Claim 16, further comprising the step of determining whether said predicted changes in medical parameter frequency characteristics are statistically significant based on medical record information for said specific patient.

17. A method according to Claim 9, further comprising the steps of:

decoding said data elements to provide a decoded format incorporating data fields conveying said data representative of said graphical trace;

initiating execution of a procedure for accessing said data fields conveying said data representative of said graphical trace and exclusive of establishing customized security protocol settings enabling access by said procedure to said formatted and encoded received data, said procedure processing said data representative of said graphical trace to provide processed data suitable for use in presenting said graphical trace in a displayed image.

18. A method according to Claim 9, wherein said generating step comprises formatting said received data into tagged fields in an HTML Active Server Page format.

19. A system for converting a graphical trace into digital data suitable for digital signal processing, comprising:

an image reader programmed to:

(1) receive data representative of an image including a graphical trace;

(2) analyze said received data to identify individual pixels comprising said graphical trace in said image representative data;

(3) receive user color selection information specifying a color;

(4) generate a plurality of pixel representative data elements of pixels of said specified color in a desired data format identifying the relative location of said individual pixels in relation to a reference point; and

a data object programmed to store said generated pixel representative data elements in a file.

20. The system of Claim 21, wherein said data object is programmed to store said generated pixel representative data elements in a file of format selected by a user from formats including, (a) a text file format, (b) a word file format, (c) a spreadsheet format and (d) a database compatible format.

21. The system of Claim 21, wherein said image reader is further programmed to determine file size of said received data from file header information and to extract pixel data for use in said generating step by performing a binary read of said file of determined size.

22. The system of Claim 21, further comprising:

a signal processor programmed to decode said data elements to provide a decoded format incorporating data fields conveying said data representative of said graphical trace; and

an imaging object programmed to initiate execution of a procedure for accessing said data fields conveying said data representative of said graphical trace and exclusive of establishing customized security protocol settings enabling access by said procedure to said formatted and encoded received data, said procedure processing said data representative of said graphical trace to provide processed data suitable for use in presenting said graphical trace in a displayed image.

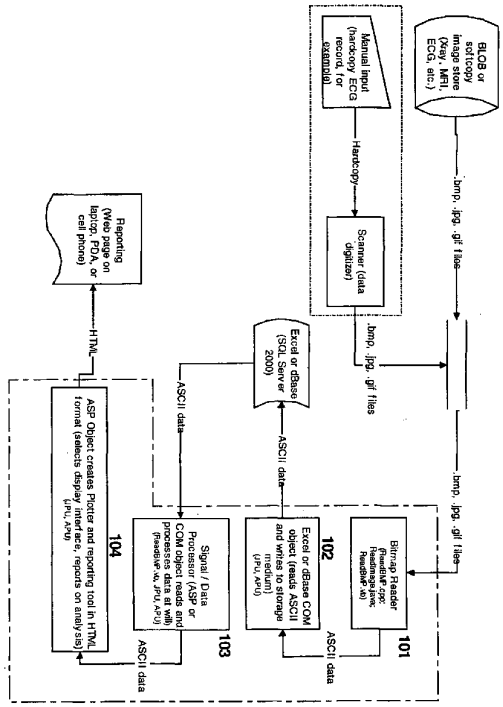

BLOB or ontology image store (Xray, MRI, ECG, etc.)
Manual input (hardcopy ECG record, for example)
Hardcopy
Scanner (data digitizer)
.bmp, .jpg, .gif files
.bmp, .jpg, .gif files
.bmp, .jpg, .gif files
Excel or dBase (SQL Server 2000)
ASCII data
ASCII data
Reporting (Web page on laptop, PDA, or cell phone)
HTML
ASP Object creates Plotter and reporting tool in HTML format (selects display interface, reports on analysis)
104
Signal / Data Processor (ASP or COM object reads and processes data as will) (ReadBlobP.ax, JPU, APU)
103
Excel or dBase COM object (reads ASCII and writes to storage medium) (JPU, APU)
102
Bitmap Reader (reads image type: ReadBlobP.ax, ReadBlobP.ax)
101
ASCII data
ASCII data

Figure 1

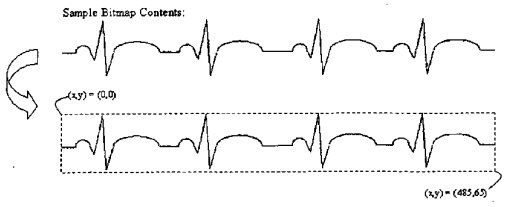


Figure 2

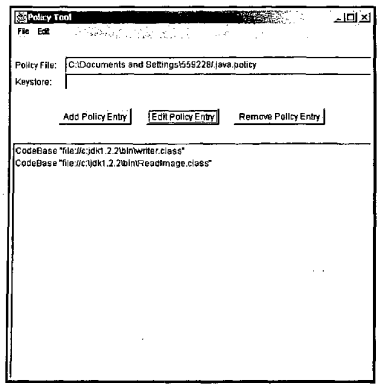

Policy Tool
File Edit
Policy File: C:\Documents and Settings\s558228f\.java.policy
Keystore:
Add Policy Entry
Edit Policy Entry
Remove Policy Entry
CodeBase "file:/c:/jdk1.2.2/bin/writer.class"
CodeBase "file:/c:/jdk1.2.2/bin/ReadImage.class"

Figure 3

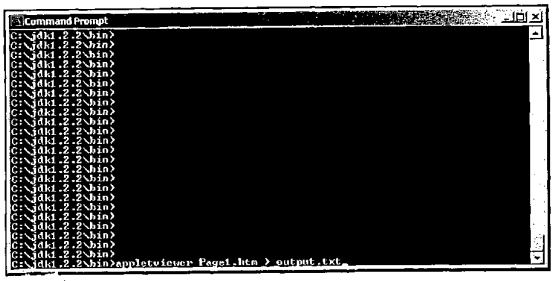

Command Prompt
C:\jdk1.2.2\bin>appletviewer Page1.htm > output.txt

Figure 4

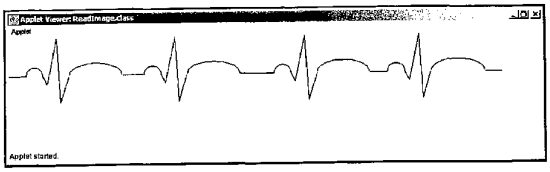


Figure 5

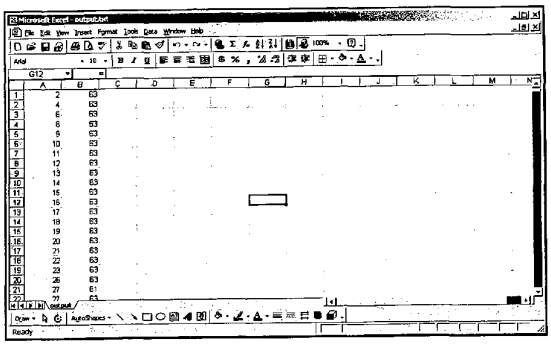


Figure 6

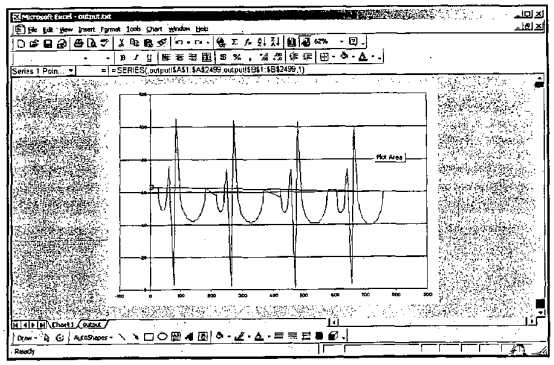


Figure 7

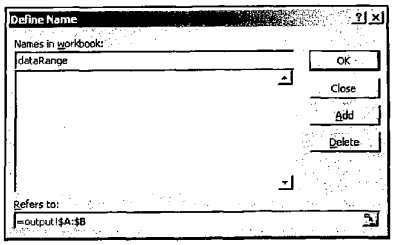

Define Name
Names in workbook:
dateRange
OK
Close
Add
Delete
Refers to:
=output!$A:$B

Figure 8

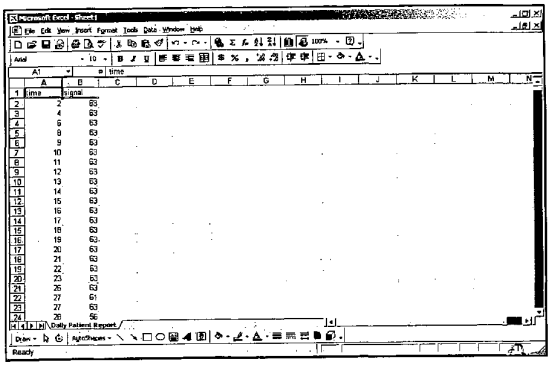

Microsoft Excel - Sheet1
time
time
signal
Daily Patient Report
Ready

Figure 9

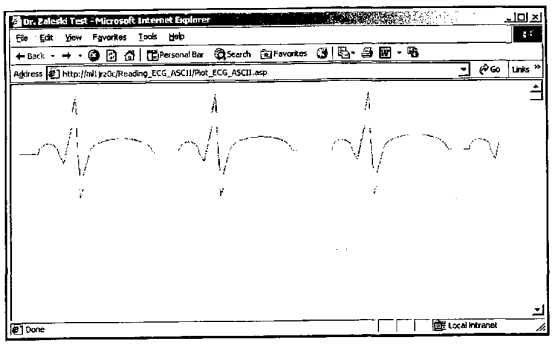

Dr. Zaleski Test - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Personal Bar
Search
Favorites
Address
Go
Links
Done
Local intranet

Figure 10

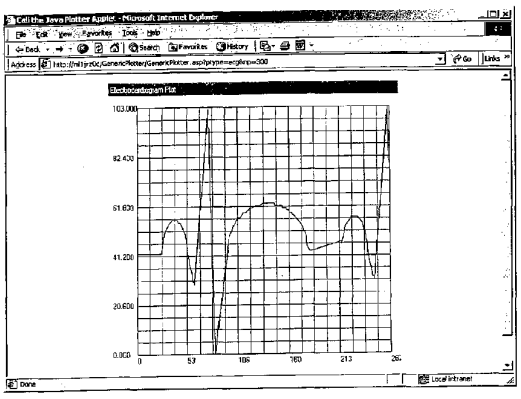

Call the Java Plotter Applet - Microsoft Internet Explorer
Electrocardiogram Plot
103.000
82.400
61.800
41.200
20.600
0.000
0
53
105
160
213
265
Done
Local intranet

Figure 11

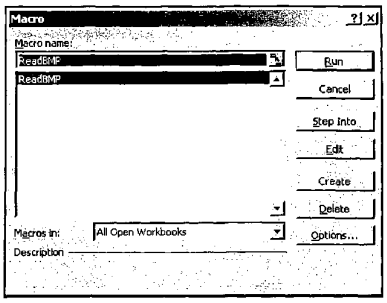

Macro
Macro name:
ReadBMP
ReadBMP
Run
Cancel
Step Into
Edit
Create
Delete
Options...
Macros in:
All Open Workbooks
Description

Figure 12

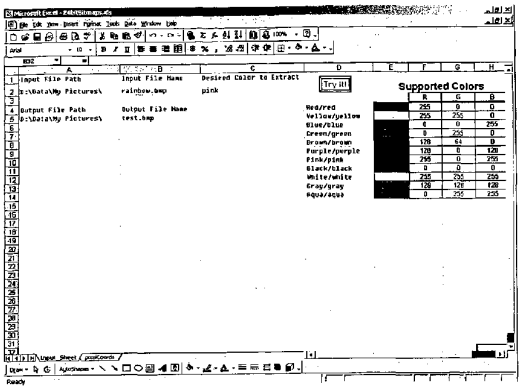

Input File Path
Input File Name
Desired Color to Extract
Try it!
Supported Colors
Output File Path
Output File Name
Red/red
Yellow/yellow
Blue/blue
Green/green
Brown/brown
Purple/purple
Pink/pink
Black/black
White/white
Gray/gray
Aqua/aqua
Ready

Figure 13

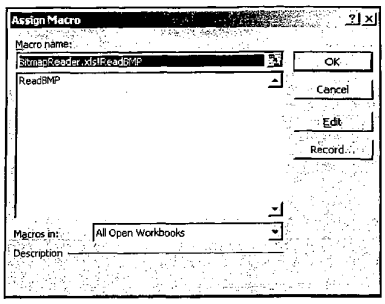

Assign Macro
Macro name:
BitmapReader.xls!ReadBMP
ReadBMP
OK
Cancel
Edit
Record...
Macros in:
All Open Workbooks
Description

Figure 14

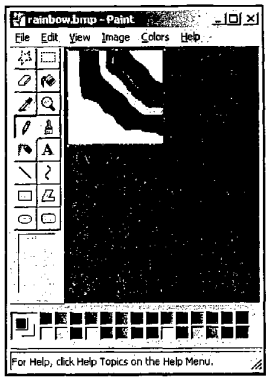

rainbow.bmp - Paint
File Edit View Image Colors Help
For Help, click Help Topics on the Help Menu.

Figure 15

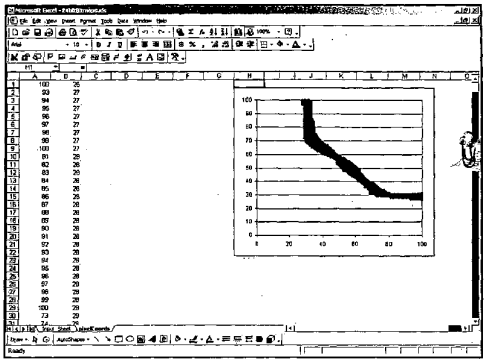


Figure 16

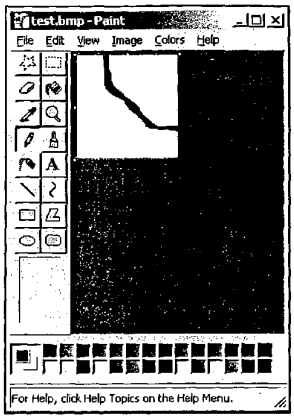

test.bmp - Paint
File Edit View Image Colors Help
For Help, click Help Topics on the Help Menu.

Figure 17

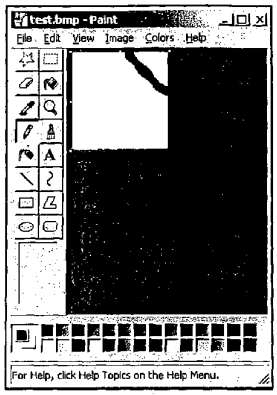

test.bmp - Paint
File Edit View Image Colors Help
For Help, click Help Topics on the Help Menu.

Figure 18(a)

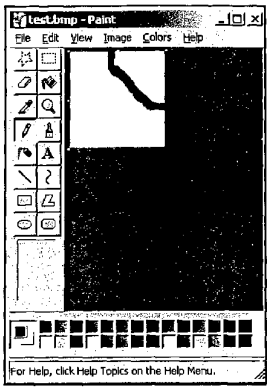

test.bmp - Paint
File Edit View Image Colors Help
For Help, click Help Topics on the Help Menu.

Figure 18(b)

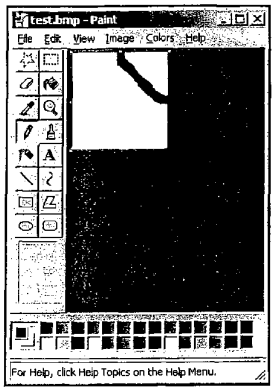

test.bmp - Paint
File Edit View Image Colors Help
For Help, click Help Topics on the Help Menu.

Figure 18(c)

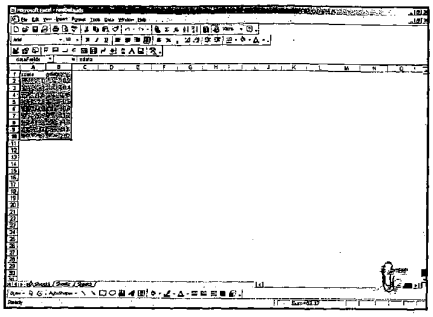


Figure 19

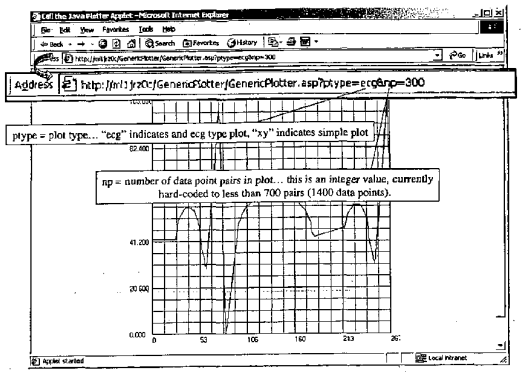

Address
http://nr1yz0c/GenericPlotter/GenericPlotter.asp?ptype=ecg&np=300
ptype = plot type... "ecg" indicates and ecg type plot, "xy" indicates simple plot
np = number of data point pairs in plot... this is an integer value, currently hard-coded to less than 700 pairs (1400 data points).

Figure 20

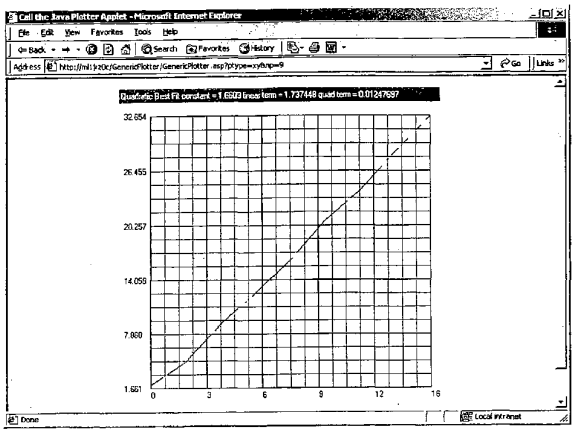

Call the Java Plotter Applet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address
Go
Links
Quadratic Best Fit constant = 1.6503 linear term = 1.737448 quad term = 0.01247697
32.654
26.455
20.257
14.059
7.860
1.661
0
3
6
9
12
15
Done
Local intranet

Figure 21

【国際公開パンフレット（コレクトバージョン）】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 March 2003 (13.03.2003)

PCT

(10) International Publication Number
WO 2003/021517 A3

(51) International Patent Classification⁷: G06F 19/00, 17/00, G06K 9/00

(21) International Application Number:
PCT/US2002/020651

(22) International Filing Date: 1 July 2002 (01.07.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/316,602 31 August 2001 (31.08.2001) US
10/107,229 27 March 2002 (27.03.2002) US

(71) Applicant: SIEMENS MEDICAL SOLUTIONS HEALTH SERVICES CORPORATION [US/US]; 51 Valley Stream Parkway, Malvern, PA 19355 (US).

(72) Inventor: ZALESKI, John, R.; 219 Elmwood Lane, West Brandywine, PA 19320 (US).

(74) Agents: BURKE, Alexander, J. et al.; Siemens Corporation - Intellectual Property Dept., 186 Wood Ave. South, Iselin, NJ 08830 (US).

(81) Designated States (national): CA, JP.

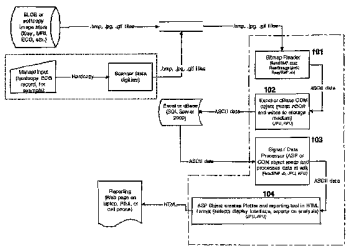
(84) Designated States (regional): European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR).

Published:
— with international search report
— before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments

(88) Date of publication of the international search report:
23 September 2004

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: A SYSTEM FOR PROCESSING IMAGE REPRESENTATIVE DATA



(57) Abstract: The present invention is directed to a system for converting a graphical trace of acquired patient data into digital data suitable for digital signal processing, which includes the steps of receiving data representative of an image including the graphical trace of acquired patient data; analyzing the received data to identify individual pixels comprising the graphical trace in the image representative data; generating a plurality of pixel representative data elements in a desired data format identifying the relative location of the individual pixels in relation to a reference point; generating the desired data format by identifying the relative location of the individual pixels and identifying at least one of, (a) color of said individual pixels and (b) luminance amplitude of said individual pixels; and storing the generated pixel representative data elements in a file. In addition, the received data may be formatted into a format compatible with a procedure located at a remote destination, wherein the procedure is for processing the data representative of the graphical trace to provide processed data suitable for use in presenting the graphical trace in a displayed image; and communicating the formatted data to the remote destination in response to a user command; and generating the displayed image at the remote destination using the formatted data.

WO 2003/021517 A3

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 02/20651

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F19/00 G06F17/00 G06K9/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F G06K G06T

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data, PAJ, IBM-TDB, BIOSIS

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	B. TUMMERS: "DataThief II, version 1.1.0" INTERNET ARTICLE, [Online] 6 November 2000 (2000-11-06), XP002273768 Retrieved from the Internet: URL:www.nikhef.nl/~keeshu/datathief/> [retrieved on 2004-03-15] page 3, line 8 - page 5, line 5 page 15, line 5 - line 26 -----	1-5, 7-11,17, 19-22
X	M.D. SILBERT: "New Graphs from Old" L'ACTUALITÉ CHIMIQUE CANADIENNE, March 2000 (2000-03), pages 8-10, XP002273769 page 9, left-hand column, line 47 - page 10, left-hand column, line 5 ----- -/--	1-5, 7-11,17, 19-22

[X] Further documents are listed in the continuation of box C. [ ] Patent family members are listed in annex.

° Special categories of cited documents :
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
23 March 2004	16 07 2004

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Itoafa, A

Form PCT/ISA/210 (second sheet) (January 2004)

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 02/20651

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	J.E. SILK, E.M. WOOLLEY: "Converting scanned graphs to (x,y) data" AMERICAN LABORATORY, [Online] February 1999 (1999-02), XP002273770 Retrieved from the Internet: URL:http://www.iscpubs.com/articles/al/a9902sil.pdf> [retrieved on 2004-03-15] page 72, middle column, line 10 - page 75, right-hand column, line 49 -----	1-5, 7-11,17, 19-22
T	ANONYMUS: "Digitizing programs for converting hard copy graphs and plots back to data" INTERNET ARTICLE, [Online] 5 April 2003 (2003-04-05), XP002273771 Retrieved from the Internet: URL:http://www.ccp14.ac.uk/solution/hardcopy2data.htm> [retrieved on 2004-03-15] page 1, line 1 - page 3, line 42 -----	1,8,19

Form PCT/ISA/210 (continuation of second sheet) (January 2004)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 02/20651

Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This International Search Report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the International Application that do not comply with the prescribed requirements to such an extent that no meaningful International Search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

see additional sheet

1. ☐ As all required additional search fees were timely paid by the applicant, this International Search Report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this International Search Report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☒ No required additional search fees were timely paid by the applicant. Consequently, this International Search Report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

1-5, 7-11, 17, 19-22

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest.

☐ No protest accompanied the payment of additional search fees.

Form PCT/ISA/210 (continuation of first sheet (1)) (July 1998)

International Application No. PCT/ US 02/20651

FURTHER INFORMATION CONTINUED FROM PCT/ISA/ 210

This International Searching Authority found multiple (groups of) inventions in this international application, as follows:

1. claims: 1-5,7-11,17,19-22

   Retrieval and coding of a region of interest from an image
   ---

2. claims: 6,18

   Transmission of a region of interest retrieved from an image from server to client
   ---

3. claims: 12-16

   Analysis and interpretation of the characteristics of a region of interest retrieved from an image
   ---

フロントページの続き

Ｆターム(参考) 4C117 XA07 XB06 XB09 XB11 XC19 XC20 XD03 XD24 XE17 XE18
XE24 XF03 XF21 XG02 XG19 XG33 XG44 XH04 XH16 XH19
XH27 XJ01 XJ03 XJ05 XJ21 XJ26 XJ27 XK03 XK33 XK39
XL01 XL06 XL12 XL15 XL19 XL26 XM12

专利名称(译)	图像显示数据处理系统		
公开(公告)号	JP2005503851A	公开(公告)日	2005-02-10
申请号	JP2003525783	申请日	2002-07-01
[标]申请(专利权)人(译)	美国西门子医疗解决公司		
申请(专利权)人(译)	西门子医疗系统健康Savishizu公司		
[标]发明人	ザレンスキジョンアール		
发明人	ザレンスキ,ジョン,アール		
IPC分类号	A61B5/00 A61B5/0432 A61B5/0476 G06F19/00 G06Q50/00 G06F17/60		
CPC分类号	G06F19/321 G06F19/3418 G16H30/40 G16H40/67		
FI分类号	A61B5/00.D G06F17/60.126.Q A61B5/04.314.A A61B5/04.324		
F-TERM分类号	4C027/AA02 4C027/AA03 4C027/FF01 4C027/HH08 4C027/HH11 4C027/KK03 4C027/KK05 4C117/XA07 4C117/XB06 4C117/XB09 4C117/XB11 4C117/XC19 4C117/XC20 4C117/XD03 4C117/XD24 4C117/XE17 4C117/XE18 4C117/XE24 4C117/XF03 4C117/XF21 4C117/XG02 4C117/XG19 4C117/XG33 4C117/XG44 4C117/XH04 4C117/XH16 4C117/XH19 4C117/XH27 4C117/XJ01 4C117/XJ03 4C117/XJ05 4C117/XJ21 4C117/XJ26 4C117/XJ27 4C117/XK03 4C117/XK33 4C117/XK39 4C117/XL01 4C117/XL06 4C117/XL12 4C117/XL15 4C117/XL19 4C117/XL26 4C117/XM12		
优先权	60/316602 2001-08-31 US 10/107229 2002-03-27 US		
外部链接	Espacenet		

摘要(译)

本发明是所收集的患者数据的系统的转换成适合于数字信号处理的数字数据的线图表示，它接收表示包括所收集的患者数据的线图表示，所接收的数据的图像数据识别构成图像显示数据中的线图表示的各个像素，并识别各个像素相对于参考点的相对位置像素的显示数据元素在所期望的数据格式生成以识别各个像素的相对位置，以确定一个亮度幅度的（ａ）单独的颜色和中的至少一个（ｂ）中的像素的各个像素从而生成期望的数据格式并将生成的像素显示数据元素存储在文件中。此外，接收到的数据格式，以在远程位置处执行以适合于处理和显示图像的线图显示的形式显示的处理的数据表示的线图目前的数据来改变的程序兼容的格式生成中，响应于用户指令将经格式化的数据到远程位置，以在远程地点产生的显示图像，通过使用格式化的数据那。