(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4340915号
(P4340915)
(45)発行日　平成21年10月7日(2009.10.7)　(24)登録日　平成21年7月17日(2009.7.17)

(51)Int.Cl.　F I
H04N 5/232 (2006.01)　H04N 5/232 Z

請求項の数 29　(全 80 頁)

(21)出願番号　特願2006-24065(P2006-24065)
(22)出願日　平成18年2月1日(2006.2.1)
(65)公開番号　特開2007-208580(P2007-208580A)
(43)公開日　平成19年8月16日(2007.8.16)
審査請求日　平成19年1月29日(2007.1.29)

(73)特許権者　000002185
ソニー株式会社
東京都港区港南１丁目７番１号
(74)代理人　100091546
弁理士　佐藤　正美
(72)発明者　倉田　徹
東京都品川区北品川６丁目７番３５号　ソニー株式会社内

審査官　日下　善之

(56)参考文献　特開２００１－２０４０３８（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｈ０４Ｎ　５／２３２

(54)【発明の名称】撮像画像信号の歪み補正方法、撮像画像信号の歪み補正装置、撮像方法および撮像装置

(57)【特許請求の範囲】
【請求項１】
１画面の画像の水平方向および／または垂直方向の歪みを補正する方法であって、
前記画像の１画面区間を複数個の分割画像区間に分割し、
前記分割画像区間のそれぞれにおける画像の動きベクトルを、２画面間の前記分割画像区間の画像情報から検出し、
検出された前記複数個の分割画像区間の前記動きベクトルのうちの、前記１画面内において隣接する分割画像区間のものの差分ベクトルを求め、求められた前記差分ベクトルに基づいて、前記分割画像区間のそれぞれ内における画像の歪みを補正する
ことを特徴とする画像信号の歪み補正方法。
【請求項２】
請求項１に記載の画像信号の歪み補正方法において、
前記差分ベクトルを求めることにより、前記分割画像区間のそれぞれごとに、当該分割画像区間の画像の歪み速度を検出し、
前記各分割画像区間内においては、当該各分割画像区間について検出された前記画像の歪み速度の時間積分値を前記画像の歪み補正対象部位の歪み変位量とし、前記時間積分値を用いて前記画像の歪みを補正する
ことを特徴とする画像信号の歪み補正方法。
【請求項３】
前記画像の歪みは、撮影時の撮像素子の、撮像画像の水平方向および／または垂直方向

の位置的変化による撮像画像の歪みである
ことを特徴とする請求項２に記載の画像信号の歪み補正方法。
【請求項４】
請求項３に記載の画像信号の歪み補正方法において、
前記撮像素子は、前記撮像画像のデータを画素単位で読み出しが可能のものであって、
前記各分割画像区間内においては、当該各分割画像区間の先頭部位に対する歪み補正対象部位の読み出し遅れ時間を用いて、前記画像の歪み速度を時間積分して得られる変位量を、前記歪み補正対象部位の変位量として前記撮像画像の歪みを補正する
ことを特徴とする画像信号の歪み補正方法。
【請求項５】
請求項３に記載の撮像信号の歪み補正方法において、
前記撮像素子からは前記撮像画像のデータをライン単位で読み出しするものであると共に、前記分割は、前記各分割画像区間が複数ラインからなるように行ない、
前記各分割画像区間内においては、それぞれの前記分割画像区間について検出された前記画像の歪み速度を前記ライン単位に時間積分して得られる変位量を各ライン位置の変位量として前記撮像画像の歪みを補正する
ことを特徴とする撮像画像信号の歪み補正方法。
【請求項６】
請求項３に記載の撮像信号の歪み補正方法において、
前記画像の歪み速度を、水平方向の成分と垂直方向の成分とに分け、
前記画像の歪み速度の水平方向成分を用いて、前記画像の水平方向の歪みを補正し、前記画像の歪み速度の垂直方向成分を用いて、前記画像の垂直方向の歪みを補正する
ことを特徴とする撮像信号の歪み補正方法。
【請求項７】
請求項５に記載の画像信号の歪み補正方法において、
前記画像の歪み速度を、水平方向の成分と垂直方向の成分とに分け、
前記各分割画像区間内においては、前記画像の歪み速度の水平方向の成分を、前記ライン単位に時間積分して得られる前記水平方向の変位量を各ラインの水平方向の変位量として前記画像の水平方向の歪みを補正すると共に、前記画像の歪み速度の垂直方向の成分を、前記ライン単位に時間積分して得られる前記垂直方向の変位量を各ラインの垂直方向の変位量として前記画像の垂直方向の歪みを補正する
ことを特徴とする画像信号の歪み補正方法。
【請求項８】
請求項７に記載の画像信号の歪み補正方法において、
前記画像の水平方向の歪みの補正処理と、前記画像の垂直方向の歪みの補正処理とを並行して行なうようにすると共に、前記画像の水平方向の歪みの補正処理の進捗度合いを参照しながら前記垂直方向の変位量の時間積分処理をする
ことを特徴とする画像信号の歪み補正方法。
【請求項９】
請求項１に記載の画像信号の歪み補正方法において、
前記分割画像区間のそれぞれにおける画像の動きベクトルを検出する方法は、
前記動きベクトルを検出する２画面は、注目画面である参照画面と当該参照画面よりも前の元画面であり、
前記元画面中の前記各分割画像区間において、所定の位置に複数の画素からなる所定の大きさの少なくとも１個のターゲットブロックを設定し、
前記ターゲットブロックと同じ大きさの参照ブロックを、前記参照画面に設定されたサーチ範囲において複数個設定し、
前記複数個の参照ブロックの内から、前記ターゲットブロックと相関の強い前記参照ブロックを検出し、
当該検出した参照ブロックの前記ターゲットブロックに対する画面上の位置ずれ量に基

づいて、前記動きベクトルを検出する
ものであることを特徴とする画像信号の歪み補正方法。
【請求項１０】
請求項９に記載の画像信号の歪み補正方法において、
前記分割画像区間のそれぞれにおける画像の動きベクトルを検出する方法は、
前記参照ブロックのそれぞれにおいて、当該参照ブロック内の各画素の画素値と、前記ターゲットブロック内で対応する位置の各画素の画素値との差分の絶対値の総和を求める差分絶対値和算出工程と、
前記参照ブロックのそれぞれの前記参照画面上の位置の、前記ターゲットブロックの画面上の位置との位置ずれ量を、方向成分も含む参照ベクトルとし、当該参照ベクトルを所定の縮小率で縮小した参照縮小ベクトルを得る参照縮小ベクトル取得工程と、
前記参照縮小ベクトルに応じた大きさの前記参照ベクトルを前記位置ずれ量とする、前記所定の縮小率に応じて削減された数の複数個の前記参照ブロックのそれぞれについての差分絶対値和を記憶する縮小差分絶対値和テーブルを生成するテーブル生成工程と、
前記縮小差分絶対値和テーブルにおける前記差分絶対値和の最小値に対応する前記参照ベクトルを少なくとも用いて、前記参照画面と前記元画面との間の前記分割画像区間のそれぞれについての動きベクトルを算出する動きベクトル算出工程と、
を備え、
前記テーブル生成工程は、
前記参照縮小ベクトル取得手段で取得された前記参照縮小ベクトルの近傍値となる複数の前記参照ベクトルを検出する近傍参照ベクトル検出工程と、
前記差分絶対値和算出工程で算出された前記参照ブロックのそれぞれについての前記差分の絶対値の総和から、前記近傍参照ベクトル検出工程で検出された前記近傍の複数の前記参照ベクトルのそれぞれに対応する差分絶対値和のそれぞれを算出する分散差分絶対値和算出工程と、
前記分散差分絶対値和算出工程で算出された前記近傍の複数の前記参照ベクトルのそれぞれに対応する前記差分絶対値和を、それまでの前記近傍の複数の前記参照ベクトルのそれぞれに対応する前記差分絶対値和に加算する分散加算工程と、
を備えることを特徴とする画像信号の歪み補正方法。
【請求項１１】
請求項９または請求項１０に記載の画像信号の歪み補正方法において、
前記各分割画像区間は、さらに再分割されると共に、当該再分割画像区間毎に前記ターゲットブロックが設定されて、それぞれのターゲットブロックに対する前記動きベクトルが検出され、
前記各分割画像区間について得られる複数のターゲットブロックに対する複数の前記動きベクトルから、前記各分割画像区間についての前記動きベクトルを検出する
ことを特徴とする画像信号の歪み補正方法。
【請求項１２】
請求項１１に記載の画像信号の歪み補正方法において、
前記各分割画像区間についての前記複数のターゲットブロックに対する複数の前記動きベクトルの平均値を、前記各分割画像区間についての前記動きベクトルとして検出する
ことを特徴とする画像信号の歪み補正方法。
【請求項１３】
請求項１２に記載の画像信号の歪み補正方法において、
前記複数のターゲットブロックに対する前記動きベクトルの平均値を求める際に、それぞれの前記ターゲットブロックについての前記動きベクトルが例外となる動きベクトルかどうかを判定し、例外であると判定された前記ターゲットブロックについての前記動きベクトルは、前記平均値を求める演算から排除する
ことを特徴とする画像信号の歪み補正方法。
【請求項１４】

注目画面の画像の１画面区間を複数個の分割画像区間に分割し、
前記分割画像区間のそれぞれにおける画像の動きベクトルを、前記注目画面と当該注目画面よりも前の元画面との間の前記分割画像区間の画像情報から検出し、
検出された前記複数個の分割画像区間の前記動きベクトルのうちの、前記１画面内において隣接する分割画像区間のものの差分ベクトルを求め、求められた前記差分ベクトルに基づいて、前記注目画面の前記分割画像区間のそれぞれ内における水平方向および／または垂直方向の画像の歪みを補正する方法において、
前記分割画像区間のそれぞれにおける画像の動きベクトルを検出する方法は、
前記元画面中の前記各分割画像区間において、所定の位置に複数の画素からなる所定の大きさの少なくとも１個のターゲットブロックを設定し、
前記ターゲットブロックと同じ大きさの参照ブロックを、前記注目画面に設定されたサーチ範囲において複数個設定し、
前記複数個の参照ブロックの内から、前記ターゲットブロックと相関の強い前記参照ブロックを検出し、
当該検出した参照ブロックの前記ターゲットブロックに対する画面上の位置ずれ量に基づいて、前記動きベクトルを検出する
ものであって、
前記参照ブロックのそれぞれにおいて、当該参照ブロック内の各画素の画素値と、前記ターゲットブロック内で対応する位置の各画素の画素値との差分の絶対値の総和を求める差分絶対値和算出工程と、
前記参照ブロックのそれぞれの前記参照画面上の位置の、前記ターゲットブロックの画面上の位置との位置ずれ量を、方向成分も含む参照ベクトルとし、当該参照ベクトルを所定の縮小率で縮小した参照縮小ベクトルを得る参照縮小ベクトル取得工程と、
前記参照縮小ベクトルに応じた大きさの前記参照ベクトルを前記位置ずれ量とする、前記所定の縮小率に応じて削減された数の複数個の前記参照ブロックのそれぞれについての差分絶対値和を記憶する縮小差分絶対値和テーブルを生成するテーブル生成工程と、
前記縮小差分絶対値和テーブルにおける前記差分絶対値和の最小値に対応する前記参照ベクトルを少なくとも用いて、前記参照画面と前記元画面との間の前記分割画像区間のそれぞれについての動きベクトルを算出する動きベクトル算出工程と、
を備え、
前記テーブル生成工程は、
前記参照縮小ベクトル取得工程で取得された前記参照縮小ベクトルの近傍値となる複数の前記参照ベクトルを検出する近傍参照ベクトル検出工程と、
前記差分絶対値和算出工程で算出された前記参照ブロックのそれぞれについての前記差分の絶対値の総和から、前記近傍参照ベクトル検出工程で検出された前記近傍の複数の前記参照ベクトルのそれぞれに対応する差分絶対値和のそれぞれを算出する分散差分絶対値和算出工程と、
前記分散差分絶対値和算出工程で算出された前記近傍の複数の前記参照ベクトルのそれぞれに対応する前記差分絶対値和を、それまでの前記近傍の複数の前記参照ベクトルのそれぞれに対応する前記差分絶対値和に加算する分散加算工程と、
を備えることを特徴とする画像信号の歪み補正方法。

【請求項１５】
撮影時の手ぶれによる撮像素子の位置的変化による撮像画像の歪を補正し、補正後の撮像画像の情報を記録するようにする撮像方法であって、
前記撮像画像の１画面区間を複数個の分割画像区間に分割し、
前記分割画像区間のそれぞれにおける前記撮像画像の動きベクトルを、２画面間の前記分割画像区間の撮像画像情報から検出し、
検出された前記分割画像区間の前記動きベクトルのうちの、前記１画面内において隣接する分割画像区間のものの差分ベクトルを求め、求められた前記差分ベクトルに基づいて、前記分割画像区間のそれぞれ内における撮像画像の歪みを補正し、

前記補正した前記撮像画像の画像情報を記録媒体に記録する
ことを特徴とする撮像方法。
【請求項１６】
１画面の画像の水平方向および／または垂直方向の歪みを補正する装置であって、
前記１画面の画像領域が複数個に分割された各分割画像区間ごとに、注目画面と当該注目画面よりも前の元画面との間の前記分割画像区間の画像情報から動きベクトルを検出する動きベクトル検出手段と、
前記動きベクトル検出手段で検出された前記分割画像区間の前記動きベクトルのうちの、前記１画面内において隣接する分割画像区間のものの差分ベクトルを求める差分ベクトル検出手段と、
前記各分割画像区間内で、前記各分割画像区間について前記差分ベクトル検出手段で検出された前記差分ベクトルに基づいて、前記画像の歪みを補正する歪み補正処理手段と、
を備えることを特徴とする画像信号の歪み補正装置。
【請求項１７】
請求項１６に記載の画像信号の歪み補正装置において、
前記歪み補正処理手段は、
前記差分ベクトルにより、前記分割画像区間のそれぞれごとに、当該分割画像区間の画像の歪み速度を検出し、前記各分割画像区間内においては、前記各分割画像区間について検出された前記画像の歪み速度の時間積分値を前記画像の歪み補正対象部位の歪み変化量として求める変位量積分手段と、
前記変位量積分手段で求められた前記時間積分値を用いて前記画像の歪みを補正する補正手段と
を備えることを特徴とする撮像画像信号の歪み補正装置。
【請求項１８】
請求項１７に記載の画像信号の歪み補正装置において、
前記画像の歪みは、撮影時の撮像素子の、撮像画像の水平方向および／または垂直方向の位置的変化による撮像画像の歪みである
ことを特徴とする画像信号の歪み補正装置。
【請求項１９】
請求項１８に記載の画像信号の歪み補正装置において、
前記撮像素子は、前記撮像画像のデータを画素単位で読み出しが可能のものであって、
前記歪み補正処理手段は、
前記各分割画像区間内においては、当該各分割画像区間の先頭部位に対する歪み補正対象部位の読み出し遅れ時間を用いて、前記画像の歪み速度を時間積分し、その時間積分値を前記歪み補正対象部位の変位量として求める変位量積分手段と、
前記変位量積分手段で得た前記変位量を用いて前記撮像画像の歪みを補正する補正手段と
を備えることを特徴とする画像信号の歪み補正装置。
【請求項２０】
請求項１８に記載の画像信号の歪み補正装置において、
前記撮像素子からは前記撮像画像のデータをライン単位で読み出しするものであると共に、前記分割は、前記各分割画像区間が複数ラインからなるように行なうものであって、
前記歪み補正処理手段は、
前記各分割画像区間内において、前記各分割画像区間について検出された前記画像の歪み速度を、前記ライン単位に時間積分し、その時間積分値を前記歪み補正対象ラインの変位量として求める変位量積分手段と、
前記変位量積分手段で得た前記変位量を各ラインの変位量として前記撮像画像の歪みを補正する補正手段と
を備えることを特徴とする画像信号の歪み補正装置。
【請求項２１】

請求項１８に記載の画像信号の歪み補正装置において、
前記歪み補正処理手段は、
前記画像の歪み速度検出手段で検出された前記画像の歪み速度の水平方向成分を用いて、前記画像の水平方向の歪みを補正する水平補正処理手段と、
前記画像の歪み速度の垂直方向成分を用いて、前記画像の垂直方向の歪みを補正する垂直補正処理手段と
を備えることを特徴とする画像信号の歪み補正装置。
【請求項２２】
請求項２０に記載の画像信号の歪み補正装置において、
前記歪み補正処理手段の前記変位量積分手段は、
前記各分割画像区間内において、前記画像の歪み速度の水平方向の成分を、前記ライン単位に時間積分して、その時間積分値を前記歪み補正対象ラインの水平方向の変位量として求める水平方向変位量算出手段と、
前記各分割画像区間内において、前記画像の歪み速度の垂直方向の成分を、前記ライン単位に時間積分して、その時間積分値を前記歪み補正対象ラインの垂直方向の変位量として求める垂直方向変位量算出手段と、
を備え、
前記歪み補正処理手段の前記補正手段は、
前記各分割画像区間内において、前記水平方向変位量算出手段で得た前記水平方向の変位量を各ラインの水平方向の変位量として前記画像の水平方向の歪みを補正する水平補正処理手段と、
前記各分割画像区間内において、前記垂直方向変位量算出手段で得た前記垂直方向の変位量を各ラインの垂直方向の変位量として前記画像の垂直方向の歪みを補正する垂直補正処理手段と
を備えることを特徴とする画像信号の歪み補正装置。
【請求項２３】
請求項１６に記載の画像信号の歪み補正装置において、
前記動きベクトル検出手段は、前記分割画像区間のそれぞれにおいて、注目画面である参照画面と、当該参照画面よりも前の元画面との間での動きベクトルを算出する手段であって、
前記元画面中の前記各分割画像区間において、所定の位置に複数の画素からなる所定の大きさの少なくとも１個のターゲットブロックが設定されると共に、前記ターゲットブロックと同じ大きさの参照ブロックが、前記注目画面に設定されたサーチ範囲において複数個設定され、
前記複数個の参照ブロックの内から、前記ターゲットブロックと相関の強い前記参照ブロックを検出し、
当該検出した参照ブロックの前記ターゲットブロックに対する画面上の位置ずれ量に基づいて、前記動きベクトルを検出する
ことを特徴とする画像信号の歪み補正装置。
【請求項２４】
請求項２３に記載の画像信号の歪み補正装置において、
前記動きベクトル検出手段は、
前記参照ブロックのそれぞれにおいて、当該参照ブロック内の各画素の画素値と、前記ターゲットブロック内で対応する位置の各画素の画素値との差分の絶対値の総和を求める差分絶対値和算出手段と、
前記参照ブロックのそれぞれの前記参照画面上の位置の、前記ターゲットブロックの画面上の位置との位置ずれ量を、方向成分も含む参照ベクトルとし、当該参照ベクトルを所定の縮小率で縮小した参照縮小ベクトルを得る参照縮小ベクトル取得手段と、
前記参照縮小ベクトルに応じた大きさの前記参照ベクトルを前記位置ずれ量とする、前記所定の縮小率に応じて削減された数の複数個の前記参照ブロックのそれぞれについての

差分絶対値和を記憶する縮小差分絶対値和テーブルを生成するテーブル生成手段と、
前記縮小差分絶対値和テーブルにおける前記差分絶対値和の最小値に対応する前記参照ベクトルを少なくとも用いて、前記参照画面と前記元画面との間の前記分割画像区間のそれぞれについての動きベクトルを算出する動きベクトル算出手段と、
を備え、
前記テーブル生成手段は、
前記参照縮小ベクトル取得手段で取得された前記参照縮小ベクトルの近傍値となる複数の前記参照ベクトルを検出する近傍参照ベクトル検出手段と、
前記差分絶対値和算出手段で算出された前記参照ブロックのそれぞれについての前記差分の絶対値の総和から、前記近傍参照ベクトル検出工程で検出された前記近傍の複数の前記参照ベクトルのそれぞれに対応する差分絶対値和のそれぞれを算出する分散差分絶対値和算出手段と、
前記分散差分絶対値和算出手段で算出した前記近傍の複数の前記参照ベクトルのそれぞれに対応する前記差分絶対値和を、それまでの前記近傍の複数の前記参照ベクトルのそれぞれに対応する前記差分絶対値和に加算する分散加算手段と、
を備えることを特徴とする画像信号の歪み補正装置。
【請求項２５】
請求項２２または請求項２３に記載の画像信号の歪み補正装置において、
前記動きベクトル検出手段は、
前記各分割画像区間を、さらに再分割すると共に、当該再分割画像区間毎に前記ターゲットブロックを設定して、それぞれのターゲットブロックに対する前記動きベクトルを検出し、
前記各分割画像区間について得られる複数のターゲットブロックに対する複数の前記動きベクトルから、前記各分割画像区間についての前記動きベクトルを検出する
ことを特徴とする画像信号の歪み補正装置。
【請求項２６】
請求項２５に記載の画像信号の歪み補正装置において、
前記動きベクトル検出手段は、
前記各分割画像区間についての前記複数のターゲットブロックに対する複数の前記動きベクトルの平均値を、前記各分割画像区間についての前記動きベクトルとして検出する
ことを特徴とする画像信号の歪み補正装置。
【請求項２７】
請求項２６に記載の画像信号の歪み補正装置において、
前記動きベクトル検出手段は、
前記複数のターゲットブロックに対する前記動きベクトルの平均値を求める際に、それぞれの前記ターゲットブロックについての前記動きベクトルが例外となる動きベクトルかどうかを判定し、例外であると判定された前記ターゲットブロックについての前記動きベクトルは、前記平均値を求める演算から排除する
ことを特徴とする画像信号の歪み補正装置。
【請求項２８】
１画面の画像の１画面区間を複数個の分割画像区間に分割し、前記分割画像区間のそれぞれにおける画像の動きベクトルを、前記注目画面である参照画面と当該参照よりも前の元画面との間の前記分割画像区間の画像情報から検出し、検出された前記分割画像区間の前記動きベクトルのうちの、前記１画面内において隣接する分割画像区間のものの差分ベクトルを求め、求められた前記差分ベクトルに基づいて、前記注目画面の前記分割画像区間のそれぞれ内における水平方向および／または垂直方向の画像の歪みを補正する装置であって、
前記分割画像区間のそれぞれにおける画像の動きベクトルを検出する動きベクトル検出手段は、前記元画面中の前記各分割画像区間において、所定の位置に複数の画素からなる所定の大きさの少なくとも１個のターゲットブロックを設定すると共に、前記ターゲット

ブロックと同じ大きさの参照ブロックを、前記注目画面に設定されたサーチ範囲において複数個設定し、前記複数個の参照ブロックの内から、前記ターゲットブロックと相関の強い前記参照ブロックを検出し、当該検出した参照ブロックの前記ターゲットブロックに対する画面上の位置ずれ量に基づいて、前記動きベクトルを検出するものである画像信号の歪み補正装置において、

前記動きベクトル検出手段は、

前記参照ブロックのそれぞれにおいて、当該参照ブロック内の各画素の画素値と、前記ターゲットブロック内で対応する位置の各画素の画素値との差分の絶対値の総和を求める差分絶対値和算出手段と、

前記参照ブロックのそれぞれの前記参照画面上の位置の、前記ターゲットブロックの画面上の位置との位置ずれ量を、方向成分も含む参照ベクトルとし、当該参照ベクトルを所定の縮小率で縮小した参照縮小ベクトルを得る参照縮小ベクトル取得手段と、

前記参照縮小ベクトルに応じた大きさの前記参照ベクトルを前記位置ずれ量とする、前記所定の縮小率に応じて削減された数の複数個の前記参照ブロックのそれぞれについての差分絶対値和を記憶する縮小差分絶対値和テーブルを生成するテーブル生成手段と、

前記縮小差分絶対値和テーブルにおける前記差分絶対値和の最小値に対応する前記参照ベクトルを少なくとも用いて、前記参照画面と前記元画面との間の前記分割画像区間のそれぞれについての動きベクトルを算出する動きベクトル算出手段と、

を備え、

前記テーブル生成手段は、

前記参照縮小ベクトル取得手段で取得された前記参照縮小ベクトルの近傍値となる複数の前記参照ベクトルを検出する近傍参照ベクトル検出手段と、

前記差分絶対値和算出工程で算出された前記参照ブロックのそれぞれについての前記差分の絶対値の総和から、前記近傍参照ベクトル検出工程で検出された前記近傍の複数の前記参照ベクトルのそれぞれに対応する差分絶対値和のそれぞれを算出する分散差分絶対値和算出手段と、

前記分散差分絶対値和算出手段で算出した前記近傍の複数の前記参照ベクトルのそれぞれに対応する前記差分絶対値和を、それまでの前記近傍の複数の前記参照ベクトルのそれぞれに対応する前記差分絶対値和に加算する分散加算手段と、

を備えることを特徴とする画像信号の歪み補正装置。

【請求項２９】

撮影時の手ぶれによる撮像素子の位置的変化による撮像画像の歪を補正し、補正後の撮像画像の情報を記録するようにする撮像装置であって、

前記撮像画像の１画面領域が複数個に分割された各分割画像区間ごとに、２画面間の前記分割画像区間の撮像画像情報から動きベクトルを検出する動きベクトル検出手段と、

前記動きベクトル検出手段で検出された前記分割画像区間の前記動きベクトルのうちの、前記１画面内において隣接する分割画像区間のものの差分ベクトルを求める差分ベクトル検出手段と、

前記各分割画像区間内で、前記各分割画像区間について前記差分ベクトル検出手段で検出された前記差分ベクトルに基づいて、前記撮像画像の歪みを補正する歪み補正処理手段と、

前記歪み補正手段で補正された前記撮像画像の画像情報を記録媒体に記録する記録手段と、

を備えることを特徴とする撮像装置。

【発明の詳細な説明】

【技術分野】

【０００１】

この発明は、撮像装置での被写体画像の撮像時の手ぶれ等による撮像画像に発生する歪を補正する方法および当該歪を補正する装置に関し、特に、Ｘ－Ｙアドレス型の固体撮像素子、例えばＣＭＯＳ（Ｃｏｍｐｌｅｍｅｎｔａｒｙ Ｍｅｔａｌ Ｏｘｉｄｅ Ｓｅｍ

ｉｃｏｎｄｕｃｔｏｒ）型の固体撮像素子（以下、ＣＭＯＳイメージャという）を用いるビデオカメラ、デジタルスチルカメラなどの撮像装置や、撮像機能付きの記録再生装置に適用して好適なものである。

【背景技術】

【０００２】

電子式に撮像画像を撮像素子に記憶し、読み出すようにする電子式の撮像装置では、撮影者によるいわゆる手ぶれや、船の上など不安定な場所での撮影などの理由により、撮影時に、撮像素子が、物理的に、撮像画像の水平方向および／または垂直方向に比較的高速の位置的変化を生じると、その撮像画像に歪みを生じる。

【０００３】

しかし、電子式の撮像装置では、撮像素子から読み出した撮像画像データに対してデジタル処理（歪み補正処理）することにより、この手ぶれなどに起因する撮像画像の歪みを補正することが可能である。

【０００４】

従来のビデオカメラやデジタルスチルカメラなどの撮像装置や、撮像機能付の記録再生装置（カメラ付きの携帯電話端末やカメラ付きのパーソナルコンピュータなど）では、撮像素子として、一般に、ＣＣＤ（Ｃｈａｒｇｅ　Ｃｏｕｐｌｅｄ　Ｄｅｖｉｃｅ）を用いた固体撮像素子（以下、ＣＣＤを用いた固定撮像素子をＣＣＤイメージャという）が広く用いられている。

【０００５】

そこで、従来は、例えば特許文献１（特許第３３８４４５９号公報）などのように、撮像素子としてＣＣＤイメージャを用いることを前提とした手ぶれ補正技術が数多く提案されており、既に、当該手ぶれ補正技術を搭載した製品としても世の中に広く出回っている。

【０００６】

この従来の手ぶれ補正技術は、ＣＣＤイメージャの構造上、ＣＣＤイメージャの全画素に蓄積された光量のサンプリング時刻が、同一（１フレームに一回）である特性を利用している。

【０００７】

すなわち、ＣＣＤイメージャでは、全画素を同時期に露光し、1フレームの画像データについては、データ取り出しタイミングが全く同一であるので、手ぶれ変位量としては、図５８において矢印で示すように、１フレーム分の全画素について、１つの手ぶれ変位量Ｖｃｓを考えればよい。つまり、図５８において、本来、被写体は実線で示す領域ＦＬａに蓄積されるものであったのに、手ぶれにより点線で示す領域ＦＬｂに移動した場合には、当該フレームで１つの手ぶれ変位量Ｖｃｓを検出し、当該手ぶれ量Ｖｃｓ分だけ、読み出し画素位置（サンプリング画素位置）を補正することにより、手ぶれに起因する撮像画像歪を補正することができる。

【０００８】

なお、図５８に示すように、一般に、撮像素子では、その全画素を有効画素として扱うのではなく、全画素からなる領域（以下、実効画像領域という）ＡＦＬのうちの、周辺の領域を除く、水平有効領域および垂直有効領域で定まる中央部を有効画像領域ＥＦＬとして用いているものが多い。

【０００９】

このようなイメージャを用いた場合、手ぶれ補正により読み出し画素位置が変化しても、手ぶれ量が、実効画像領域ＡＦＬと有効画像領域ＥＦＬとの差分よりも小さい範囲では、イメージャが元々有している画素のデータを用いて歪み補正をすることができるので、補間処理などにより、手ぶれ補正に必要なデータを生成する場合に比べて、画像の劣化は少なくなる。

【００１０】

ところで、最近は、上述のような電子式の撮像装置の撮像素子として、画面の水平方向

（Ｘ方向）の位置と、垂直方向（Ｙ方向）位置とを指定することにより、画素単位で撮像データを読み出すことが可能なＸ－Ｙアドレス型の固体撮像素子、例えばＣＭＯＳ型の固体撮像素子（以下、ＣＭＯＳイメージャという）が用いられるようになっている。

【００１１】

このＣＭＯＳイメージャは、

（ａ）増幅型であり、増幅した信号を読み出すので、高感度である。

【００１２】

（ｂ）ＣＭＯＳ回路を用いるので低消費電力である。

【００１３】

（ｃ）低コストである。

【００１４】

（ｄ）原理的に、１画素単位でランダムにアクセスする（読み出す）ことが可能である。

【００１５】

などの特長を有する。

【００１６】

ＣＭＯＳイメージャは、１画素単位でも撮像画像データの読み出しは可能であるが、実用上は、１水平ライン分の画素群単位で読み出し（サンプリング）して、撮像画像データを出力するものが一般的である。

【００１７】

このように、ＣＭＯＳイメージャから撮像データを水平ライン単位で読み出す場合には、図５９に示すように、各水平ラインのついての露光期間は、水平ライン単位の読み出し時間差に応じたΔｔだけ、時間的なずれを生じる。なお、１画素単位で撮像画像データの読み出しがなされる場合であっても、ライン間の読み出し時間差に比べて、画素間の読み出し時間差は無視できるほど小さい。そこで、１画素単位で撮像画像データの読み出しがなされる場合であっても、同様な露光期間の時間的なずれを生じるものとすることができる。

【００１８】

このため、ＣＭＯＳイメージャを用いた撮像装置により、例えば走る電車の中から外の景色を撮影すると、本来は図６０（Ａ）のような画像が得られるべきものであるのに、図６０（Ｂ）に示すように、本来鉛直に立っている家や木が傾いた画像が出力撮像画像となってしまう。これがＣＭＯＳイメージャ特有のフォーカルプレーン現象である。

【００１９】

図６０（Ｂ）の画像例は、水平方向に移動しながら撮影したものであるため、被写体が傾いて撮像されるが、例えば垂直方向に移動しながら撮影した場合には、図示は省略するが、被写体が垂直方向に縮んだり、伸びたりするような画像となる。

【００２０】

この現象が発現するのは、ＣＭＯＳイメージャを使用する撮像装置を持つ撮影者が高速に移動しながら撮影したり、逆に、固定位置にいる撮影者が高速に移動する被写体を撮影したりした場合であり、撮像装置と被写体の相対速度が速い場合に顕著となる。ただし、一般的な撮影において、このような状況は稀であるといえる。

【００２１】

しかしながら、撮影者が撮像装置を手で持って撮影している場合に、撮像装置を持っている撮影者の手が、微小かつ高速に振動した場合、すなわち、手ぶれが発生した場合、上述したフォーカルプレーン現象が発現することになる。

【００２２】

なぜなら、ＣＭＯＳイメージャの場合の手ぶれ（以下、ＣＭＯＳ手ぶれという）の値は、ＣＣＤイメージャのように１フレーム内で１個の値ではなく、前述したように、１フレーム内における画素やライン毎のサンプリング時刻の違いにより、画素や水平ラインごとに異なるものとなる。このため、ＣＭＯＳイメージャを用いた撮像装置においては、単に、１フレーム単位での手ぶれ量を用いた補正を行っても、前述したフォーカルプレーン現

象による歪みは、補正されずに残ってしまう。
【００２３】
この際、手ぶれの方向、大きさ、速度が、１フレーム内（１枚の撮像画像内）で一律でないことが原因となって、ＣＭＯＳイメージャを用いた撮像装置で、ＣＭＯＳ手ぶれが生じた時の被写体の撮像画像出力は、図６０（Ｃ）に示すように、グニャグニャした奇妙な画像歪みが生じたものとなる。
【００２４】
ところで、デジタルスチルカメラのように静止画を撮影する装置においては、元々静止画撮影の前提として手ぶれ量が限られていることと、メカシャッターと併用することで、ＣＭＯＳ手ぶれに起因するフォーカルプレーン現象を抑えることが、比較的容易である。
【００２５】
一方、ビデオカメラのような動画撮影を前提とした撮像装置であっては、業務用や高級機種では、1フレームにおける最大サンプリング時間差（一番上のラインと一番下のラインのサンプリング時刻の差）を短くするため、超高速の読み出しを行なうことで、実質的にＣＭＯＳ手ぶれに伴うフォーカルプレーンの発生を防ぐ方法が採られていることがある。
【００２６】
また、撮像画像に対する相対的な手ぶれ量の大きさは、光学ズームの倍率が大きくなるのに伴って増大するため、動画撮影用途であっても、光学ズームを持たないか、倍率の小さい機種であれば、ＣＭＯＳ手ぶれは大きな問題となっていない。そもそも、従来のＣＣＤイメージャで行っているような、加速度センサを用いた手ぶれ補正すら搭載していない安価な撮像装置では、ＣＭＯＳ手ぶれの悪影響は相対的に小さくなり、問題にすらなっていない場合が多い。
【００２７】
上記の特許文献は、次の通りである。
【特許文献１】特許第３３８４４５９号公報
【発明の開示】
【発明が解決しようとする課題】
【００２８】
上述の状況を鑑みると、静止画撮影はできるが動画撮影を主用途とし、高倍率の光学ズームを搭載した撮像装置においては、メカシャッターと併用するなどの特殊な装置や、高速なサンプリングクロックを利用することが必要になる。しかし、そのような構成とした場合には、非常に高精度でコスト高となる問題がある。
【００２９】
また、従来、手ぶれを検出する方法としては、ジャイロ（角速度）センサなどの機械的な部品を用いる方法が一般的であるが、ジャイロセンサを用いた場合には、撮像装置の小型、軽量化、低コスト化を実現する場合の支障となるという問題がある。
【００３０】
また、従来は、ジャイロセンサを用いた手ブレ補正の弱点である、ジャイロセンサ自体の検出精度の甘さが、これまでの主要な応用先である動画撮影においては、問題にならなかったが、ここ数年においては、デジタルスチルカメラの急速な普及と、それと機を同じくした急速な高画素化の流れが、新たな問題を生み始めている。それは、低照度（露光時間が長い）ときの静止画においても、手ぶれ補正が強く求められているものの、解がジャイロセンサ等のセンサを用いたものしか存在せず、先に挙げたジャイロセンサの弱点やその他の問題が露呈しつつある点である。
【００３１】
現在市場に出回っている民生機における、静止画用途の手ぶれ補正は、全て、遍くジャイロセンサもしくは加速度センサを使って手ぶれベクトルを計測し、それを機構系にフィードバックして、ＣＣＤ（Ｃｈａｒｇｅ Ｃｏｕｐｌｅｄ Ｄｅｖｉｃｅ）イメージャやＣＭＯＳ（Ｃｏｍｐｌｅｍｅｎｔａｒｙ Ｍｅｔａｌ Ｏｘｉｄｅ Ｓｅｍｉｃｏｎｄｕ

ｃｔｏｒ）イメージャなどのイメージセンサに射影される像が、ぶれを起こさないように高速に制御する、というものである。

【００３２】

ここでいう機構系としては、レンズ、プリズム、イメージャ（もしくはイメージャと一体化したモジュール）が提案されており、それぞれ、レンズシフト、プリズムシフト、イメージャシフトと呼ばれている。

【００３３】

このような方法で手ぶれ補正がなされている以上、先に挙げたジャイロセンサ自体の精度誤差に加え、機構系へのフィードバック遅延、もしくはフィードバック遅延を回避するための予測誤差、そして、機構系の制御誤差も重畳され、とてもピクセル精度で補正をかけることは不可能である。

【００３４】

以上に挙げたように、現状のセンサを使用した手ぶれ補正には、原理的に、精度を追求できない、という大きな問題があるにも関わらず、市場で高評価を得ているのは、手ぶれを補正できないまでも低減できるからである。

【００３５】

しかしながら、今後益々の高画素化が予想される中、ピクセルサイズが小さくなるに従って、補正限界がピクセル精度と益々開いて行かざるを得ない、という事実に市場が気付くのも時間の問題である。

【００３６】

この発明は、以上の点にかんがみ、上述のような問題を回避しつつ、ＣＭＯＳイメージャなどのＸ－Ｙアドレス型の撮像素子を用いた撮像装置においても、手ぶれ等が原因で生じるフォーカルプレーン現象による撮像画像の歪を軽減することができる方法および装置を、ジャイロセンサなどの機構系部品を使用することなく、デジタル信号処理により高精度かつ低コストに実現することを目的とする。

【課題を解決するための手段】

【００３７】

上記の課題を解決するために、請求項１の発明は、

１画面の画像の水平方向および／または垂直方向の歪みを補正する方法であって、

前記画像の１画面区間を複数個の分割画像区間に分割し、

前記分割画像区間のそれぞれにおける画像の動きベクトルを、２画面間の前記分割画像区間の画像情報から検出し、

検出された前記複数個の分割画像区間の前記動きベクトルのうちの、前記１画面内において隣接する分割画像区間のものの差分ベクトルを求め、求められた前記差分ベクトルに基づいて、前記分割画像区間のそれぞれ内における画像の歪みを補正する

ことを特徴とする画像信号の歪み補正方法を提供する。

【００３８】

上述の構成の請求項１の発明によれば、１画面区間分が複数に分割されたそれぞれの分割画像区間で、ジャイロセンサなどの機構部品を用いることなく、当該分割画像区間の画像情報が用いられて、当該分割画像区間における画像の動きベクトルが検出される。

【００３９】

そして、検出された動きベクトルの、１画面内において隣接する分割画像区間のものの差分ベクトルを求めることにより、分割画像区間のそれぞれごとの画像の歪み速度に対応する動き速度ベクトルが検出される。そして、当該分割画像区間について検出された差分ベクトルに基づいて、分割画像区間のそれぞれごとの画像の歪みが補正され、例えばフォーカルプレーン現象を含む画像の歪みを補正することができる。

【００４０】

また、請求項２の発明は、請求項１に記載の画像信号の歪み補正方法において、

前記差分ベクトルを求めることにより、前記分割画像区間のそれぞれごとに、当該分割画像区間の画像の歪み速度を検出し、

前記各分割画像区間内においては、当該各分割画像区間について検出された前記画像の歪み速度の時間積分値を前記画像の歪み補正対象部位の歪み変位量とし、前記時間積分値を用いて前記画像の歪みを補正する

ことを特徴とする画像信号の歪み補正方法を提供する。

【００４１】

上述の構成の請求項２の発明によれば、分割画像区間のそれぞれ内においては、当該分割画像区間について検出された画像の歪み速度（動き速度ベクトル）の時間積分値を画像の歪み補正対象部位の変位量として求め、当該求めた時間積分値を用いて前記画像の歪みを補正するので、少ない処理遅延で、例えばフォーカルプレーン現象を含む画像の歪みを補正することができると共に、時間積分値を用いるので、画像の歪み速度が変わる他の分割画像区間の画像との境界においても、画像にずれを生じることはない。

【００４２】

また、請求項１４の発明は、

注目画面の画像の１画面区間を複数個の分割画像区間に分割し、

前記分割画像区間のそれぞれにおける画像の動きベクトルを、前記注目画面と当該注目画面よりも前の元画面との間の前記分割画像区間の画像情報から検出し、

検出された前記複数個の分割画像区間の前記動きベクトルのうちの、前記１画面内において隣接する分割画像区間のものの差分ベクトルを求め、求められた前記差分ベクトルに基づいて、前記注目画面の前記分割画像区間のそれぞれ内における水平方向および／または垂直方向の画像の歪みを補正する方法において、

前記分割画像区間のそれぞれにおける画像の動きベクトルを検出する方法は、

前記元画面中の前記各分割画像区間において、所定の位置に複数の画素からなる所定の大きさの少なくとも１個のターゲットブロックを設定し、

前記ターゲットブロックと同じ大きさの参照ブロックを、前記注目画面に設定されたサーチ範囲において複数個設定し、

前記複数個の参照ブロックの内から、前記ターゲットブロックと相関の強い前記参照ブロックを検出し、

当該検出した参照ブロックの前記ターゲットブロックに対する画面上の位置ずれ量に基づいて、前記動きベクトルを検出する

ものであって、

前記参照ブロックのそれぞれにおいて、当該参照ブロック内の各画素の画素値と、前記ターゲットブロック内で対応する位置の各画素の画素値との差分の絶対値の総和を求める差分絶対値和算出工程と、

前記参照ブロックのそれぞれの前記参照画面上の位置の、前記ターゲットブロックの画面上の位置との位置ずれ量を、方向成分も含む参照ベクトルとし、当該参照ベクトルを所定の縮小率で縮小した参照縮小ベクトルを得る参照縮小ベクトル取得工程と、

前記参照縮小ベクトルに応じた大きさの前記参照ベクトルを前記位置ずれ量とする、前記所定の縮小率に応じて削減された数の複数個の前記参照ブロックのそれぞれについての差分絶対値和を記憶する縮小差分絶対値和テーブルを生成するテーブル生成工程と、

前記縮小差分絶対値和テーブルにおける前記差分絶対値和の最小値に対応する前記参照ベクトルを少なくとも用いて、前記参照画面と前記元画面との間の前記分割画像区間のそれぞれについての動きベクトルを算出する動きベクトル算出工程と、

を備え、

前記テーブル生成工程は、

前記参照縮小ベクトル取得工程で取得された前記参照縮小ベクトルの近傍値となる複数の前記参照ベクトルを検出する近傍参照ベクトル検出工程と、

前記差分絶対値和算出工程で算出された前記参照ブロックのそれぞれについての前記差分の絶対値の総和から、前記近傍参照ベクトル検出工程で検出された前記近傍の複数の前記参照ベクトルのそれぞれに対応する差分絶対値和のそれぞれを算出する分散差分絶対値和算出工程と、

前記分散差分絶対値和算出工程で算出された前記近傍の複数の前記参照ベクトルのそれぞれに対応する前記差分絶対値和を、それまでの前記近傍の複数の前記参照ベクトルのそれぞれに対応する前記差分絶対値和に加算する分散加算工程と、

を備えることを特徴とする画像信号の歪み補正方法を提供する。

【００４３】

上述の構成の請求項１４の発明による画像信号の歪み補正方法における動きベクトルの検出方法においては、注目画面のサーチ範囲内の参照ブロックのそれぞれにおいて、当該参照ブロック内の各画素の画素値と、前記ターゲットブロック内で対応する位置の各画素の画素値との差分の絶対値の総和（この差分の絶対値の総和を差分絶対値和と呼ぶ。以下、この差分絶対値和をＳＡＤ（Ｓｕｍ　ｏｆ　Ａｂｓｏｌｕｔｅ　Ｄｉｆｆｅｒｅｎｃｅ）と記載することとする）を求める。

【００４４】

この請求項１４の発明においては、求めた参照ブロックのＳＡＤの値（ＳＡＤ値）を、その参照ブロックの参照ベクトルに対応して記憶するのではなく、その参照ベクトルを縮小した参照縮小ベクトルに対応して記憶する。

【００４５】

この場合に、参照縮小ベクトルは、参照ブロックのそれぞれが対応する参照ベクトルとは一致していないので、テーブル生成工程では、先ず、参照縮小ベクトルの近傍の複数個の参照ベクトルを検出する。そして、それら近傍の複数の参照ベクトルのそれぞれに対応するＳＡＤ値を、ＳＡＤ算出工程で算出したＳＡＤ値から算出する。

【００４６】

そして、算出した近傍の複数の参照ベクトルのそれぞれに対応するＳＡＤ値を、それまでの当該近傍の複数の参照ベクトルのそれぞれに対応するＳＡＤ値に加算する。

【００４７】

こうして得られたＳＡＤテーブルは、参照縮小ベクトルのそれぞれの近傍の複数個の参照ベクトルのみについてのＳＡＤ値のテーブルとなっており、参照ベクトルの縮小率に応じて縮小されたものとなっている。

【００４８】

つまり、換言すると、生成されたＳＡＤテーブルは、参照ベクトルの縮小率に応じて縮小されたフレーム画像についてのＳＡＤテーブルに対応する。この場合において、ターゲットブロックおよび参照ブロックの大きさは縮小されていないので、生成されたＳＡＤテーブルサイズは、小さくなる。

【００４９】

この縮小されたＳＡＤテーブルにおいて、ＳＡＤ値が最小値となっている参照ベクトルを検出し、その検出した参照ベクトルから求める動きベクトルを算出する。例えば、求めたＳＡＤ値が最小値となっている参照ベクトルを、参照ベクトルの縮小率に応じて拡大して、動きベクトルを算出することができる。

【００５０】

そして、この請求項１４の発明においては、画像を縮小する処理は不要であるので、画像の縮小変換に伴う処理時間の増大やメモリ容量の消費は無く、また、ＳＡＤ値は、元のフレームの全ての画素を用いて求め、当該求めたＳＡＤ値を参照縮小ベクトルの近傍の複数個の参照ベクトルに対応して分散させながら求めるものであるので、ＳＡＤ値を分散する時点で自ずと縮小倍率に応じた適切なフィルタ演算をしていることと等価であり、画像を縮小した場合のようなローパスフィルタの実装は、不要となるものである。

【発明の効果】

【００５１】

この発明によれば、ＣＭＯＳイメージャなどのＸ－Ｙアドレス型の撮像素子を用いた撮像装置において、ジャイロセンサなどの機構系部品を使用することなく、デジタル信号処理により高精度かつ低コストで、手ぶれ等が原因で生じるフォーカルプレーン現象による撮像画像の歪を軽減することができる。

【発明を実施するための最良の形態】
【００５２】
以下、この発明による画像信号の歪み補正装置および方法の実施の形態を、ＣＭＯＳイメージャを撮像素子に用いた撮像装置に適用した場合を例に、図を参照しながら説明する。
【００５３】
［実施の形態におけるＣＭＯＳ手ぶれに起因するフォーカルプレーン現象の歪み補正について］
ＣＭＯＳ手ぶれに起因するフォーカルプレーン現象の歪み補正および抑制技術としては、先に特許文献２（特開２００４－２６６３２２公報）に記載されたものがある。
【００５４】
この特許文献２に記載される補正方法は、１水平ライン単位で撮像画像データがＣＭＯＳイメージャから読み出されるとき、各水平ラインについての手ぶれによる位置変位量（以下、手ぶれ量という）を検出し、当該検出した手ぶれ量分だけ、発生した手ぶれとは逆方向にずれた位置から当該水平ラインのデータを読み出すように補正するものである。
【００５５】
しかし、手ぶれ量を各水平ラインについて得ることは、手ぶれを検出するセンサのサンプリング周波数などの条件から困難であることに鑑み、この特許文献２においては、図６１（Ａ）に示すように、画面の垂直方向の複数ライン分置きの位置での手ぶれ量を、離散的に検出するようにしている。図６１では、５０ライン置きのライン位置での手ぶれ量Ｑ１，Ｑ２，Ｑ３・・・（図６１（Ｂ）参照；なお、図では水平方向の手ぶれ量のみを示している）を検出している。
【００５６】
そして、手ぶれ検出を行なうライン以外の、手ぶれ量を直接的には検出しない４９ラインに対する手ぶれ量は、検出された手ぶれ量Ｑ１，Ｑ２，Ｑ３・・・を用いて、補間により求めるようにする。この補間の方法としては、例えば図６１（Ｃ）に例を示すように、幾通りかの方法がある。図６１（Ｃ）の補間の方法は、基本的には、手ぶれ量を直接的には検出しない４９ラインの直前ラインおよび直後ラインの２位置での２つの手ぶれ量ＱｎおよびＱｎ＋１（ｎは１以上の整数）を用いて行なう。
【００５７】
例えば、補間（１）の方法では、直前ラインで検出された手ぶれ量Ｑｎを、前記４９ラインのうちの前半のラインではそのまま用い、前記４９ラインのうちの後半のラインでは、直後ラインで検出された手ぶれ量Ｑｎ＋１を用いるようにする。また、補間（２）の方法は、直前ラインで検出された手ぶれ量Ｑｎと直後ラインで検出された手ぶれ量Ｑｎ＋１とを直線で結んだときの各ラインでの値を、当該ラインの手ぶれ量とする、すなわち、いわゆる平均値補間を行なう方法である。
【００５８】
この特許文献２の方法によれば、手ぶれが原因で生じるＣＭＯＳイメージャにおけるフォーカルプレーン現象を含む歪を補正することが可能である。
【００５９】
しかしながら、この特許文献２で採用されている方法の場合には、離散的なサンプリング位置で手ぶれを検出し、その検出結果から前記離散的なサンプリング位置での手ぶれ量を求め、当該離散的なサンプリング位置での手ぶれ量を用いて補間をすることにより、各ライン位置での手ぶれ量を推定するようにしている。
【００６０】
このため、特許文献２の場合には、前記直前ラインの手ぶれ量Ｑｎと、前記直後ラインの手ぶれ量Ｑｎ＋１との両方が得られた後でないと、前記直前ライン後の各ラインの手ぶれ量を、補間処理により得るようにすることができない。このため、前記直前ラインの後の各ラインについての手ぶれ補正処理が、最大、離散的なサンプリング周期区間である複数ライン区間分だけ遅延することになる。

【００６１】
また、上述の補間（１）の方法を用いた場合には、補間値としての手ぶれ量の変化時間で、画像位置がずれる可能性がある。
【００６２】
また、補間（２）の方法の場合には、直前ラインの手ぶれ量Ｑｎと直後ラインの手ぶれ量Ｑｎ＋１とから、手ぶれ量の変化の傾きを求め、その傾きに、直前ラインからの対象ラインまでのライン数を乗算することにより、当該対象ラインの手ぶれ量を求めるようにしなければならない。このため、乗算器が必要になると共に、乗算パラメータを別途設定するためのレジスタを設ける必要があり、ハードウエアが複雑化すると共に、回路規模が大きくなる。さらに、乗算誤差のため、手ぶれ量のサンプリング値が変わる境界位置において、図６１（Ｃ）の補間（２）の場合の図のように連続的にはならずに、画像位置がずれる可能性がある。
【００６３】
以下に説明する実施の形態では、上述のような問題を回避しつつ、ＣＭＯＳイメージャなどのＸ－Ｙアドレス型の撮像素子を用いた撮像装置においても、手ぶれ等が原因で生じるフォーカルプレーン現象による撮像画像の歪を軽減することができるようにしている。
【００６４】
先ず、図１を用いて、この発明による画像信号の歪み補正方法の実施形態におけるフォーカルプレーン現象による撮像画像の歪を軽減する方法の概要を説明する。以下に説明する実施形態は、撮像素子としてＸ－Ｙアドレス型の固体撮像素子の代表例であるＣＭＯＳイメージャを用いて撮像した撮像画像について、前述したＣＭＯＳ手ぶれを補正する場合の例である。なお、この発明は、撮像画像が動画像である場合と静止画像である場合の、何れの場合にも適用できるものである。
【００６５】
図１の概要説明図においては、図１（Ａ）に示すように、画像に生じるＣＭＯＳ手ぶれによる歪みを判りやすく示すために、歪みの無いオリジナルの画像として多数個の長方形からなる格子縞模様の画像を想定している。したがって、ＣＭＯＳ手ぶれによる画像歪みは、当該格子縞模様を構成する長方形の変形歪みとして現れるものとなる。
【００６６】
なお、この実施形態においても、ＣＭＯＳイメージャは、図５８に示したような１画面とする水平有効領域および垂直有効領域で定まる有効画像領域ＥＦＬ（１垂直周期分）よりも広い範囲の実効画像領域ＡＦＬを有するものが用いられるものである。なお、図１（Ａ）～（Ｄ）に示した１画面分の画像は、図５８の有効画像領域ＥＦＬの大きさの画像である。
【００６７】
この実施形態では、水平方向の画素クロックの速度は、手ぶれの速度よりも十分に早いとし、また、撮像画像データの読み出しは、１水平ライン単位で順次に行なうものとして、手ぶれ補正は、１水平ライン単位で行なうようにする。
【００６８】
また、この実施形態では、図１（Ｂ），（Ｃ），（Ｄ）に示すように、ＣＭＯＳイメージャの有効画像領域ＥＦＬ分の画像区間は、垂直方向に複数に等分割し、各分割画像区間Ｐｄｉｖでは、同じ手ぶれ速度の手ぶれ変位を受けるものと仮定している。ここで、分割数は、手ぶれ補正したときに、必要十分な補正効果が得られる分割画像区間Ｐｄｉｖの大きさを考慮して定められる。以下に説明する例では、有効画像領域ＥＦＬの画像区間を垂直方向に８分割するようにする。すなわち、この例では、分割画像区間Ｐｄｉｖは、垂直同期信号の１周期を１／８に分割した区間となる。
【００６９】
１つの分割画像区間Ｐｄｉｖは、複数水平ライン分からなるが、この実施形態では、同じ分割画像区間Ｐｄｉｖに含まれる複数水平ラインは同じ手ぶれ速度の手ぶれ変位を受けるものと仮定して手ぶれ補正処理を行なうものである。

【００７０】
そして、この実施形態では、８個の分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７のそれぞれの先頭の水平ラインに対する手ぶれ速度ベクトルを、図１（Ｂ）の左側の矢印に示すようなものとして、手ぶれ速度検出手段により検出する。すなわち、この例では、手ぶれ速度ベクトルは、１画面分の撮像画像について、８箇所において、離散的に検出するようにする。
【００７１】
図１（Ｂ）の画像は、手ぶれ速度ベクトルが、各分割画像区間Ｐｄｉｖで、左側の矢印のようなものとした時に生じる手ぶれによる撮像画像の歪みを示している。この実施形態では、この手ぶれによる撮像画像の歪みを、水平方向処理と垂直方向処理とに分けて、それぞれの方向の歪みを別々に補正するようにする。
【００７２】
補正処理の詳細については後述するが、この実施形態では、先ず、水平方向の歪みについて補正処理を行ない、次いで、垂直方向の歪みについての補正処理を行なう。この場合に、１画面分の撮像画像のすべてのデータについての水平方向の歪み補正処理が完了する前であっても、垂直方向の歪み補正処理が可能になった段階で、垂直方向の歪み補正処理を開始して、水平方向の歪み補正処理と並行して行なうようにすることにより、効率良く歪み補正処理を行なうようにする。
【００７３】
そこで、この実施形態では、先ず、手ぶれによる水平方向の画像歪みを補正するために、図１（Ｃ）の左側の矢印および図２の左側の矢印で示すように、各分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７について検出した手ぶれ速度ベクトルＶｅｃ（Ｖｅｃ＿０～Ｖｅｃ＿７）の水平方向成分の逆符号成分（以下、この手ぶれ速度ベクトルの水平方向成分の逆符号成分を、水平補正速度成分という）Ｘ＿ＳＴＢ（Ｘ＿ＳＴＢ＿０～Ｘ＿ＳＴＢ＿７）を求める。
【００７４】
そして、各分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７内においては、各水平ラインのそれぞれについての水平方向の手ぶれ量を補正するための補正量として、上述のようにして求めた水平補正速度成分Ｘ＿ＳＴＢ＿０～Ｘ＿ＳＴＢ＿７を、それぞれの水平ラインについて時間積分（各分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７の先頭の水平ライン時点を基準点とした時間で積分）して算出する。図２の右側に、その積分結果の値としての水平方向の手ぶれ補正量（以下、水平手ぶれ補正量という）ＳＸ＿ＡＤＤを示す。
【００７５】
この図から判るように、各水平ラインのそれぞれについての水平手ぶれ補正量ＳＸ＿ＡＤＤは、各分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７においては、一定の傾きで変化し、かつ、積分であるので、分割画像区間の境目では折れ線とはなるが、連続的なものとなり、ずれは生じない。
【００７６】
水平方向の歪み補正処理においては、以上のようにして求められた各ラインについての手ぶれ水平補正量ＳＸ＿ＡＤＤを用いて、ＣＭＯＳイメージャ上での各水平ラインの水平方向の読み出し開始位置を補正することにより、撮像画像を補正処理する。すなわち、各水平ラインの読み出し開始位置を、図１（Ｂ）における手ぶれに応じて水平方向にずれた位置からとすることにより、図１（Ｃ）の画像に示すように、水平方向の歪みが補正されることになる。
【００７７】
この図１（Ｃ）の画像は、図１（Ｂ）の画像歪みの内の水平方向の歪みは補正されて除去されているが、垂直方向の歪みが残っている。
【００７８】
そこで、この実施形態では、手ぶれによる垂直方向の画像歪みを補正するため、図１（Ｄ）の左側の矢印に示すように、各分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７について検

出した手ぶれ速度ベクトルの垂直方向成分の逆符号成分（以下、この手ぶれ速度ベクトルの垂直方向成分の逆符号成分を、垂直補正速度成分という）Ｙ＿ＳＴＢ＿０～Ｙ＿ＳＴＢ＿７を求める。

【００７９】

そして、各分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７内においては、各水平ラインのそれぞれについての垂直方向の手ぶれ量を補正するための補正量として、上述のようにして求めた垂直補正速度成分Ｙ＿ＳＴＢ＿０～Ｙ＿ＳＴＢ＿７を、それぞれの水平ラインについて時間積分（各分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７の先頭の水平ライン時点を基準点とした時間で積分）して算出する。

【００８０】

この積分結果としての垂直方向の手ぶれ補正量（以下、垂直手ぶれ補正量という）ＳＹ＿ＡＤＤは、ここでは、図示を省略するが、各分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７においては、一定の傾きで変化し、かつ、積分であるので、分割画像区間の境目では折れ線とはなるが、連続的なものとなり、ずれは生じない。

【００８１】

垂直方向の歪み補正処理においては、以上のようにして求められた各ラインについての垂直手ぶれ補正量ＳＹ＿ＡＤＤを用いて、ＣＭＯＳイメージャ上での各水平ラインの垂直方向の読み出し開始位置を補正することにより、撮像画像を補正処理する。これにより、図１（Ｄ）の画像に示すように、水平方向の場合と同様にして垂直方向の歪みが補正され、歪みの除去された撮像画像データ出力が得られる。

【００８２】

水平手ぶれ補正量および垂直手ぶれ補正量を用いて、手ぶれによるＣＭＯＳイメージャについてのフォーカルプレーン現象による画像歪みが補正される様子は、前述した特許文献２と同様である。

【００８３】

しかし、特許文献２では、手ぶれ量を、垂直方向の離散的な水平ラインについて検出すると共に、直接的に手ぶれ量を検出しない、離散的な水平ラインの間の各水平ラインについては、前記離散的に検出した手ぶれ量を用いた補間処理により得るようにしているのに対して、この実施形態では、垂直方向の離散的な水平ラインについては手ぶれ速度を検出し、当該検出した手ぶれ速度を、各水平ラインについて順次に積分してゆくことにより、各水平ラインの手ぶれ量（手ぶれ補正量）を検出する点が大きく異なる。

【００８４】

このように、この実施形態では、各水平ラインの手ぶれ量（手ぶれ補正量）は、手ぶれ速度を積分することにより算出するものであるので、分割画像区間の境目では折れ線とはなるが、連続的なものとなり、ずれは生じない。

【００８５】

なお、この実施形態では、手ぶれ速度ベクトルから水平成分および垂直成分を抽出するときに、その逆符号成分を算出して、補正速度成分とするようにしたが、速度成分あるいは手ぶれ量を逆符号として、補正速度成分あるいは手ぶれ補正量とするのは、撮像画像データに対する手ぶれ補正処理の実行開始前であれば、どの時点であっても良い。

【００８６】

［手ぶれ動きベクトルおよび手ぶれ速度ベクトルの算出］

この実施の形態では、撮像画像の手ぶれ動きベクトルを検出する方法として、２画面間の相関を求めるブロックマッチングを用いる。このブロックマッチングを用いる方法は、ジャイロ（角速度）センサなどの機械的な部品が不要なので、撮像装置の小型、軽量化を実現することができるという点で有利である。

【００８７】

図３および図４は、ブロックマッチングの概要を図示したものである。また、図５には、その処理フローチャートの一般例を示す。

【００８８】

ブロックマッチングは、撮像装置部からの撮像画像について、注目画面である参照画面と、当該参照画面よりも１画面分前の撮像画面である元画面との間の１画面分単位での動きベクトルを、所定の大きさの矩形領域のブロックについて、参照画面と元画面との相関を算出することにより算出する方法である。

【００８９】

なお、ここで画面とは、１フレームまたは１フィールドの画像データからなる画像を意味しているが、この明細書では、説明の便宜上、画面は１フレームからなるものとして、画面をフレームと称することとする。したがって、参照画面は参照フレーム、元画面は元フレームと称する。

【００９０】

例えば、参照フレームの画像データは、撮像装置部からの現フレームの画像データ、または現フレームの画像データがフレームメモリに格納されて１フレーム分遅延されたものとされる。元フレームの画像データは、参照フレームの画像データがさらにフレームメモリに格納されて１フレーム分遅延されたものとされる。

【００９１】

図３および図４は、従来のブロックマッチングの概要を説明するための図である。また、図５は、従来のブロックマッチング処理のフローチャートの一例である。

【００９２】

ブロックマッチングにおいては、図３に示すように、元フレーム１０１の任意の所定の位置において、水平方向の複数画素および垂直方向の複数ライン分からなる所定の大きさの矩形領域からなるターゲットブロック１０３が設定される。

【００９３】

これに対して、参照フレーム１０２において、元フレームのターゲットブロック１０３の位置と同じ位置に、ターゲットブロックの射影イメージブロック１０４（図３の点線参照）を想定し、このターゲットブロックの射影イメージブロック１０４を中心としたサーチ範囲１０５（図３の一点鎖線参照）を設定すると共に、ターゲットブロック１０３と同じ大きさの参照ブロック１０６を考える。

【００９４】

そして、この参照ブロック１０６の位置を参照フレーム１０２のサーチ範囲１０５内において移動させ、各位置において参照ブロック１０６に含まれる画像内容と、ターゲットブロック１０３の画像内容との相関を求め、最も相関が強いとして検出された参照ブロック１０６の位置を、元フレームのターゲットブロック１０３が、参照フレーム１０２において移動した位置として検出するようにする。そして、その検出した参照フレーム１０６の位置と、ターゲットブロックの位置との間の位置ずれ量を、方向成分を含む量としての動きベクトルとして検出するようにする。

【００９５】

この場合、参照ブロック１０６は、サーチ範囲１０５を、例えば水平方向および垂直方向に、１画素または複数画素単位で移動させるようにする。したがって、サーチ範囲１０５内には、複数個の参照ブロックが設定されることになる。

【００９６】

ここで、ターゲットブロック１０３と、サーチ範囲１０５内を移動する各参照ブロック１０６との相関は、ターゲットブロック１０３内の各画素の輝度値と、参照ブロック１０６内の対応する各画素の輝度値との差分の絶対値の、ブロック内の全画素についての総和（この差分の絶対値の総和を差分絶対値和と呼ぶ。以下、この差分絶対値和をＳＡＤ（Ｓｕｍ　ｏｆ　Ａｂｓｏｌｕｔｅ　Ｄｉｆｆｅｒｅｎｃｅ）と記載することとする）を求めることにより検出する。すなわち、ＳＡＤ値が最小となる位置の参照ブロック１０６が最も相関が強い参照ブロックとして検出され、その検出された参照ブロック１０６のターゲットブロック１０３の位置に対する位置ずれ量が動きベクトルとして検出される。

【００９７】

ブロックマッチングでは、サーチ範囲１０５内に設定される複数個の参照ブロック１０

６のそれぞれの、ターゲットブロック１０３の位置に対する位置ずれ量は、方向成分を含む量としての参照ベクトル１０７（図３参照）で表現される。各参照ブロック１０６の参照ベクトル１０７は、参照ブロック１０６の参照フレーム１０２上の位置に応じた値となるが、従来のブロックマッチングでは、ＳＡＤ値が最小値となる参照ブロック１０６の参照ベクトルを、動きベクトルとして検出するものである。

【００９８】

そこで、ブロックマッチングでは、一般に、図４に示すように、サーチ範囲１０５内において設定される複数個の参照ブロック１０６のそれぞれとターゲットブロック１０３との間におけるＳＡＤ値（以下、説明の簡単のため参照ブロックについてのＳＡＤ値という）を、それぞれの参照ブロック１０６の位置に応じた参照ベクトル１０７のそれぞれに対応させて、メモリに記憶しておき、そのメモリに記憶された全ての参照ブロック１０６についてのＳＡＤ値の中から、最小のＳＡＤ値の参照ブロック１０６を検出することで、動きベクトル１１０を検出するようにしている。

【００９９】

サーチ範囲１０５内に設定された複数個の参照ブロック１０６の位置に応じた参照ベクトル１０７のそれぞれに対応させて、それぞれの参照ブロック１０６についてのＳＡＤ値を記憶したものを、差分絶対値和テーブル（以下ＳＡＤテーブルという）と呼ぶ。図４のＳＡＤテーブル１０８が、これを示しており、このＳＡＤテーブル１０８において、それぞれの参照ブロック１０６についてのＳＡＤ値をＳＡＤテーブル要素１０９という。

【０１００】

なお、上述の説明において、ターゲットブロック１０３および参照ブロック１０６の位置とは、それらのブロックの任意の特定の位置、例えば中心位置を意味するものであり、参照ベクトル１０７は、参照フレーム１０２におけるターゲットブロック１０３の射影イメージブロック１０４の位置と、参照ブロック１０６の位置との間のずれ量（方向を含む）を示すものである。図３および図４の例では、ターゲットブロック１０３は、フレームの中心位置にあるとしている場合である。

【０１０１】

そして、各参照ブロック１０６に対応する参照ベクトル１０７は、参照フレーム１０２上のターゲットブロック１０３に対応する位置に対する当該各参照ブロック１０６の位置ずれとなっているので、参照ブロック１０６の位置が特定されると、その位置に対応して参照ベクトルの値も特定される。したがって、ＳＡＤテーブル１０８のメモリにおける参照ブロックのＳＡＤテーブル要素のアドレスが特定されると、対応する参照ベクトルは特定されることになる。

【０１０２】

以上説明した従来のブロックマッチングの処理を、図５のフローチャートを参照して説明すると、次のようになる。

【０１０３】

先ず、サーチ範囲１０５内の１つの参照ブロックＩｉを指定するが、これは、当該参照ブロックＩｉに対応する参照ベクトルを指定することに等しい（ステップＳ１）。ここで、図５において、（ｖｘ，ｖｙ）は、ターゲットブロックのフレーム上の位置を基準位置（０，０）としたときに、指定された参照ベクトルにより示される位置を示し、ｖｘは指定された参照ベクトルによる、基準位置からの水平方向のずれ量成分であり、また、ｖｙは指定された参照ベクトルによる、基準位置からの垂直方向成分のずれ量成分である。

【０１０４】

ここでは、ずれ量ｖｘ、ｖｙは、画素を単位とした値とされ、例えばｖｘ＝＋１は、基準位置（０，０）に対して、水平方向の右方向に１画素分ずれた位置を示し、また、ｖｘ＝－１は、基準位置（０，０）に対して、水平方向の左方向に１画素分ずれた位置を示している。また、例えばｖｙ＝＋１は、基準位置（０，０）に対して、垂直方向の下方向に１画素分ずれた位置を示し、また、ｖｙ＝－１は、基準位置（０，０）に対して、垂直方向の上方向に１画素分ずれた位置を示している。

【０１０５】
以上のように、（ｖｘ、ｖｙ）は、参照ベクトルで示される基準位置に対する位置（以下、簡単のため、参照ベクトルで示される位置という）を示しており、参照ベクトルのそれぞれに対応している。つまり、ｖｘおよびｖｙを整数としたとき、（ｖｘ、ｖｙ）は参照ベクトルのそれぞれを表すことになる。したがって、以下の説明においては、（ｖｘ、ｖｙ）の位置を示す参照ベクトルを、参照ベクトル（ｖｘ、ｖｙ）と記載することがある。

【０１０６】
ここで、サーチ範囲の中心位置をターゲットブロックの位置、つまり前記基準位置（０，０）とし、サーチ範囲を、水平方向には±Ｒｘ、垂直方向には±Ｒｙとしたとき、

$$-Rx \leqq vx \leqq +Rx,\quad -Ry \leqq vy \leqq +Ry$$

とされるものである。

【０１０７】
次に、ターゲットブロックＩｏ内の１つの画素の座標（ｘ，ｙ）を指定する（ステップＳ２）。次に、ターゲットブロックＩｏ内の指定された１つの座標（ｘ，ｙ）の画素値Ｉｏ（ｘ，ｙ）と、参照ブロックＩｉ内の対応する画素位置の画素値Ｉｉ（ｘ＋ｖｘ，ｙ＋ｖｙ）との差分の絶対値αを算出する（ステップＳ３）。すなわち、差分絶対値αは、

$$\alpha = |Io(x, y) - Ii(x + vx, y + vy)| \quad \cdots (式1)$$

として算出される。

【０１０８】
そして、算出した差分絶対値αを、当該参照ブロックＩｉの参照ベクトル（ｖｘ，ｖｙ）が指し示すアドレス（テーブル要素）の、それまでのＳＡＤ値に加算し、その加算であるＳＡＤ値を、当該アドレスに書き戻すようにする（ステップＳ４）。すなわち、参照ベクトル（ｖｘ，ｖｙ）に対応するＳＡＤ値を、ＳＡＤ（ｖｘ，ｖｙ）と表すと、

$$SAD(vx, vy) = \Sigma\alpha = \Sigma |Io(x, y) - Ii(x + vx, y + vy)| \quad \cdots (式2)$$

として算出し、当該参照ベクトル（ｖｘ，ｖｙ）が指し示すアドレスに書き込むようにする。

【０１０９】
次に、ターゲットブロックＩｏ内の全ての座標（ｘ，ｙ）の画素について、上記の演算を行なったか否かを判別し（ステップＳ５）、ターゲットブロックＩｏ内の全ての座標（ｘ，ｙ）の画素については、未だ、演算は終了していないと判別したときには、ステップＳ２に戻り、ターゲットブロックＩｏ内の次の座標（ｘ，ｙ）の画素位置を指定し、このステップＳ２以降の処理を繰り返す。

【０１１０】
また、ステップＳ５で、ターゲットブロックＩｏ内の全ての座標（ｘ，ｙ）の画素について、上記の演算を行なったと判別したときには、当該参照ブロックについてのＳＡＤ値の算出が終了したと判別して、サーチ範囲内の全ての参照ブロック、すなわち、全ての参照ベクトル（ｖｘ，ｖｙ）についての上記の演算処理を完了したか否か判別する（ステップＳ６）。

【０１１１】
ステップＳ６で、未だ、上記の演算処理を完了していない参照ベクトル（ｖｘ，ｖｙ）があると判別すると、ステップＳ１に戻り、上記の演算処理を完了していない次の参照ベクトル（ｖｘ，ｖｙ）を設定して、このステップＳ１以降の処理を繰り返す。

【０１１２】
そして、ステップＳ６で、上記の演算処理を完了していない参照ベクトル（ｖｘ，ｖｙ）はサーチ範囲内になくなったと判別すると、ＳＡＤテーブルが完成したとして、当該完成したＳＡＤテーブルにおいて、最小値となっているＳＡＤ値を検出する（ステップＳ７）。そして、当該最小値となっているＳＡＤ値のアドレスに対応する参照ベクトルを動きベクトルとして検出する（ステップＳ８）。ここで、ＳＡＤの最小値をＳＡＤ（ｍｘ，ｍ

ｙ）と書き表すと、目的とする動きベクトルは、位置（ｍｘ，ｍｙ）を示すベクトル（ｍｘ，ｍｙ）として算出される。

【０１１３】

以上で、１つのターゲットブロックに対するブロックマッチングによる動きベクトルの検出処理は、終了となる

以上のようなブロックマッチングを用いたセンサレスの手ぶれ動きベクトルの検出方法の場合、原理的に、ピクセル精度の手ぶれベクトル検出が実現可能であり、また、センサやレンズシフト等の機構を削減できるため、コスト的にも相当優位である。

【０１１４】

しかし、上述した従来のブロックマッチングに依存する技術の延長では、１画面分の画素数に比例してＳＡＤテーブルの規模が増加するため、現在の５００万画素オーバーの静止画サイズの動きベクトル検出を、現実的な回路規模で実現するのは非常に困難である。

【０１１５】

過去、各社様々な工夫を凝らしながら、ＮＴＳＣ（Ｎａｔｉｏｎａｌ　Ｔｅｌｅｖｉｓｉｏｎ　Ｓｙｓｔｅｍ　Ｃｏｍｍｉｔｔｅｅ）動画の高々１７万画素の手ぶれベクトル検出の回路規模削減に苦慮していた背景がある上に、ＮＴＳＣ動画の場合、６０ｆｐｓ（ｆｒａｍｅ　ｐｅｒ　ｓｅｃｏｎｄ；フレーム／秒）のため手ぶれのサーチ範囲は狭くて済むが、静止画の場合、３ｆｐｓ程度が前提となり、サーチ範囲が極端に大きくなることも問題を困難にしている一因である。画素数と同じく、サーチ範囲にも比例してＳＡＤテーブルのテーブル要素数が増加するからである。

【０１１６】

センサレス手ぶれ補正を静止画で実現する手法としては、特許文献３（特開平７－２８３９９９号公報）を始め、幾つか提案はされている。特許文献３は、手ぶれの発生しない程度の短い露光時間で何枚かの静止画を連写撮影し、その静止画間の手ぶれベクトルを求め、その手ぶれベクトルに従って、前記連射撮影した複数枚の静止画を平行移動（およびロール軸方向に回転）させながら加算（もしくは平均化）して行くことで、最終的に手ぶれと低照度ノイズの無い高画質の静止画を得る、というアルゴリズムである。

【０１１７】

実現できるレベルの現実的な提案としては、特許文献４（特開２００５－３８３９６公報）を挙げることができる。この特許文献４に示されたものは、画像を縮小変換したサイズで動ベクトルを求める手段と、同一のＳＡＤテーブルを複数のブロックで共有する手段から構成される。画像の縮小変換と、ＳＡＤテーブルの複数ブロックでの共有化は、ＳＡＤテーブルサイズの削減を実現するための、非常に良い手法であり、ＭＰＥＧ（Ｍｏｖｉｎｇ　Ｐｉｃｔｕｒｅ　Ｅｘｐｅｒｔｓ　Ｇｒｏｕｐ）画像圧縮方式における動きベクトル検出やシーンチェンジ検出等、他分野でも使われている。

【０１１８】

しかし、この特許文献４のアルゴリズムの問題点として、画像の縮小変換と、その際のメモリ（ＤＲＡＭ（Ｄｙｎａｍｉｃ　ＲＡＭ（Ｒａｎｄｏｍ　Ａｃｃｅｓｓ　Ｍｅｍｏｒｙ））アクセスに、時間とメモリ容量を消費することと、ＳＡＤテーブルを複数ブロックで時分割アクセスする手法のため、メモリアクセスが非常に増加し、この処理にも時間を要してしまうという課題がある。動画の手ぶれ補正においては、リアルタイム性と同時にシステム遅延時間の短縮が求められるため、この処理時間の問題が課題となってしまうのである。

【０１１９】

また、元画像を縮小変換する際には、エイリアシング（折り返し歪み）や、低照度ノイズ除去のためのローパスフィルタを、縮小処理の前処理として実装するが必要である。しかし、縮小倍率に応じて、ローパスフィルタの特性が変化する上、特に、垂直方向のローパスフィルタの場合、多タップのデジタルフィルタとした場合に、多くのラインメモリと演算ロジックを必要としなければならず、回路規模増加の問題が生じる。

【０１２０】

他方、ブロックマッチングを使用しないアルゴリズムも提案されている（例えば特許文献５（特開平６－８６１４９号公報）、特許文献６（特開２００４－３４３４８３公報）など）。これらは、２つのフレーム画像内それぞれで、何らかの特徴点を複数点検出し、それらの２つのフレーム間の対応付けを取ることで全体の手ぶれベクトル（グローバルベクトル）を求める手法である。もしくは、片方のフレームのみで特徴点を検出し、その特徴点の周囲に関してのみ、もう一方のフレームに対してブロックマッチングを施しても良い。

【０１２１】

この特許文献５や特許文献６のアルゴリズムは、回路規模を削減するうえで非常に有効であり、理想的ではあるものの、現実には、２フレームに共通して存在する、画像全体の真に特徴のある特徴点を、どこまで効率的に数を絞って特定出来るかに、その実効性が掛かっている。したがって、この特許文献５や特許文献６のアルゴリズムに比べると、民生機では森羅万象が撮像対象である限り、ロバスト性においてブロックマッチング手法に一日の長があると考えられる。

【０１２２】

上述の通り、デジタルカメラなどの撮像装置では、今後益々高密度画素化が進み、高性能化が求められると予想されるが、このような状況においては、静止画の撮影時の手ぶれ補正を、ジャイロ（角速度）センサを用いないセンサレスで実現する意義は非常に大きい。

【０１２３】

そこで、上述したように、ブロックマッチングを用いて、センサレスで、手ぶれ動きベクトルを検出し、その検出した動きベクトルを用いて手ぶれ補正を行なうことが有望であるが、ブロックマッチングの現実解として、回路規模、処理速度、ロバスト性を全て満たす提案はなされていないのが現状である。

【０１２４】

そして、ブロックマッチング技法において、最大の問題となっているのが、ＳＡＤテーブルの増大である。既に述べたように、デジタルカメラにおいては、そのイメージャが５００万画素以上が前提の昨今において、画素数に比例してＳＡＤテーブルサイズが大きくならざるを得ない上、静止画の場合、３ｆｐｓ程度のため、動画の６０ｆｐｓの手ぶれ範囲と比較して、何１０倍も広いサーチ範囲が必要であり、当該サーチ範囲の拡大は、即ちＳＡＤテーブルの増大と等しいからである。

【０１２５】

多人数評価の結果、３ｆｐｓの静止画の場合における手ぶれ範囲は、全フレームを１００％として±１０％程度であることが判明している。既に高級機では世に出ている１２００万画素を仮定し、現状で提案されている技術のまま、必要なＳＡＤテーブルサイズを見積もると、約８０メガビットである。しかも、現実的な処理速度を満たそうとすると、このＳＡＤテーブル情報を格納するメモリは、内蔵ＳＲＡＭ（Ｓｔａｔｉｃ　ＲＡＭ（Ｒａｎｄｏｍ　Ａｃｃｅｓｓ　Ｍｅｍｏｒｙ））であることが求められる。半導体プロセスルールが進んだとは言え、このサイズは、ほぼ3桁程度、現実的なレベルからはかけ離れている。

【０１２６】

この実施の形態では、以上の点にかんがみ、ブロックマッチングを用いて２フレーム間の動きベクトルを検出する場合において、ＳＡＤテーブルサイズの大幅な削減が可能である画像処理方法および装置を提供するようにする。

【０１２７】

また、前述したブロックマッチングにおける従来の提案手法のうち、特許文献４に記載された画像の縮小変換によるＳＡＤテーブルの削減手法に関して、２つの問題を提起した。画像の縮小変換に伴う処理時間の増大並びにメモリ容量の消費と、画像の縮小変換に伴うエイリアシング回避のための適切なローパスフィルタの実装に伴う回路増大の問題である。以下に説明する実施の形態においては、これらの問題点をも解決したものである。

【０１２８】
［実施の形態で用いる新規なブロックマッチング手法の概要］
この実施の形態においても、上述したブロックマッチングを用いて、２フレーム間の動きベクトルを検出するのであるが、ターゲットブロックと参照ブロック間において求められるＳＡＤ値を、参照ブロックの参照ベクトルに対応して記憶するのではなく、当該参照ベクトルを縮小し、その縮小した参照縮小ベクトルに対応する、当該参照縮小ベクトルの近傍の複数の参照ベクトルに分散加算して記憶するようにする。
【０１２９】
これにより、従来のＳＡＤテーブルに比較して、ＳＡＤテーブルのサイズを大幅に縮小するようにするものである。
【０１３０】
図６～図８は、実施の形態で用いる新規なブロックマッチング手法の概要を説明するための図である。図６は、従来のＳＡＤテーブルＴＢＬｏと、実施形態で用いる新規なブロックマッチング手法において生成される縮小ＳＡＤテーブルＴＢＬｓとの関係を示すものである。
【０１３１】
この実施形態においても、図３に示したように、従来と同様に参照フレームにおいて、元フレームに設定されたターゲットブロックの位置を中心としてサーチ範囲が設定される。そして、このサーチ範囲において、前述したような複数の参照ブロックが設定され、各参照ブロック内の画素とターゲットブロック内の対応する画素の輝度値の差分の絶対値の総和、つまり、ＳＡＤ値が求められる。
【０１３２】
従来は、求められたＳＡＤ値は、図６に示すように、対象となっている参照ブロックの参照ベクトルＲＶに対応するアドレスのテーブル要素ｔｂｌとしてＳＡＤテーブルＴＢＬｏに書き込まれる。
【０１３３】
したがって、従来のブロックマッチングでは、ターゲットブロックと参照ブロックとのフレーム画像上における位置ずれ量を表わす参照ベクトルＲＶと、ＳＡＤテーブルＴＢＬｏの各テーブル要素である参照ブロックのＳＡＤ値とは、１対１に対応している。すなわち、従来のＳＡＤテーブルＴＢＬｏでは、サーチ範囲で取り得る参照ベクトルＲＶと等しい数のＳＡＤ値のテーブル要素数を備えるものとなっている。
【０１３４】
これに対して、この実施形態におけるブロックマッチングでは、図６および図７（Ａ）、（Ｂ）に示すように、対象となっている参照ブロックの参照ベクトルＲＶは、縮小率１／ｎ（ｎは自然数）で縮小されて参照縮小ベクトルＣＶとされる。
【０１３５】
ここで、以下の説明においては、説明の便宜上、水平方向縮小倍率と垂直方向の縮小倍率とを同じとしているが、水平方向縮小倍率と垂直方向の縮小倍率とは独立の異なる値でも良い。また、後で述べるが、水平方向縮小倍率と垂直方向の縮小倍率とを独立に任意の自然数分の１として設定できるようにしておく方が、柔軟性も高く好都合である。
【０１３６】
この実施形態においても、前述の従来のブロックマッチング手法について説明したのと同様に、サーチ範囲の中心とされるターゲットブロックの位置を基準位置（０，０）とし、参照ベクトルは、当該基準位置からの画素単位の水平方向および垂直方向のずれ量（ｖｘ，ｖｙ）（ｖｘ、ｖｙは、整数）を指し示すものとされ、参照ベクトルＲＶのそれぞれは、参照ベクトル（ｖｘ，ｖｙ）で表される。
【０１３７】
参照ベクトル（ｖｘ，ｖｙ）が、水平方向および垂直方向のそれぞれについて１／ｎに縮小された参照縮小ベクトル（ｖｘ／ｎ，ｖｙ／ｎ）で示される位置（ｖｘ／ｎ，ｖｙ／ｎ）は、整数ではなく、小数成分が発生することがある。このため、この実施形態では、

縮小前の元の参照ベクトルＲＶに対応して求められたＳＡＤ値を、参照縮小ベクトルＣＶに最も近い１つの参照ベクトルに対応するテーブル要素として記憶してしまうと誤差が生じることになる。

【０１３８】

そこで、この実施形態では、まず、参照縮小ベクトルＣＶで示される位置（ｖｘ／ｎ，ｖｙ／ｎ）の近傍の複数の参照ベクトルで示される複数の位置（テーブル要素）を検出する。そして、参照ベクトルＲＶの参照ブロックについて求められたＳＡＤ値は、その検出した近傍の複数の参照ベクトルに対応するＳＡＤ値に分散して加算するようにする。

【０１３９】

この場合に、この実施形態では、参照縮小ベクトルＣＶで示される位置の近傍の周囲の複数個の参照ベクトルで示される位置に対応するテーブル要素ｔｂｌに書き込むべき成分として分散加算する値は、縮小前の元の参照ベクトルＲＶに対応して求められたＳＡＤ値から、参照縮小ベクトルとその近傍の参照ベクトルとのそれぞれが示す位置の関係を用いて、前記近傍の参照ベクトルのそれぞれに対応して分散加算するＳＡＤ値を算出し、算出したＳＡＤ値のそれぞれを、対応する参照ベクトルのテーブル要素成分として加算するようにする。

【０１４０】

ここで、分散するだけでなく加算するというのは、参照縮小ベクトルの近傍の複数の参照ベクトルは、異なる複数の参照縮小ベクトルについて重複して検出されることになるので、１つの参照ベクトルについて、重複したＳＡＤ値は加算するという意味である。

【０１４１】

なお、参照縮小ベクトルＣＶが示す位置（ｖｘ／ｎ，ｖｙ／ｎ）が、参照ベクトルが示す位置に一致する場合、つまり、ｖｘ／ｎおよびｖｙ／ｎの値が整数であるときには、周辺の複数の参照ベクトルを検出する必要はなく、当該位置（ｖｘ／ｎ，ｖｙ／ｎ）を示す参照ベクトルに対応して、縮小前の元の参照ベクトルＲＶに対応して求められたＳＡＤ値を記憶するようにするものである。

【０１４２】

次に、具体例を挙げて、以上の処理を説明する。例えば、ターゲットブロックの位置を基準（０，０）としたときに、図７（Ａ）に示すように、（－３，－５）の位置を示す参照ブロックＲＶを、水平方向および垂直方向に、１／ｎ＝１／４倍に縮小すると、その参照縮小ベクトルＣＶで示される位置は、図７（Ｂ）に示すように、（－０．７５，－１．２５）となる。

【０１４３】

したがって、参照縮小ベクトルＣＶで示される位置は小数成分が発生し、参照ベクトルで示される位置とは一致しない。

【０１４４】

そこで、この場合には、図８に示すように、当該参照縮小ベクトルＣＶが示す位置の近傍位置を示す複数個の近傍参照ベクトルが検出される。図８の例では、１つの参照縮小ベクトルＣＶに対して、４個の近傍参照ベクトルＮＶ１，ＮＶ２，ＮＶ３，ＮＶ４が検出される。

【０１４５】

そして、前述したように、この実施形態では、参照ベクトルＲＶの参照ブロックについて求められたＳＡＤ値は、これら４個の近傍参照ベクトルＮＶ１，ＮＶ２，ＮＶ３，ＮＶ４に対応するＳＡＤ値として分散加算される。

【０１４６】

この場合に、この実施形態では、４個の近傍参照ベクトルＮＶ１，ＮＶ２，ＮＶ３，ＮＶ４のそれぞれに分散加算するＳＡＤ値は、参照縮小ベクトルＣＶで示される位置Ｐ０（図８において×印として示す）と、４個の近傍参照ベクトルＮＶ１，ＮＶ２，ＮＶ３，ＮＶ４のそれぞれで示される位置Ｐ１，Ｐ２，Ｐ３，Ｐ４（図８において○印として示す）との位置関係を用いて線形加重分散値として算出する。

【０１４７】
図８の例の場合には、参照縮小ベクトルＣＶで示される位置Ｐ０は、周辺近傍の４個の参照ベクトルＮＶ１，ＮＶ２，ＮＶ３，ＮＶ４のそれぞれで示される位置Ｐ１，Ｐ２，Ｐ３，Ｐ４を、水平方向に１：３、垂直方向に３：１に内分する位置にある。
【０１４８】
そこで、縮小前の元の参照ベクトルＲＶに対応して求められたＳＡＤ値をＳαとしたとき、周辺近傍の４個の参照ベクトルＮＶ１，ＮＶ２，ＮＶ３，ＮＶ４のそれぞれで示される位置Ｐ１，Ｐ２，Ｐ３，Ｐ４に対応するＳＡＤテーブル要素に分散加算する値ＳＡＤｐ１，ＳＡＤｐ２，ＳＡＤｐ３，ＳＡＤｐ４のそれぞれは、
ＳＡＤｐ１＝Ｓα×９／１６
ＳＡＤｐ２＝Ｓα×３／１６
ＳＡＤｐ３＝Ｓα×３／１６
ＳＡＤｐ４＝Ｓα×１／１６
となる。
【０１４９】
そして、この実施形態では、求められた値ＳＡＤｐ１，ＳＡＤｐ２，ＳＡＤｐ３，ＳＡＤｐ４のそれぞれを、近傍の４個の参照ベクトルＮＶ１，ＮＶ２，ＮＶ３，ＮＶ４のそれぞれで示される位置Ｐ１，Ｐ２，Ｐ３，Ｐ４に対応するＳＡＤテーブル要素にそれぞれ加算する。
【０１５０】
この実施形態では、以上の処理を、サーチ範囲内のすべての参照ブロックについて行なう。
【０１５１】
以上のことから、この実施形態では、参照ベクトルＲＶを１／ｎに縮小する場合には、全ての参照ベクトルに１対１に対応する従来サイズのＳＡＤテーブルＴＢＬｏに対して、水平方向に１／ｎ、また、垂直方向に１／ｎに縮小した縮小ＳＡＤテーブルＴＢＬｓを用意して、この縮小ＳＡＤテーブルＴＢＬｓのテーブル要素として、参照縮小ベクトルＣＶの近傍の参照ベクトルに対応するＳＡＤ値を求めるようにすれば良い（図６参照）。
【０１５２】
したがって、この実施形態の場合には、縮小ＳＡＤテーブルＴＢＬｓのテーブル要素の数は、従来のＳＡＤテーブルＴＢＬｏのテーブル要素数の１／n^2となり、テーブルサイズを大幅に小さくすることが可能である。
【０１５３】
なお、上述の実施形態の説明においては、参照縮小ベクトルＣＶの近傍の４個の参照ベクトルを検出し、当該４個の近傍参照ベクトルに対応するＳＡＤテーブル要素に対して、対照の参照ブロック（参照ベクトルＲＶ）について算出したＳＡＤ値を線形加重分散加算するようにしたが、参照縮小ベクトルＣＶの近傍の複数の参照ベクトルの選び方およびその近傍参照ベクトルに対応するＳＡＤテーブル要素に対する分散加算の方法は、上述の例に限られるものではない。
【０１５４】
例えば、参照縮小ベクトルＣＶの近傍の９個もしくは１６個の参照ベクトルを検出し、当該９個もしくは１６個の近傍参照ベクトルに対応するＳＡＤテーブル要素に対して、いわゆるキュービック（Ｃｕｂｉｃ）補間による分散加算を行なうようにすれば、より精度は高くなる。しかし、リアルタイム性と演算回路の削減を重視すると、上述した近傍４個の参照ベクトルに対応するテーブル要素への線形加重分散加算が、より有効である。
【０１５５】
この実施形態においても、参照ブロックをサーチ範囲内で遍く移動させ、全ての参照ブロックのＳＡＤ値に対してＳＡＤテーブル（この実施形態では縮小ＳＡＤテーブル）への代入を行う点は、従来手法と同じである。
【０１５６】

ただし、従来は、参照ベクトルとＳＡＤテーブル要素のアドレスが１対１に対応していたため、ＳＡＤテーブルへは単なる代入で済んだが、この実施形態による手法では、参照ブロックについて算出したＳＡＤ値を分散加算させるため、縮小ＳＡＤテーブルにおいて、参照ベクトルとテーブルアドレスは１対１ではない。したがって、この実施形態の手法の場合には、ＳＡＤ値のテーブルアドレスへの単なる代入ではなく、加算して代入する、いわゆる代入加算である必要がある。また、そのため、ＳＡＤテーブル（縮小ＳＡＤテーブル）の各テーブル要素は、最初に初期化（０クリア）しておかなければならない。

【０１５７】

ところで、従来のブロックマッチング手法では、以上のようにして完成させたＳＡＤテーブルにおいて、ＳＡＤ値が最小値となるテーブル要素を探索し、その最小値となるテーブル要素のテーブルアドレスを、参照ベクトルに変換すれば、動ベクトルの検出を完了した。

【０１５８】

これに対して、この実施形態による手法では、ＳＡＤテーブルは、参照ベクトルを縮小した参照縮小ベクトルに対応した縮小ＳＡＤテーブルであるため、当該縮小ＳＡＤテーブルの最小値がそのまま正確な動きベクトルには対応していない。

【０１５９】

もっとも、ある程度の誤差を許す装置の場合には、縮小ＳＡＤテーブルの最小値となるテーブル要素のテーブルアドレスを、参照ベクトルに変換したものを、さらに縮小率１／ｎの逆数倍、つまりｎ倍することで、動きベクトルを検出するようにすることもできる。

【０１６０】

しかし、より正確な動きベクトルを検出するようにする場合には、以下に説明するように、縮小ＳＡＤテーブルのテーブル要素値に対して補間処理を施すことで、元のベクトル精度で、正確な動きベクトルを検出するようにする。

【０１６１】

［より正確な動きベクトルを検出するための補間処理の第１の例］

より正確な動きベクトルを検出するための補間処理の第１の例は、縮小ＳＡＤテーブルにおける複数個のＳＡＤテーブル要素値（ＳＡＤ値）を、１つの２次曲面で近似する手法である。この手法は、前述した特許文献１に記載されている手法を縮小ＳＡＤテーブルに対して適用した手法である。

【０１６２】

すなわち、縮小ＳＡＤテーブルにおいて、ＳＡＤ値が最小値となるテーブル要素（整数精度最小値テーブル要素（整数精度テーブルアドレス））と、この整数精度最小値テーブル要素を中心とする複数の整数精度テーブル要素とを求め、それらのテーブル要素のＳＡＤ値を用いて、最小自乗法によりＳＡＤ値の２次曲面を決定し、この２次曲面の最小値となるＳＡＤ値を検出し、当該検出した最小値となるＳＡＤ値に対応する位置（参照フレーム上において、基準位置に対してずれた位置）を検出し、当該検出した位置を小数精度の最小値テーブルアドレス（縮小ＳＡＤテーブルにおいてＳＡＤ値が最小値となるベクトル（最小値ベクトルという）に対応）とする。

【０１６３】

この場合、一意の２次曲面を定めるためには、図９（Ａ）または（Ｂ）に示すように、整数精度最小値テーブル要素ｔｍと、当該テーブル要素ｔｍをその両側から挟む位置の、当該テーブル要素ｔｍの近傍の４個の整数精度テーブル要素ｔ１，ｔ２，ｔ３，ｔ４が最低限必要である。

【０１６４】

そして、図１０に示すように、参照フレームのサーチ範囲内の縮小ＳＡＤテーブルに対応する参照縮小ベクトルの範囲内において、ターゲットフレームの位置を基準位置（０，０）として、水平方向および垂直方向のずれ量（参照縮小ベクトルに対応）の軸ｖｘ／ｎおよび軸ｖｙ／ｎを考えると共に、これらの軸ｖｘ／ｎおよび軸ｖｙ／ｎに垂直な軸として、ＳＡＤ値の軸を考え、これら３軸からなる座標空間を想定する。

【０１６５】
そして、例えば、整数精度最小値テーブル要素ｔｍのＳＡＤ値と、当該整数精度最小値テーブル要素ｔｍを挟む２個のテーブル要素ｔ１、ｔ３のＳＡＤ値とから、図１０の座標空間において２次曲線を生成する。また、整数精度最小値テーブル要素ｔｍのＳＡＤ値と、当該最小値テーブル要素ｔｍを挟む他の２個のテーブル要素ｔ２、ｔ４のＳＡＤ値とから、図１０の座標空間において、他の２次曲線を生成する。そして、これら２個の２次曲線を含む２次曲面２０１を、最小自乗法により求め、その２次曲面２０１を、図１０に示すように、座標空間において生成する。
【０１６６】
そして、生成されたＳＡＤ値の２次曲面２０１の最小値２０２を検出し、その最小値を取るＳＡＤ値に対応する位置（ｖｘ／ｎ，ｖｙ／ｎ）（図１０の位置２０３）を検出し、当該検出した位置（ｖｘ／ｎ，ｖｙ／ｎ）を、小数精度のテーブル要素（テーブルアドレス）として検出する。そして、検出した小数精度テーブル要素に対応するベクトル（最小値ベクトル）２０４を、図１１に示すようにｎ倍して、元の大きさ精度の動きベクトル２０５を得る。
【０１６７】
例えば、図１２に示すように、参照ベクトルを１／４に縮小した場合における縮小ＳＡＤテーブルＴＢＬｓの、小数精度テーブル要素の最小値アドレスから求められる最小値ベクトル２０４が、（－０．７７７，－１．４９２）の場合に、これらを4倍した（－３．１０８，－５．９６８）が、動きベクトル２０５となる。この動きベクトル２０５は、元画像のスケールにおける動きベクトルを再現したものとなっている。
【０１６８】
以上の説明は、整数精度最小値テーブル要素ｔｍと、その近傍の４テーブル要素を用いた場合として説明したが、ＳＡＤ値の２次曲面を最小自乗法により求めるためには、より多くの複数近傍テーブル要素を用いたほうがよい。そこで、一般には、整数精度最小値テーブル要素ｔｍを中心として、その周囲の水平方向×垂直方向＝ｍ×ｍ個（ｍは３以上の整数）の矩形領域のテーブル要素を用いるようにする。
【０１６９】
しかし、この複数近傍テーブル要素の数は、多ければ良いというものではなく、広い範囲のテーブル要素を用いると、演算量の増加を招く上、画像パターンに依存するローカルミニマムの偽値を使用してしまう可能性も高まるので、適切な複数近傍テーブル要素の数からなる矩形領域のテーブル要素を用いるようにする。
【０１７０】
適切な複数近傍テーブル要素の数からなる矩形領域のテーブル要素の例として、この実施形態では、整数精度最小値テーブル要素ｔｍを中心として、その周囲の水平方向×垂直方向＝３×３個の矩形領域のテーブル要素を用いるようにする例と、整数精度最小値テーブル要素ｔｍを中心として、その周囲の水平方向×垂直方向＝４×４個の矩形領域のテーブル要素を用いるようにする例とについて説明する。
【０１７１】
［３×３個の矩形領域のテーブル要素を用いる例］
図１３に、整数精度最小値テーブル要素ｔｍを中心として、その周囲の水平方向×垂直方向＝３×３個の矩形領域（図１３で塗りを付して示してある）のテーブル要素を用いるようにする例を示す。
【０１７２】
この図１３の例の場合には、図１３（Ａ）に示すように、整数精度最小値テーブル要素ｔｍと、その近傍の８個の近傍テーブル要素のＳＡＤ値を用いて、図１３（Ｂ）に示すような２次曲面２０１を、最小自乗法により生成する。そして、生成されたＳＡＤ値の２次曲面２０１の最小値２０２を検出し、その最小値を取るＳＡＤ値に対応する位置（ｖｘ／ｎ，ｖｙ／ｎ）（図１３（Ｂ）の位置２０３）を検出し、当該検出した位置２０３を、小数精度の最小値テーブル要素位置（小数精度最小値テーブルアドレス）として検出する。

【０１７３】
そして、検出した小数精度テーブル要素位置２０３に対応するベクトル（最小値ベクトル）２０４を、前述した図１１に示すようにｎ倍して、元の大きさ精度の動きベクトル２０５を得る。
【０１７４】
ここで、ＳＡＤ値の２次曲面２０１の最小値２０２に対応する位置２０３の算出方法は、次のようになる。すなわち、図１４に示すように、整数精度最小値テーブル要素ｔｍの位置を原点（０，０）とする（ｘ，ｙ）座標を考える。この場合、周辺の８個のテーブル要素の位置は、３個のｘ軸方向の位置、すなわち、ｘ＝－１、ｘ＝０、ｘ＝１と、３個のＹ軸方向の位置、すなわち、ｙ＝－１、ｙ＝０、ｙ＝１との繰り合わせで表され、（－１，－１）、（０，－１）、（１，－１）、（－１，０）、（０，１）、（－１，１）、（０，１）、（１，１）の８位置となる。
【０１７５】
そして、図１４のテーブルにおける各テーブル要素のＳＡＤ値を、Ｓｘｙとする。したがって、例えば、整数精度最小値テーブル要素ｔｍ（位置（０，０））のＳＡＤ値は$S_{0\ 0}$と表され、また、右下の位置（１，１）のテーブル要素値のＳＡＤ値は$S_{1\ 1}$と表される。
【０１７６】
すると、整数精度最小値テーブル要素ｔｍの位置を原点（０，０）とする（ｘ，ｙ）座標における小数精度の位置（ｄｘ、ｄｙ）は、図１５に示す（式Ａ）および（式Ｂ）により、求めることができる。
【０１７７】
図１５の（式Ａ）および（式Ｂ）において、
ｘ＝－１のとき、Ｋｘ＝－１
ｘ＝０のとき、Ｋｘ＝０
ｘ＝０のとき、Ｋｘ＝１
となる。また、
ｙ＝－１のとき、Ｋｙ＝－１
ｙ＝０のとき、Ｋｙ＝０
ｙ＝０のとき、Ｋｙ＝１
となる。
【０１７８】
こうして求められた小数精度の位置（ｄｘ，ｄｙ）は、整数精度最小値テーブル要素ｔｍの位置を原点（０，０）とする位置であるので、この小数精度の位置（ｄｘ，ｄｙ）と整数精度最小値テーブル要素ｔｍの位置とから、求めるサーチ範囲の中心位置からの位置２０３を検出することができる。
【０１７９】
［４×４個の矩形領域のテーブル要素を用いる例］
図１６に、整数精度最小値テーブル要素ｔｍをほぼ中心として、その周囲の水平方向×垂直方向＝４×４個の矩形領域のテーブル要素（図１６で塗りを付して示してある）を用いるようにする例を示す。
【０１８０】
この場合に、整数精度最小値テーブル要素ｔｍと、その近傍の８テーブル要素（３×３）や、その近傍の２４テーブル要素（５×５）のように、前記ｍの値が奇数である場合には、整数精度最小値テーブル要素ｔｍは、常に、使用する矩形領域の複数のテーブル要素の中心になるため、使用するテーブル範囲は単純に決定する。
【０１８１】
これに対して、近傍の１５テーブル要素（４×４）のように、ｍが偶数である場合には、整数精度最小値テーブル要素ｔｍは、使用する矩形領域の複数のテーブル要素の中心位置とはならないので、若干の工夫が必要となる。

【０１８２】
つまり、整数精度最小値テーブル要素ｔｍから見て、水平方向に左右の隣接テーブル要素のＳＡＤ値を比較し、小さい値となった側の方向の、当該方向の隣接テーブル要素に隣接するテーブル要素を近傍テーブル要素の４列目として採用する。同様に、垂直方向に上下の隣接テーブル要素のＳＡＤ値を比較し、小さい値となった側の方向の、当該方向の隣接テーブル要素に隣接するテーブル要素を近傍テーブル要素の４行目として採用する。
【０１８３】
図１６の例では、整数精度最小値テーブル要素ｔｍの水平方向に左右の隣接テーブル要素のＳＡＤ値は、「１７７」と「１７３」であるので、ＳＡＤ値が小さい右隣の値「１７３」のテーブル要素のさらに右隣の列を第４列目として採用する。また、整数精度最小値テーブル要素ｔｍの垂直方向に上下の隣接テーブル要素のＳＡＤ値は、「１６８」と「１８２」であるので、ＳＡＤ値が小さい上隣の値「１６８」のテーブル要素のさらに上隣の行を第４行目として採用する。
【０１８４】
そして、図１６の例の場合には、整数精度最小値テーブル要素ｔｍと、その近傍の１５個の近傍テーブル要素のＳＡＤ値を用いて、２次曲面２０１を、最小自乗法により生成する。そして、生成されたＳＡＤ値の２次曲面２０１の最小値２０２を検出し、その最小値を取るＳＡＤ値に対応する位置（ｖｘ／ｎ，ｖｙ／ｎ）（図１６の位置２０３）を検出し、当該検出した位置２０３を、小数精度の最小値テーブル要素位置（小数精度最小値テーブルアドレス）として検出する。
【０１８５】
そして、検出した小数精度テーブル要素位置２０３に対応するベクトル（最小値ベクトル）２０４を、前述した図１１に示すようにｎ倍して、元の大きさ精度の動きベクトル２０５を得る。
【０１８６】
ここで、この例の場合におけるＳＡＤ値の２次曲面２０１の最小値２０２に対応する位置２０３の算出方法は、次のようになる。すなわち、図１７に示すように、整数精度最小値テーブル要素ｔｍの位置を原点（０，０）とする（ｘ，ｙ）座標を考える。
【０１８７】
この例の場合には、１６テーブル要素からなる矩形領域中における整数精度最小値テーブル要素ｔｍの位置に応じて、図１７（Ａ），（Ｂ），（Ｃ），（Ｄ）のような４通りのテーブル要素配置を考える必要がある。
【０１８８】
この場合、周辺の１５個のテーブル要素の位置は、図１７（Ａ），（Ｂ），（Ｃ），（Ｄ）から分かるように、４個のｘ軸方向の位置、すなわち、ｘ＝－１、ｘ＝０、ｘ＝１、ｘ＝２またはｘ＝－２と、４個のＹ軸方向の位置、すなわち、ｙ＝－１、ｙ＝０、ｙ＝１、ｙ＝２またはｙ＝－２との繰り合わせで表される１５位置となる。
【０１８９】
そして、図１７のテーブルにおける各テーブル要素のＳＡＤ値を、Ｓｘｙとする。したがって、例えば、整数精度最小値テーブル要素ｔｍ（位置（０，０））のＳＡＤ値は$S_{0\ 0}$と表され、また、位置（１，１）のテーブル要素値のＳＡＤ値は$S_{1\ 1}$と表される。
【０１９０】
すると、整数精度最小値テーブル要素ｔｍおよびその周辺の１６テーブル要素からなる矩形領域中の中心位置を原点（０，０）とする（ｘ，ｙ）座標における小数精度の位置（ｄｘ，ｄｙ）は、図１８に示す（式Ｃ）および（式Ｄ）により、求めることができる。
【０１９１】
ここで、図１８の（式Ｃ）および（式Ｄ）において、ＫｘおよびＫｙは、整数精度最小値テーブル要素ｔｍおよびその周辺の１６テーブル要素からなる矩形領域中の中心位置を原点（０，０）とする（Ｋｘ，Ｋｙ）座標を考えたときの、前記図１７（Ａ），（Ｂ），（Ｃ），（Ｄ）に示した４通りのテーブル要素配置に応じた値となる。このときの（Ｋｘ

，Ｋｙ）座標を図１９に示す。

【０１９２】

すなわち、図１７（Ａ）の場合には、

ｘ＝－２のとき、Ｋｘ＝－１．５

ｘ＝－１のとき、Ｋｘ＝－０．５

ｘ＝０のとき、Ｋｘ＝０．５

ｘ＝１のとき、Ｋｘ＝１．５

となる。また、

ｙ＝－２のとき、Ｋｙ＝－１．５

ｙ＝－１のとき、Ｋｙ＝－０．５

ｙ＝０のとき、Ｋｙ＝０．５

ｙ＝１のとき、Ｋｙ＝１．５

となる。

【０１９３】

また、図１７（Ｂ）の場合には、

ｘ＝－２のとき、Ｋｘ＝－１．５

ｘ＝－１のとき、Ｋｘ＝－０．５

ｘ＝０のとき、Ｋｘ＝０．５

ｘ＝１のとき、Ｋｘ＝１．５

となる。また、

ｙ＝－１のとき、Ｋｙ＝－１．５

ｙ＝０のとき、Ｋｙ＝－０．５

ｙ＝１のとき、Ｋｙ＝０．５

ｙ＝２のとき、Ｋｙ＝１．５

となる。

【０１９４】

また、図１７（Ｃ）の場合には、

ｘ＝－１のとき、Ｋｘ＝－１．５

ｘ＝０のとき、Ｋｘ＝－０．５

ｘ＝１のとき、Ｋｘ＝０．５

ｘ＝２のとき、Ｋｘ＝１．５

となる。また、

ｙ＝－２のとき、Ｋｙ＝－１．５

ｙ＝－１のとき、Ｋｙ＝－０．５

ｙ＝０のとき、Ｋｙ＝０．５

ｙ＝１のとき、Ｋｙ＝１．５

となる。

【０１９５】

また、図１７（Ｄ）の場合には、

ｘ＝－１のとき、Ｋｘ＝－１．５

ｘ＝０のとき、Ｋｘ＝－０．５

ｘ＝１のとき、Ｋｘ＝０．５

ｘ＝２のとき、Ｋｘ＝１．５

となる。また、

ｙ＝－１のとき、Ｋｙ＝－１．５

ｙ＝０のとき、Ｋｙ＝－０．５

ｙ＝１のとき、Ｋｙ＝０．５

ｙ＝２のとき、Ｋｙ＝１．５

となる。

【０１９６】

また、図１８の（式Ｃ）および（式Ｄ）におけるΔｘおよびΔｙは、（Ｋｘ，Ｋｙ）座標に対する図１７（Ａ），（Ｂ），（Ｃ），（Ｄ）の各テーブル要素配置における（ｘ，ｙ）座標とのずれ量を表しており、

図１７（Ａ）の場合には、Δｘ＝－０．５、Δｙ＝－０．５、

図１７（Ｂ）の場合には、Δｘ＝－０．５、Δｙ＝０．５、

図１７（Ｃ）の場合には、Δｘ＝０．５、Δｙ＝－０．５、

図１７（Ｄ）の場合には、Δｘ＝０．５、Δｙ＝０．５、

となる。

【０１９７】

こうして求められた小数精度の位置（ｄｘ，ｄｙ）は、整数精度最小値テーブル要素ｔｍの位置を原点（０，０）とする位置であるので、この小数精度の位置（ｄｘ，ｄｙ）と整数精度最小値テーブル要素ｔｍの位置とから、求めるサーチ範囲の中心位置からの位置２０３を検出することができる。

【０１９８】

［より正確な動きベクトルを検出するための補間処理の第２の例］

より正確な動きベクトルを検出するための補間処理の第２の例は、縮小ＳＡＤテーブルにおける整数精度最小値テーブル要素を含む複数個の水平方向のテーブル要素のＳＡＤ値を用いて水平方向の３次曲線を生成すると共に、整数精度最小値テーブル要素を含む複数個の垂直方向のテーブル要素のＳＡＤ値を用いて垂直方向の３次曲線を生成し、それぞれの３次曲線の極小値となる位置（ｖｘ，ｖｙ）を検出して、検出した位置を小数精度の最小値アドレスとするものである。

【０１９９】

図２０は、この第２の例を説明するための図である。前述の第１の例と同様にして、整数精度最小値テーブル要素ｔｍと、この整数精度最小値テーブル要素を中心とする複数の整数精度テーブル要素、図２０の例では、４×４＝１６個のテーブル要素を求める（図２０（Ａ）で塗りを付した部分参照）。

【０２００】

次に、第１の例と同様にして、図２０（Ｂ）に示すように、参照フレームのサーチ範囲内の縮小ＳＡＤテーブルに対応する参照縮小ベクトルの範囲内において、ターゲットフレームの位置を基準位置（０，０）として、水平方向および垂直方向のずれ量（参照縮小ベクトルに対応）の軸ｖｘ／ｎおよび軸ｖｙ／ｎを考えると共に、これらの軸ｖｘ／ｎおよび軸ｖｙ／ｎに垂直な軸として、ＳＡＤ値の軸を考え、これら３軸からなる座標空間を想定する。

【０２０１】

次に、整数精度最小値テーブル要素ｔｍの周囲の１６個のテーブル要素のうち、整数精度最小値テーブル要素ｔｍを含む４個の水平方向のテーブル要素のＳＡＤ値を用いて、前記座標空間に水平方向の３次曲線２０６を生成する。この水平方向の３次曲線２０６の極小値に対応する水平方向の位置ｖｘ／ｎとして、小数精度最小値テーブル要素位置の水平方向位置を検出する。

【０２０２】

次に、整数精度最小値テーブル要素ｔｍの周囲の１６個のテーブル要素のうち、整数精度最小値テーブル要素ｔｍを含む４個の垂直方向のテーブル要素のＳＡＤ値を用いて、前記座標空間に垂直方向の３次曲線２０７を生成する。この垂直方向の３次曲線２０７の極小値に対応する垂直方向の位置ｖｙ／ｎとして、小数精度最小値テーブル要素位置の垂直方向位置を検出する。

【０２０３】

以上により求めた小数精度最小値テーブル要素位置の水平方向の位置と、垂直方向の位置から、小数精度最小値テーブル要素位置（小数精度最小値テーブルアドレス）２０８を検出する。そして、当該検出した小数精度テーブル要素位置２０８に対応するベクトル（最小値ベクトル）２０９を、前述した図１１に示すようにｎ倍して、元の大きさ精度の動

きベクトルを得る。

【０２０４】

すなわち、第２の例は、第１の例で説明した方法により、水平方向、垂直方向のそれぞれの４個のテーブル要素を確定し、図２０（Ｂ）に示すように、水平方向、垂直方向のそれぞれで、３次曲線を一意に定める手法である。

【０２０５】

ここで、ＳＡＤ値の３次曲線２０６および２０７の最小値２０２に対応する位置２０８の算出方法は、次のようになる。すなわち、水平方向または垂直方向のいずれかの方向における３次曲線において、最小値の近傍の４点のＳＡＤ値を、前記水平方向または垂直方向のいずれかの方向に沿った順番に、$Ｓ_0$、$Ｓ_1$、$Ｓ_2$、$Ｓ_3$としたとき、小数精度の最小値が、図２１に示す３つの区間Ｒａ，Ｒｂ，Ｒｃのいずれにあるかにより、最小値を取る小数成分ｕを算出する式が異なる。

【０２０６】

ここで、区間Ｒａは$Ｓ_0$となる位置と$Ｓ_1$となる位置との間の区間、Ｒｂは$Ｓ_1$となる位置と$Ｓ_2$となる位置との間の区間、Ｒｃは$Ｓ_2$となる位置と$Ｓ_3$となる位置との間の区間である。

【０２０７】

そして、小数精度の最小値が、図２１に示す区間Ｒａにあるときには、図２２の（式Ｅ）により、整数精度の最小値の位置に対する最小値を取る位置までのずれの小数成分ｕが算出される。

【０２０８】

また、同様に、小数精度の最小値が、図２１に示す区間Ｒｂにあるときには、図２２の（式Ｆ）により、整数精度の最小値の位置に対する最小値を取る位置までのずれの小数成分ｕが算出される。

【０２０９】

さらに、小数精度の最小値が、図２１に示す区間Ｒｃにあるときには、図２２の（式Ｇ）により、整数精度の最小値の位置に対する最小値を取る位置までのずれの小数成分ｕが算出される。

【０２１０】

そして、小数精度の最小値が、図２１に示す３つの区間Ｒａ，Ｒｂ，Ｒｃのいずれにあるかの判別は、次のようにして行なう。

【０２１１】

すなわち、図２３は、その判別を説明するための図である。図２３（Ａ），（Ｂ），（Ｃ）に示すように、先ず、整数精度のＳＡＤ値の最小値Ｓｍｉｎと、２番目に小さい整数精度のＳＡＤ値Ｓｎ２とを検出し、小数精度の最小値は、検出された整数精度のＳＡＤ値の最小値Ｓｍｉｎの位置と、２番目に小さい整数精度のＳＡＤ値Ｓｎ２の位置との間の区間に存在するとして検出する。次に、整数精度のＳＡＤ値の最小値Ｓｍｉｎと、２番目に小さい整数精度のＳＡＤ値Ｓｎ２とが、図２１に示したＳＡＤ値$Ｓ_0$、$Ｓ_1$、$Ｓ_2$、$Ｓ_3$のいずれの位置となっているかにより、検出した区間が区間Ｒａ，Ｒｂ，Ｒｃのいずれであるかの判別を行なう。

【０２１２】

なお、図２３（Ｄ）に示すように、整数精度のＳＡＤ値の最小値ＳｍｉｎがＳＡＤ値のまたは位置にあって、４個のテーブル要素値の端に位置する場合には、最小位置が推定できないとして、この実施形態では、エラーとして扱い、最小値位置の算出は行なわないようにする。もっとも、この図２３（Ｄ）のような場合においても、最小値位置を算出するようにしてもよい。

【０２１３】

以上のようにして、この実施形態によれば、$１／ｎ^2$にスケールダウンした小さいサイズの縮小ＳＡＤテーブルを用いて、元の画像スケールにおける動きベクトルを検出することができる。この場合に、$１／ｎ^2$にスケールダウンした小さいサイズの縮小ＳＡＤテー

ブルを用いているにも関わらず、従来とほぼ同様のベクトル検出結果が得られることを図２４に示す。

【０２１４】

図２４の横軸は、水平方向または垂直方向の一方についての１次元方向の縮小倍率ｎであり、また、縦軸は、検出される動きベクトルについての誤差（ベクトル誤差）を示している。図２４のベクトル誤差の数値は画素数で表されている。

【０２１５】

図２４において、曲線３０１は、縮小倍率に対するベクトル誤差の平均値である。また、曲線３０２は、縮小倍率に対するベクトル誤差の分散σの３倍値（３σ（９９．７％））を示している。曲線３０３は、曲線３０２の近似曲線を示している。

【０２１６】

図２４は、１次元方向の縮小倍率ｎに対するベクトル誤差を示しているが、ＳＡＤテーブルは２次元のため、図２４に示されるものの２乗の割合でテーブルサイズ（テーブル要素数）が削減されるのに対し、ベクトル誤差は、線形程度にしか増加しないことから、この実施形態による手法の有用性が分かる。

【０２１７】

また、ｎ＝６４（縮小率１／６４）倍の縮小倍率でも、ベクトル誤差は小さく、全く異なる動きベクトルを検出出力とするような破綻は見られないことから、実質、１／４０９６に、ＳＡＤテーブルのサイズを削減可能であると言える。

【０２１８】

また、前述したように、動画の手ぶれ補正においては、リアルタイム性とシステム遅延の削減が強く求められるのに対し、精度については、破綻した全く異なる動きベクトルが検出される場合を除き、ある程度のベクトル検出誤差に対して寛容である。したがって、破綻しないままＳＡＤテーブルのサイズを大きく削減することができる、この実施形態は有用性が高いと言える。

【０２１９】

以上説明した実施形態の画像処理方法は、従来手法として説明した特許文献４に記載された画像を縮小変換したサイズで動きベクトルを検出する手法に比べて、次に挙げる２つの点において、大きく異なるメリットを有するものである。

【０２２０】

まず、第一に、この実施形態による手法は、特許文献４に記載された従来手法と異なり、画像を縮小変換するプロセスを全く必要としない。この実施形態による手法においては、参照ブロックについて算出したＳＡＤ値を、ＳＡＤテーブル（縮小ＳＡＤテーブル）に代入加算する際に、同時に縮小倍率に相当するアドレス変換を行なうからである。

【０２２１】

これにより、この実施形態による手法においては、特許文献４に記載された従来手法のような画像の縮小変換のためのロジックも、縮小した画像をメモリに格納する時間およびバンド幅の浪費も、縮小画像をメモリに貼る領域確保も必要ない、というメリットを有する。

【０２２２】

特許文献４に記載された従来手法のもう１つ重要な問題点として、前述も使用にしたように、画像を縮小変換する際のエイリアシング（折り返し歪み）や、低照度ノイズ除去のためのローパスフィルタの存在の問題がある。すなわち、画像縮小する際には、適切なローパスフィルタを通してからリサンプリングを行なわなければならず、さもないと、不要なエイリアシングが発生し、その縮小画像を用いた動きベクトルの精度が著しく損なわれるからである。

【０２２３】

縮小変換の際の理想的なローパスフィルタの特性としては、ｓｉｎｃ関数に類似した関数であることが、理論的に証明されている。ｓｉｎｃ関数自体は、ｓｉｎ（ｘπ）／（ｘπ）の形で表されるカットオフ周波数ｆ／2の無限タップのＦＩＲ（Ｆｉｎｉｔｅ　Ｉｍ

ｐｕｌｓｅ　Ｒｅｓｐｏｎｓｅ）フィルタであるが、縮小倍率１／nのときの理想的なカットオフ周波数ｆ／（２ｎ）のローパスフィルタとしては、ｓｉｎ（ｘπ/ｎ）／（ｘπ/ｎ）と表される。しかし、これもｓｉｎｃ関数の一形態として良い。

【０２２４】

図２５～図２７の上側には、それぞれ縮小倍率が１／２倍、１／４倍、１／８倍のときのｓｉｎｃ関数（ローパスフィルタの理想特性）の形状を示す。この図２５～図２７から、縮小倍率が大きくなればなる程、関数がタップ軸方向に拡大して行くことが分かる。つまり、無限タップのｓｉｎｃ関数を主要な係数のみで近似する場合にも、ＦＩＲフィルタのタップ数を増加させなければならないと言える。

【０２２５】

また、一般的に、より低い帯域のカットオフ周波数を実現するフィルタは、フィルタ形状よりもタップ数が、その性能に対して支配的になって行くことが知られている。

【０２２６】

したがって、特許文献４に記載の従来手法の縮小画像を用いる動きベクトル演算手法の場合、画像の縮小倍率が大きくなればなる程、そのＳＡＤテーブル削減効果が大きいにも関わらず、画像生成する際の前処理用フィルタとしてのローパスフィルタは、縮小倍率が大きくなればなる程、コストが増加してしまう、という矛盾を併せ持つのである。

【０２２７】

一般に、高次タップのＦＩＲフィルタを実現する場合、演算ロジックのコストがタップ数の２乗に比例して増加するため、問題となるが、より大きい問題は、垂直フィルタ実現のためのラインメモリ数の増加である。近年のデジタルスチルカメラにおいては、画素数向上に伴うラインメモリのサイズ削減のため、いわゆる短冊処理を行なっているが、例え、１ライン当たりのサイズを削減したとしても、ラインメモリそのものの本数が増加することは、物理レイアウトエリアで換算されるトータルコストを著しく押し上げる。

【０２２８】

以上、述べたように、特許文献４に記載の従来手法の画像縮小によるアプローチは、特に垂直ローパスフィルタの実現において、大きな壁が立ちはだかっていることが分かる。それに対し、この発明の手法は、全く異なる形で簡潔にこの問題を解決している。

【０２２９】

図２５～図２７の下側に、この実施の形態で用いる新規なブロックマッチング手法におけるローパスフィルタのイメージを示す。この実施の形態で用いる新規なブロックマッチング手法においては、画像縮小処理を伴っていないが、縮小ＳＡＤテーブルの生成演算過程におけるローパスフィルタのイメージを図示したものである。

【０２３０】

図２５～図２７の下側に示されるように、このローパスフィルタの特性は、ｓｉｎｃ関数の主要係数部分を線形で近似した、シンプルなフィルタ特性ではあるものの、縮小倍率に連動してタップ数が増加していることが分かる。これは、先に述べた、カットオフ周波数が低くなる程、ローパスフィルタの性能はタップ数が支配的になる、という事実に好適である。つまり、実施形態の線形加重分散加算を行なう処理のような、この発明におけるＳＡＤ値の分散加算を行なう処理そのものが、倍率連動の高性能ローパスフィルタを、シンプルな回路で実現していることと等価なのである。

【０２３１】

このローパスフィルタに絡んで、他にもメリットがある。特許文献４記載の従来手法では、ローパスフィルタをかけた後、リサンプリングすることで画像を縮小するが、この時点で相当数の画像情報が失われる。つまり、ローパスフィルタの演算において、画像情報の輝度値の語長は大幅に丸められてメモリに格納され、殆どの画素情報の下位ビットは、縮小後の画像に影響を与えないのである。

【０２３２】

一方、この実施の形態で用いる新規なブロックマッチング手法においては、全ての画素の輝度値の全ビット情報を、遍く平等に使用してＳＡＤ値を演算し、その分散加算値を求

めて縮小ＳＡＤテーブルに加算する。縮小ＳＡＤテーブルの各テーブル要素値の語長さえ増やせば、最終的なＳＡＤ値の出力まで、一切の丸め誤差を含まない形で演算可能である。縮小ＳＡＤテーブルの面積はフレームメモリに比較して小さいため、縮小ＳＡＤテーブルの語長拡張は大きな問題にならない。その結果として、縮小ＳＡＤテーブル並びに動きベクトル検出を、高精度に実現できるのである。

【０２３３】

以上の説明は、注目画面である参照フレーム１０２について、一つのターゲットブロックを考え、当該ターゲットブロックを中心としたサーチ範囲を設定して、そのサーチ範囲における動きベクトルを検出する場合の説明であるが、実際の手ぶれ補正システムにおいては、元フレーム１０１を複数の領域に分割し、それぞれの分割領域において、それぞれターゲットブロック設定する。そして、参照フレーム１０２では、それぞれのターゲットブロックの射影イメージを中心としてサーチ範囲を設定して、それぞれのサーチ範囲において動きベクトル２０５を検出するようにする。

【０２３４】

そして、ＣＣＤイメージャを撮像素子として使用する手ぶれ補正システムにおいては、参照フレームにおいて取得したそれらの複数個のターゲットブロックについて得られた複数個の動きベクトルを用いて、過去のフレームにおけるそれらの動きベクトルからの推移も加味しながら統計的に処理することで、１フレームについて１つのグローバルベクトル、即ち、フレームの手ぶれベクトルを確定するようにする。

【０２３５】

これに対して、この実施の形態では、一つのフレームについて、複数個のターゲットブロックおよびサーチ範囲を設定し、それらの複数個のターゲットブロックについて得られた複数個の動きベクトルを用いて、前述した分割画像区間のそれぞれについての、手ぶれ速度ベクトルを算出するようにする。

【０２３６】

すなわち、この実施の形態では、図２８に示すように、１フレームの画像領域を、前述した分割画像区間に対応して、垂直方向に８分割すると共に、この例では、水平方向にも８分割して、合計６４個の分割領域にする。

【０２３７】

そして、図２８に示すように、６４個の分割領域のそれぞれにおいて検出したい６４個の動きベクトル２０５の基準位置ＰＯ１～ＰＯ６４のそれぞれを中心とするサーチ範囲ＳＲ１，ＳＲ２，・・・，ＳＲ６４を定め、各サーチ範囲において、ターゲットブロックの射影イメージブロックＩＢ１，ＩＢ２，・・・，ＩＢ６４を想定する。この場合、ターゲットブロックの射影イメージブロックＩＢ１，ＩＢ２，・・・，ＩＢ６４は、隣り合うもの同士で重なり合うことはないが、サーチ範囲ＳＲ１，ＳＲ２，・・・，ＳＲ６４は、図２８から明らかなように、隣り合うもの同士では、互いに領域的に重なり合うものとなってもよい。

【０２３８】

そして、この射影イメージブロックＩＢ１，ＩＢ２，・・・，ＩＢ６４と同じ大きさの参照ブロックを設定し、各サーチ範囲ＳＲ１，ＳＲ２，・・・，ＳＲ６４内を、設定した参照ブロックを移動させて、上述と同様にして、縮小ＳＡＤテーブルを生成し、各サーチ範囲ＳＲ１，ＳＲ２，・・・，ＳＲ６４のターゲットブロックＴＧ１～ＴＧ６４（ターゲットブロックＴＧ１～ＴＧ６４の図示は、省略）についての合計６４個の動きベクトル２０５Ｖ０１～２０５Ｖ６４を検出するようにする。

【０２３９】

そして、この実施の形態では、図２９に示すように、前述した分割画像領域Ｐｄｉｖ＿０～Ｐｄｉｖ＿７のそれぞれに対応する同じ垂直方向の領域に含まれる８個のターゲットブロックについての８個の動き手ぶれベクトルを平均化することにより、それぞれの分割画像領域Ｐｄｉｖ＿０～Ｐｄｉｖ＿７についての手ぶれ動きベクトルＶＣｄｉｖ＿０～ＶＣｄｉｖ＿７を算出するようにする。

【０２４０】
すなわち、検出された動きベクトル２０５Ｖ１～２０５Ｖ８について、平均化処理をすることにより、分割画像区間Ｐｄｉｖ＿０についての手ぶれ動きベクトルＶＣｄｉｖ＿０を算出し、検出された動きベクトル２０５Ｖ９～２０５Ｖ１６について、平均化処理をすることにより、分割画像区間Ｐｄｉｖ＿１についての手ぶれ動きベクトルＶＣｄｉｖ＿１を算出し、検出された動きベクトル２０５Ｖ１７～２０５Ｖ２４について、平均化処理をすることにより、分割画像区間Ｐｄｉｖ＿２についての手ぶれ動きベクトルＶＣｄｉｖ＿２を算出し、検出された動きベクトル２０５Ｖ２５～２０５Ｖ３２について、平均化処理をすることにより、分割画像区間Ｐｄｉｖ＿３についての手ぶれ動きベクトルＶＣｄｉｖ＿３を算出し、検出された動きベクトル２０５Ｖ３３～２０５Ｖ４０について、平均化処理をすることにより、分割画像区間Ｐｄｉｖ＿４についての手ぶれ動きベクトルＶＣｄｉｖ＿４を算出し、検出された動きベクトル２０５Ｖ４１～２０５Ｖ４８について、平均化処理をすることにより、分割画像区間Ｐｄｉｖ＿５についての手ぶれ動きベクトルＶＣｄｉｖ＿５を算出し、検出された動きベクトル２０５Ｖ４９～２０５Ｖ５６について、平均化処理をすることにより、分割画像区間Ｐｄｉｖ＿６についての手ぶれ動きベクトルＶＣｄｉｖ＿６を算出し、検出された動きベクトル２０５Ｖ５７～２０５Ｖ６４について、平均化処理をすることにより、分割画像区間Ｐｄｉｖ＿７についての手ぶれ動きベクトルＶＣｄｉｖ＿７を算出する。

【０２４１】
ここで、平均化処理をする際には、各分割画像区間に含まれる８個の動きベクトル２０５Ｖｉ～２０５Ｖ（ｉ＋７）（ただし、ｉ＝１、９、１７、２５、３３、４１、４９、５７である）について、例外的な動きベクトルがあれば、当該例外的な動きベクトルは平均化処理から除去するようにする。

【０２４２】
例外的な動きベクトルであるかどうかの判別は、この実施の形態では、各動きベクトルについて、それと水平方向および垂直方向に隣接するターゲットブロックについて検出された動きベクトルと比較して、その比較結果のベクトル差が、予め定めたしきい値よりも大きいか否かにより行い、前記比較結果のベクトル差が、予め定めたしきい値よりも大きいと判別したときには、その動きベクトルは平均化処理から除去する動きベクトルであると判別するようにする。

【０２４３】
なお、例外的な動きベクトルであるかどうかの判別は、上述のような判別方法に限られるものではない。例えば、一つの分割画像区間の動きベクトルについての例外的な動きベクトルの判別に際し、当該分割画像区間およびその上下に（垂直方向に）隣接する分割画像区間のすべての動きベクトルの平均値および分散を求め、求めた平均値と、判別対象の動きベクトルとの差が、求めた分散の範囲内であるか否かにより判別するようにしても良い。

【０２４４】
そして、この実施の形態では、分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖのそれぞれの手ぶれ速度ベクトルＶｅｃ＿０～Ｖｅｃ＿７は、自分割画像区間の手ぶれ動きベクトルと、一つ後の隣接する分割画像区間の手ぶれ動きベクトルとの差分ベクトル（変化分）として検出する。すなわち、

Ｖｅｃ＿０＝ＶＣｄｉｖ＿１－ＶＣｄｉｖ＿０
Ｖｅｃ＿１＝ＶＣｄｉｖ＿２－ＶＣｄｉｖ＿１
Ｖｅｃ＿２＝ＶＣｄｉｖ＿３－ＶＣｄｉｖ＿２
Ｖｅｃ＿３＝ＶＣｄｉｖ＿４－ＶＣｄｉｖ＿３
Ｖｅｃ＿４＝ＶＣｄｉｖ＿５－ＶＣｄｉｖ＿４
Ｖｅｃ＿５＝ＶＣｄｉｖ＿６－ＶＣｄｉｖ＿５
Ｖｅｃ＿６＝ＶＣｄｉｖ＿７－ＶＣｄｉｖ＿６
・・・（式３）

として手ぶれ速度ベクトルＶｅｃ＿０～Ｖｅｃ＿６を算出する。

【０２４５】

１フレームの画像の垂直方向の最後の分割画像区間Ｐｄｉｖ＿７の後の垂直方向の分割画像区間は存在しないので、当該分割画像区間Ｐｄｉｖ＿７についての手ぶれ速度ベクトルは、上述のような差分演算では算出することができない。

【０２４６】

この実施の形態では、この分割画像区間Ｐｄｉｖ＿７についての手ぶれ動きベクトルは、当該分割画像区間Ｐｄｉｖ＿７よりも垂直方向に前の１～複数の分割画像区間の手ぶれ速度ベクトルから予測したものを用いるようにする。なお、垂直方向に一つ前の分割画像区間の手ぶれ速度ベクトルを、当該分割画像区間Ｐｄｉｖ＿７の手ぶれ速度ベクトルとして用いるようにしても良い。

【０２４７】

なお、求めた各分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７についての手ぶれ速度ベクトルＶｅｃ＿０～Ｖｅｃ＿７の逆符号成分が、手ぶれ補正およびフォーカルプレーン現象による画像歪みを補正に用いられることは前述したとおりである。

【０２４８】

そして、この実施の形態では、１フレームの垂直方向の先頭の分割画像区間Ｐｄｉｖ＿０についての手ぶれ動きベクトルＶＣｄｉｖ＿０は、当該１フレームの手ぶれベクトルとする。そして、当該フレームの画像データの画像メモリからの読み出しを、前フレームの画像データに対して、この分割画像区間Ｐｄｉｖ＿０についての手ぶれ動きベクトルＶＣｄｉｖ＿０に基づいて算出された変位量だけシフトした画像位置（対応するメモリアドレス位置）から開始するようにする。

【０２４９】

さらに、この実施の形態では、各分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７についての手ぶれ速度ベクトルＶｅｃ＿０～Ｖｅｃ＿７の逆符号成分を時間積分することにより算出した画像歪み変位量分だけ、画像メモリからの各画像データの水平方向および垂直方向の読み出し位置をシフトするようにすることにより、手ぶれおよびフォーカルプレーン現象に基づく画像歪みをキャンセルするように補正する。

【０２５０】

［この発明による画像信号の歪み補正装置の第１の実施形態］

以上のような特徴を備える画像信号の歪み補正装置の第１の実施の形態として、撮像装置の場合を例にとって、図を参照しながら説明する。図３０は、この発明の画像信号の歪み補正装置の実施形態としての撮像装置の一例のブロック図を示すものである。

【０２５１】

図３０に示すように、この実施形態の撮像装置は、システムバス２にＣＰＵ（Ｃｅｎｔｒａｌ　Ｐｒｏｃｅｓｓｉｎｇ　Ｕｎｉｔ）１が接続されると共に、システムバス２に、撮像信号処理系１０や、ユーザ操作入力部３、画像メモリ部４、記録再生装置部５などが接続されて構成されている。なお、この明細書においては、ＣＰＵ１は、種々のソフトウエア処理を行なうプログラムを記憶するＲＯＭ（Ｒｅａｄ　Ｏｎｌｙ　Ｍｅｍｏｒｙ）やワークエリア用ＲＡＭなどを含むものとしている。

【０２５２】

ユーザ操作入力部３を通じた撮像記録開始操作を受けて、図３０の撮像装置は、後述するような撮像画像データの記録処理を行なう。また、ユーザ操作入力部３を通じた撮像記録画像の再生開始操作を受けて、図３０の撮像装置は、記録再生装置部５の記録媒体に記録された撮像画像データの再生処理を行なう。

【０２５３】

そして、この実施の形態では、ユーザ操作入力部３を通じて、ユーザからの画像拡大・縮小指示入力や、解像度指示入力、また、ズーム倍率の値の指示入力が制御部としてのＣＰＵ１に与えられる。ＣＰＵ１は、それらの指示入力に応じた制御信号を形成し、タイミング信号発生部１２や各処理部に供給する。

【０２５４】
図３０に示すように、撮像レンズ１０Ｌを備えるカメラ光学系（図示は省略）を通じた被写体からの入射光は、撮像素子１１に照射されて撮像される。この例では、撮像素子１１は、ＣＭＯＳ（Ｃｏｍｐｌｅｍｅｎｔａｒｙ Ｍｅｔａｌ Ｏｘｉｄｅ Ｓｅｍｉｃｏｎｄｕｃｔｏｒ）イメージャで構成されている。
【０２５５】
この例の撮像装置においては、撮像記録開始操作がなされると、撮像素子１１からは、タイミング信号発生部１２からのタイミング信号によりサンプリングされることにより、赤（Ｒ）、緑（Ｇ）、青（Ｂ）の３原色から構成されるベイヤー配列のＲＡＷ（Ｒｅａｄ Ａｆｔｅｒ Ｗｒｉｔｅ）信号であるアナログ撮像信号が出力される。
【０２５６】
この例では、撮像素子１１からは、タイミング信号発生部１２からのタイミング信号により１ライン単位が同時に取り出される（サンプリングされる）ことによりアナログ撮像信号とされて出力される。
【０２５７】
出力されたアナログ撮像信号は、前処理部１３に供給され、欠陥補正やγ補正等の前処理が施され、データ変換部１４に供給される。
【０２５８】
データ変換部１４は、これに入力されたアナログ撮像信号から、輝度信号成分Ｙと、色差信号成分Ｃｂ／Ｃｒとにより構成されるデジタル撮像信号（ＹＣデータ）に変換し、そのデジタル撮像信号をシステムバスを介して、画像メモリ部４に供給する。
【０２５９】
画像メモリ部４は、この図３０の例においては、２個のフレームメモリ４１，４２からなり、データ変換部１４からのデジタル撮像信号は、先ず、フレームメモリ４１に格納される。そして、１フレーム経過すると、フレームメモリ４１に記憶されているデジタル撮像信号が、フレームメモリ４２に転送されると共に、フレームメモリ４１には、データ変換部１４からの新たなフレームのデジタル撮像信号が書き込まれる。したがって、フレームメモリ４２には、フレームメモリ４１に格納されているフレーム画像よりも１フレーム分前のフレーム画像が格納されている。
【０２６０】
そして、手ぶれ動きベクトル検出部１５は、システムバス２を介して、これら２個のフレームメモリ４１およびフレームメモリ４２をアクセスして、その格納データを読み出し、前述したような分割画像区間毎の動きベクトル検出処理を実行する。この場合、フレームメモリ４２に格納されているフレーム画像は、元フレームの画像とされ、また、フレームメモリ４１に格納されているフレーム画像は、参照フレームの画像とされる。
【０２６１】
そして、手ぶれ動きベクトル検出部１５は、その検出結果である前述したような分割画像区間毎の動きベクトルを、その後段の歪み補正解像度変換部１６に制御信号として伝達する。
【０２６２】
歪み補正解像度変換部１６は、手ぶれ動きベクトル検出部１５から受け取った分割画像区間毎の動きベクトルから手ぶれ速度ベクトルを算出し、算出した手ぶれ速度ベクトルに基づいて画像歪み補正量を算出する。そして、算出した画像歪み補正量にしたがって、フレームメモリ４２に格納されている遅延フレームの画像データを切り出しながら、必要な解像度および画像サイズに変換する処理をする。このフレームメモリ４２からの算出した画像歪み補正量にしたがった切り出しにより、変換後の画像は、手ぶれおよびフォーカルプレーン現象に基づく画像歪みが除去された画像となる。
【０２６３】
この歪み補正解像度変換部１６からの手ぶれおよびフォーカルプレーン現象に基づく画像歪みが除去された画像データは、ＮＴＳＣ（Ｎａｔｉｏｎａｌ Ｔｅｌｅｖｉｓｉｏｎ

Ｓｙｓｔｅｍ Ｃｏｍｍｉｔｔｅｅ）エンコーダ１８によりＮＴＳＣ方式の標準カラー映像信号に変換され、電子式ビューファインダーを構成するモニターディスプレイ６に供給され、撮影時の画像がその表示画面にモニター表示される。

【０２６４】

このモニター表示と並行して、歪み補正解像度変換部１６からの手ぶれが除去された画像データはコーデック部１７で記録変調などのコーディング処理された後、記録再生装置部５に供給されて、ＤＶＤ（Ｄｉｇｉｔａｌ Ｖｅｒｓａｔｉｌｅ Ｄｉｓｃ）などの光ディスクやハードディスクなどの記録媒体に記録される。

【０２６５】

この記録再生装置部５の記録媒体に記録された撮像画像データは、ユーザ操作入力部３を通じた再生開始操作に応じて読み出され、コーデック部１７に供給されて、再生デコードされる。そして、再生デコードされた画像データはＮＴＳＣエンコーダ１８を通じてモニターディスプレイ６に供給され、再生画像がその表示画面に表示される。なお、図３０では、図示を省略したが、ＮＴＳＣエンコーダ１８からの出力映像信号は、映像出力端子を通じて外部に導出することが可能とされている。

【０２６６】

上述した手ぶれ動きベクトル検出部１５は、ハードウエアにより構成することできるし、また、ＤＳＰ（Ｄｉｇｉｔａｌ Ｓｉｇｎａｌ Ｐｒｏｃｅｓｓｏｒ）を用いて構成することもできる。さらには、ＣＰＵ１によりソフトウエア処理とすることもできる。同様に、歪み補正解像度変換部１６も、ハードウエアにより構成することできるし、また、ＤＳＰ（Ｄｉｇｉｔａｌ Ｓｉｇｎａｌ Ｐｒｏｃｅｓｓｏｒ）を用いて構成することもできる。さらには、ＣＰＵ１によりソフトウエア処理とすることもできる。

【０２６７】

［手ぶれ動きベクトル検出部１５における処理動作］

＜第１の例＞

この手ぶれ動きベクトル検出部１５における処理動作の第１の例の流れを、図３１および図３２のフローチャートを参照して、以下に説明する。

【０２６８】

先ず、前述の図３あるいは図２８に示したようなサーチ範囲１０５あるいはサーチ範囲ＳＲ１～ＳＲ６４のそれぞれ内の１つの参照ブロックＩｉに対応する参照ベクトル（ｖｘ、ｖｙ）を指定する（ステップＳ１０１）。前述したように、（ｖｘ，ｖｙ）は、ターゲットブロックのフレーム上の位置（サーチ範囲の中心位置である）を基準位置（０，０）としたときに、指定された参照ベクトルにより示される位置を示し、ｖｘは指定された参照ベクトルによる、基準位置からの水平方向のずれ量成分であり、また、ｖｙは指定された参照ベクトルによる、基準位置からの垂直方向成分のずれ量成分である。そして、前述の従来例で述べたのと同様に、ずれ量ｖｘ、ｖｙは、画素を単位とした値とされている。

【０２６９】

ここで、サーチ範囲の中心位置を前記基準位置（０，０）とし、サーチ範囲を、水平方向には±Ｒｘ、垂直方向には±Ｒｙとしたとき、

$$-Rx \leqq vx \leqq +Rx,\quad -Ry \leqq vy \leqq +Ry$$

とされるものである。

【０２７０】

次に、ターゲットブロックＩｏ内の１つの画素の座標（ｘ，ｙ）を指定する（ステップＳ１０２）。次に、ターゲットブロックＩｏ内の指定された１つの座標（ｘ，ｙ）の画素値Ｉｏ（ｘ，ｙ）と、参照ブロックＩｉ内の対応する画素位置の画素値Ｉｉ（ｘ＋ｖｘ，ｙ＋ｖｙ）との差分絶対値αを、前述した（式１）に示したようにして算出する（ステップＳ１０３）。

そして、算出した差分絶対値αを、当該参照ブロックＩｉの参照ベクトル（ｖｘ，ｖｙ）が指し示すアドレス（テーブル要素）の、それまでのＳＡＤ値に加算し、その加算であるＳＡＤ値を、当該アドレスに書き戻すようにする（ステップＳ１０４）。すなわち、参

照ベクトル（ｖｘ，ｖｙ）に対応するＳＡＤ値を、ＳＡＤ（ｖｘ，ｖｙ）と表したとき、これを、前述した（式２）、すなわち、

ＳＡＤ（ｖｘ，ｖｙ）＝Σα＝Σ｜Ｉｏ（ｘ，ｙ）－Ｉｉ（ｘ＋ｖｘ，ｙ＋ｖｙ）｜
・・・（式２）

として算出し、当該参照ベクトル（ｖｘ，ｖｙ）が指し示すアドレスに書き込むようにする。

【０２７１】

次に、ターゲットブロックＩｏ内の全ての座標（ｘ，ｙ）の画素について、上記のステップＳ１０２～ステップＳ１０４の演算を行なったか否かを判別し（ステップＳ１０５）、ターゲットブロックＩｏ内の全ての座標（ｘ，ｙ）の画素については、未だ、その演算は終了していないと判別したときには、ステップＳ１０２に戻り、ターゲットブロックＩｏ内の次の座標（ｘ，ｙ）の画素位置を指定し、このステップＳ１０２以降の処理を繰り返す。

【０２７２】

以上のステップＳ１０１～ステップＳ１０５までの処理は、図５に示したフローチャートのステップＳ１～ステップＳ５と全く同様である。

【０２７３】

この実施形態では、ステップＳ１０５で、ターゲットブロックＩｏ内の全ての座標（ｘ，ｙ）の画素について、上記の演算を行なったと判別したときには、縮小倍率を１／ｎとして、参照ベクトル（ｖｘ，ｖｙ）を１／ｎに縮小した参照縮小ベクトル（ｖｘ／ｎ，ｖｙ／ｎ）を算出する（ステップＳ１０６）。

【０２７４】

次いで、参照縮小ベクトル（ｖｘ／ｎ，ｖｙ／ｎ）の近傍の複数の参照ベクトル、この例では、上述したように４個の近傍参照ベクトルを検知する（ステップＳ１０７）。そして、検知した４個の近傍参照ベクトルのそれぞれに対応するテーブル要素として分散加算すべき値を、前述したように、参照縮小ベクトルと近傍参照ベクトルとがそれぞれ示す位置の関係に基いて、ステップＳ１０４で求めたＳＡＤ値から、線形加重分散値として求める（ステップＳ１０８）。そして、求めた４個の線形加重分散値を、近傍参照ベクトルのそれぞれに対応するＳＡＤテーブル要素値に加算する（ステップＳ１０９）。

【０２７５】

このステップＳ１０９が終了すると、注目中の参照ブロックについてのＳＡＤ値の算出が終了したと判別して、サーチ範囲内の全ての参照ブロック、すなわち、全ての参照ベクトル（ｖｘ，ｖｙ）についての上記のステップＳ１０１からステップＳ１０９までの演算処理を完了したか否か判別する（図３２のステップＳ１１１）。

【０２７６】

ステップＳ１１１で、未だ、上記の演算処理を完了していない参照ベクトル（ｖｘ，ｖｙ）があると判別すると、ステップＳ１０１に戻り、上記の演算処理を完了していない次の参照ベクトル（ｖｘ，ｖｙ）を設定して、このステップＳ１０１以降の処理を繰り返す。

【０２７７】

そして、ステップＳ１１１で、上記の演算処理を完了していない参照ベクトル（ｖｘ，ｖｙ）はサーチ範囲内になくなったと判別すると、縮小ＳＡＤテーブルが完成したとして、当該完成した縮小ＳＡＤテーブルにおいて、最小値となっているＳＡＤ値を検出する（ステップＳ１１２）。

【０２７８】

次に、当該最小値となっているテーブル要素アドレス（ｍｘ，ｍｙ）のＳＡＤ値（最小値）と、その近傍の複数個、この例では、上述したように１５個の近傍テーブル要素のＳＡＤ値を用いて２次曲面を生成し（ステップＳ１１３）、その２次曲面の最小値のＳＡＤ値が対応する小数精度の位置を示す最小値ベクトル（ｐｘ，ｐｙ）を算出する（ステップＳ１１４）。この最小値ベクトル（ｐｘ，ｐｙ）は、小数精度の最小テーブル要素アドレ

スに対応している。

【０２７９】

そして、算出した小数精度の位置を示す最小値ベクトル（ｐｘ，ｐｙ）をｎ倍することにより、求めるべく動きベクトル（ｐｘ×ｎ，ｐｙ×ｎ）を算出する（ステップＳ１１５）。

【０２８０】

以上で、１つのターゲットブロックに対する、この実施形態におけるブロックマッチングによる動きベクトルの検出処理は、終了となる。図２８に示したような、１フレームについて分割した領域において、複数個の動きベクトルを検出する場合には、サーチ範囲および縮小倍率１／ｎを再設定して、上述の図３１および図３２に示した処理を、各分割領域のターゲットブロックについて繰り返すものである。

【０２８１】

なお、小数精度の位置を示す最小値ベクトル（ｐｘ，ｐｙ）を算出する方法としては、前述した水平方向および垂直方向の３次曲線を用いる方法を用いても良いことは言うまでもない。

【０２８２】

＜第２の例＞

上述の第１の例においては、１つの参照ブロック（参照ベクトル）について、そのＳＡＤ値を求めた後、そのＳＡＤ値から、参照縮小ベクトルの近傍の複数参照ベクトルについての分散加算値を求め、分散加算処理を行なうようにした。

【０２８３】

これに対して、この第２の例においては、参照ブロック内の各画素の、ターゲットブロックの画素との差分を検出したときに、その差分値から、参照縮小ベクトルの近傍の複数参照ベクトルについての分散加算値（ＳＡＤ値ではなく差分値）を求め、求めた差分値を分散加算処理するようにする。この第２の例によれば、１つの参照ブロック内のすべての画素についての差分演算を終了したときには、縮小ＳＡＤテーブルが生成されることになる。

【０２８４】

図３３および図３４は、この第２の例による動きベクトル検出処理のフローチャートを示すものである。

【０２８５】

図３３のステップＳ１２１～ステップＳ１２３までの処理は、図３１のステップＳ１０１～ステップＳ１０３までの処理と全く同様であるので、ここでは、その詳細な説明は省略する。

【０２８６】

この第２の例においては、ステップＳ１２３で、座標（ｘ，ｙ）の画素についての参照ブロックとターゲットブロック間での差分値αが算出すると、次には、縮小倍率を１／ｎとして、参照ベクトル（ｖｘ，ｖｙ）を１／ｎに縮小した参照縮小ベクトル（ｖｘ／ｎ，ｖｙ／ｎ）を算出する（ステップＳ１２４）。

【０２８７】

次に、参照縮小ベクトル（ｖｘ／ｎ，ｖｙ／ｎ）の近傍の複数の参照ベクトル、この例では、上述したように４個の近傍参照ベクトルを検知する（ステップＳ１２５）。そして、検知した４個の近傍参照ベクトルのそれぞれに対応するテーブル要素として分散加算すべき差分値を、前述したように、ステップＳ１２３で求めた差分値αから、参照縮小ベクトルと近傍参照ベクトルとがそれぞれ示す位置の関係に基いて、線形加重分散値（差分値）として求める（ステップＳ１２６）。

【０２８８】

そして、求めた４個の線形加重分散値を、近傍参照ベクトルのそれぞれに対応するテーブル要素値に加算する（ステップＳ１２７）。

【０２８９】

このステップＳ１２７が終了したら、ターゲットブロックＩｏ内の全ての座標（ｘ，ｙ）の画素について、上記のステップＳ１２２～ステップＳ１２７の演算を行なったか否かを判別し（ステップＳ１２８）、ターゲットブロックＩｏ内の全ての座標（ｘ，ｙ）の画素については、未だ、その演算は終了していないと判別したときには、ステップＳ１２２に戻り、ターゲットブロックＩｏ内の次の座標（ｘ，ｙ）の画素位置を指定し、このステップＳ１２２以降の処理を繰り返す。

【０２９０】

ステップＳ１２８で、ターゲットブロックＩｏ内の全ての座標（ｘ，ｙ）の画素について、上記の演算を行なったと判別したときには、注目中の参照ブロックについてのＳＡＤ値の算出が終了したと判別して、サーチ範囲内の全ての参照ブロック、すなわち、全ての参照ベクトル（ｖｘ，ｖｙ）についての上記のステップＳ１２１からステップＳ１２８までの演算処理を完了したか否か判別する（図３４のステップＳ１３１）。

【０２９１】

ステップＳ１３１で、未だ、上記の演算処理を完了していない参照ベクトル（ｖｘ，ｖｙ）があると判別すると、ステップＳ１２１に戻り、上記の演算処理を完了していない次の参照ベクトル（ｖｘ，ｖｙ）を設定して、このステップＳ１２１以降の処理を繰り返す。

【０２９２】

そして、ステップＳ１３１で、上記の演算処理を完了していない参照ベクトル（ｖｘ，ｖｙ）はサーチ範囲内になくなったと判別すると、縮小ＳＡＤテーブルが完成したとして、当該完成した縮小ＳＡＤテーブルにおいて、最小値となっているＳＡＤ値を検出する（ステップＳ１３２）。

【０２９３】

次に、当該最小値となっているテーブル要素アドレス（ｍｘ，ｍｙ）のＳＡＤ値（最小値）と、その近傍の複数個、この例では、上述したように１５個の近傍テーブル要素のＳＡＤ値を用いて２次曲面を生成し（ステップＳ１３３）、その２次曲面の最小値のＳＡＤ値が対応する小数精度の位置を示す最小値ベクトル（ｐｘ，ｐｙ）を算出する（ステップＳ１３４）。この最小値ベクトル（ｐｘ，ｐｙ）は、小数精度の最小テーブル要素アドレスに対応している。

【０２９４】

そして、算出した小数精度の位置を示す最小値ベクトル（ｐｘ，ｐｙ）をｎ倍することにより、求めるべく動きベクトル（ｐｘ×ｎ，ｐｙ×ｎ）を算出する（ステップＳ１３５）。

【０２９５】

以上で、１つのターゲットブロックに対する、この第２の例におけるブロックマッチングによる動きベクトルの検出処理は、終了となる。図２８に示したような、１フレームについて分割した領域において、複数個の動きベクトルを検出する場合には、サーチ範囲および縮小倍率１／ｎを再設定して、上述の図３３および図３４に示した処理を、各分割領域について繰り返すものである。

【０２９６】

なお、この第２の例においても、小数精度の位置を示す最小値ベクトル（ｐｘ，ｐｙ）を算出する方法としては、前述した水平方向および垂直方向の３次曲線を用いる方法を用いても良いことは言うまでもない。

【０２９７】

＜第３の例＞

図２４に示したように、この実施形態による動きベクトルの検出手法を用いた場合には、参照ベクトルの縮小倍率が１／６４の場合でも、全く異なる動きベクトルを出力するような破綻は見られないことから、実質的に１／４０９６に、ＳＡＤテーブルを削減可能である。

【０２９８】

つまり、１／４０９６に削減した縮小ＳＡＤテーブルを用意しておき、例えば１回目の縮小倍率１／ｎａ＝１／６４で１回目の動きベクトルを検出する。次に、１回目で検出したその動きベクトルを中心にしてサーチ範囲を狭め、２回目の検出を、１回目の縮小倍率１／ｎａよりも小さい２回目の縮小倍率１／ｎｂ、例えば１／ｎｂ＝１／８で行なうようにすればよい。すなわち、１回目と２回目とで縮小倍率を変えて、１回目のベクトル誤差範囲内に収まるように、２回目の縮小倍率を設定すれば、かなりの高精度で、動きベクトル検出が可能である。

【０２９９】

この第３の例の場合における動きベクトル検出処理を、図３５～図３８のフローチャートを参照しながら説明する。

【０３００】

この図３５～図３８に示す第３の例は、基本的な動き検出処理として上述した第１の例を用いている。したがって、図３５のステップＳ１４１～ステップＳ１４９の処理ステップおよび図３６のステップＳ１５１～ステップＳ１５５までの処理ステップは、図３１のステップＳ１０１～ステップＳ１０９の処理ステップおよび図３２のステップＳ１１１～ステップＳ１１５までの処理ステップと全く同様である。

【０３０１】

この第３の例においては、図３６のステップＳ１５５で動きベクトルを算出したら、そこで処理を終了するのではなく、当該ステップＳ１５５で算出した動きベクトルは、１回目の動きベクトルとして、次のステップＳ１５６において、この１回目で算出した動きベクトルに基づき、同じ参照フレーム内で、サーチ範囲を絞り、また、参照ベクトルの縮小倍率を、１回目の縮小倍率１／ｎａよりも小さい縮小倍率１／ｎｂに変更する。

【０３０２】

すなわち、１回目の処理で、動きベクトルが算出されると、その算出された動きベクトルから、参照フレームと元フレームとの間で、相関のあるブロック範囲がおおよそ検出できる。そこで、その相関のあるブロック範囲を中心とした、絞ったサーチ範囲を設定することができる。そして、１回目よりも縮小倍率を小さくすることで、より誤差の少ない状態で、２回目の動きベクトルの算出が可能になると期待できる。

【０３０３】

こうして、ステップＳ１５６で、絞ったサーチ範囲を設定し、新たな縮小倍率を設定したら、１回目と全く同様にして、２回目の動きベクトルの検出処理を、ステップＳ１５７～ステップＳ１５８、図３７のステップＳ１６１～ステップＳ１６８、さらに、図３８のステップＳ１７１～ステップＳ１７４により実行する。これらのステップの処理は、図３１のステップＳ１０１～ステップＳ１０９の処理ステップおよび図３２のステップＳ１１１～ステップＳ１１５までの処理ステップと全く同様である。

【０３０４】

こうして、最終的に、ステップＳ１７４において、２回目の動きベクトルとして、目的とする動きベクトルが得られる。

【０３０５】

以上の例は、動きベクトルの検出方法として、前述した第１の例を用い、それを２段階、繰り返した場合であるが、サーチ範囲をさらに絞り、かつ、必要に応じて縮小倍率を変更しながら、２段階以上、繰り返すようにしても、勿論良い。

【０３０６】

また、動きベクトルの検出方法としては、前述した第１の例の代わりに、前述した第２の例を用いることができることは言うまでもない。また、小数精度の位置を示す最小値ベクトル（ｐｘ，ｐｙ）を算出する方法としては、前述した水平方向および垂直方向の３次曲線を用いる方法を用いても良いことは前述の例と同様である。

【０３０７】

［歪み補正解像度変換部１６の構成例および処理動作］

図３９は、この実施の形態における歪み補正解像度変換部１６の構成例を示すブロック

図である。

【０３０８】

この実施の形態における歪み補正解像度変換部１６は、信号処理部５０と、この信号処理部５０に接続される、画像メモリ６０と、初期座標算出部７０と、速度ベクトル処理部８０とからなる。

【０３０９】

信号処理部５０は、画像メモリ部４のフレームメモリ４２からの撮像画像データＤｉｎについて、この信号処理部５０に接続されている画像メモリ６０への書き込み／読み出しを制御しながら、手ぶれ補正などの補正処理や、撮像画像の電子ズーム処理（電子的拡大・縮小処理）を行なうとともに、指定された解像度の出力画像データＤｏｕｔを生成して出力する。

【０３１０】

信号処理部５０には、例えばタイミング信号発生部１２から、出力画像データの水平周期に同期する信号、すなわち、水平同期信号Ｈ－ＳＹＮＣと、出力画像データの垂直周期に同期する信号、すなわち、垂直同期信号Ｖ－ＳＹＮＣと、１画素周期のクロック信号ＣＬＫと、それらに同期するクロックが、処理タイミング信号Ｓｔとして供給される（図３０においては、これらの信号の図示は省略）。

【０３１１】

速度ベクトル処理部８０には、上述した手ぶれ動きベクトル検出部１５からの、この例では６４個の動きベクトル２０５Ｖ１～２０５Ｖ６４が供給される。この速度ベクトル処理部８０は、ハードウエアにより構成することもできるし、ＣＰＵ１によるソフトウエア処理により構成することも可能である。

【０３１２】

速度ベクトル処理部８０は、図３９に示すように、手ぶれ動きベクトル検出部１５からの動きベクトル２０５Ｖ１～２０５Ｖ６４から、８個の分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７のそれぞれについての手ぶれ速度ベクトルＶｅｃ＿０～Ｖｅｃ＿７を生成する速度ベクトル生成部８１と、この速度ベクトル生成部８１で生成された手ぶれ速度ベクトルＶｅｃ＿０～Ｖｅｃ＿７が供給される水平補正速度成分検出部８２および垂直補正速度成分検出部８３とからなる。

【０３１３】

図４０に、速度ベクトル生成部８１のハードウエア構成例を示す。図４０に示すように、速度ベクトル生成部８１は、８個の分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７のそれぞれに対応して設けられている平均化演算部８１０，８１１，８１２，８１３，８１４，８１５，８１６，８１７と、差分演算を行う減算部８２１，８２２，８２３，８２４，８２５，８２６，８２７と、手ぶれベクトル出力部８３０と、速度ベクトル出力部８４０とからなる。

【０３１４】

平均化演算部８１０～８１７のそれぞれは、手ぶれベクトル検出部１５からの、それぞれが対応する分割画像区間の８個の動きベクトル２０５Ｖｉ～２０５Ｖ（ｉ＋７）を取り込み、前述した例外除去処理を含む平均化処理を行って、各分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７についての手ぶれ動きベクトルＶＣｄｉｖ＿０～ＶＣｄｉｖ＿７を算出して出力する。

【０３１５】

そして、減算部８２１～８２７は、平均化演算部８１０～８１７からの手ぶれ動きベクトルＶＣｄｉｖ＿０～ＶＣｄｉｖ＿７を受けて、前述した（式３）に示した差分演算を行い、分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿６についての手ぶれ速度ベクトルＶｅｃ＿０～Ｖｅｃ＿６を算出し、速度ベクトル出力部８４０に供給する。

【０３１６】

速度ベクトル出力部８４０は、前述したようにして、垂直方向の最下部の分割画像区間Ｐｄｉｖ＿７についての手ぶれ速度ベクトルＶｅｃ＿７を、この例では、算出された他の

手ぶれ速度ベクトルから予測して求め、減算部８２１～８２７からの手ぶれ速度ベクトルＶｅｃ＿０～Ｖｅｃ＿６と共に、水平補正速度成分検出部８２および垂直補正速度成分検出部８３に出力する。

【０３１７】

図４１は、この速度ベクトル生成部８０をソフトウエアで構成した場合の処理のフローチャートを示すものである。なお、以下の図４１のフローチャートの説明においては、動きベクトル２０５Ｖｉ～２０５（ｉ＋７）は、ベクトルＶ（ｘ、ｙ）（ｘ＝１，２，・・・，８；ｙ＝１，２，・・・，８）と表す。

【０３１８】

まず、手ぶれベクトル検出部１５からの動きベクトルＶ（ｘ、ｙ）を取り込む（ステップＳ１８１）。次に、取り込んだ動きベクトル（ｘ、ｙ）を、当該取り込んだ動きベクトル（ｘ、ｙ）と同じ分割画像区間に含まれる他のベクトルＶ（０～７、ｙ）と比較する（ステップＳ１８２）。次に、取り込んだ動きベクトル（ｘ、ｙ）を、取り込んだ動きベクトル（ｘ、ｙ）に対し垂直方向に隣接する分割画像区間の他のベクトルＶ（０～７、ｙ±１）と比較する（ステップＳ１８３）。

【０３１９】

次に、ステップＳ１８２およびステップＳ１８３の比較結果のベクトル差が、すべて予め定められたしきい値以下であって、取り込んだ動きベクトルが例外的な動きベクトルでないと確認されたか否かを判別する（ステップＳ１８４）。

【０３２０】

ステップＳ１８４で、取り込んだ動きベクトルが例外的な動きベクトルであると判別したときには、取り込んだ動きベクトルは、平均化処理から除外する（ステップＳ１８５）。そして、ステップＳ１８１に戻る。

【０３２１】

また、ステップＳ１８４で、取り込んだ動きベクトルが例外的な動きベクトルではないと判別したときには、取り込んだ動きベクトルを、平均化処理に反映させることを決定する（ステップＳ１８６）。そして、１画面内のすべての動きベクトルについて、上記のステップＳ１８１～１８６までの処理が終了したか否かを判別する（ステップＳ１８７）。

【０３２２】

ステップＳ１８７で、１画面内のすべての動きベクトルについて、上記のステップＳ１８１～１８６までの処理が終了してはいないと判別したときには、ステップＳ１８１に戻る。

【０３２３】

また、ステップＳ１８７で、１画面内のすべての動きベクトルについて、上記のステップＳ１８１～１８６までの処理が終了したと判別したときには、平均化処理を実行する（ステップＳ１８８）。

【０３２４】

次に、平均化処理して得られた各分割画像区間ごとの動きベクトルの差分演算を行って、手ぶれ速度ベクトルを算出する。次に、すべての分割画像区間についての平均化処理を終了したか否かを判別する（ステップＳ１９０）。

【０３２５】

ステップＳ１９０で、すべての分割画像区間についての平均化処理を終了していないと判別したときには、ステップＳ１８８に戻り、このステップＳ１８８以降の処理を実行する。また、ステップＳ１９０で、すべての分割画像区間についての平均化処理を終了したと判別したときには、垂直方向の最後の分割画像区間Ｐｄｉｖ＿７についての動き速度ベクトルを前述したようにして予測して算出し（ステップＳ１９１）、この処理ルーチンを終了する。

【０３２６】

速度ベクトル出力部８４０は、以上のようにして生成した各分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７についての動き速度ベクトルＶＣｄｉｖ＿０～ＶＣｄｉｖ＿７を、前述し

たように、水平補正速度成分検出部８２および垂直補正速度成分検出部８３に出力する。
【０３２７】
水平補正速度成分検出部８２では、分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７のそれぞれにおける手ぶれ速度ベクトルＶｅｃ＿０～Ｖｅｃ＿７の水平方向成分を抽出し、その逆符号成分として、水平補正速度成分Ｘ＿ＳＴＢ＿０～Ｘ＿ＳＴＢ＿７を求める。そして、その算出した水平補正速度成分Ｘ＿ＳＴＢ＿０～Ｘ＿ＳＴＢ＿７を信号処理部５０に供給する。
【０３２８】
また、垂直補正速度成分検出部８３では、分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７のそれぞれにおける手ぶれ速度ベクトルＶｅｃ＿０～Ｖｅｃ＿７の垂直方向成分を抽出し、その逆符号成分として、垂直補正速度成分Ｙ＿ＳＴＢ＿０～Ｙ＿ＳＴＢ＿７を求める。そして、その算出した垂直補正速度成分Ｙ＿ＳＴＢ＿０～Ｙ＿ＳＴＢ＿７を信号処理部５０に供給する。
【０３２９】
また、分割画像区間Ｐｄｉｖ＿０についての平均化演算部８１０で算出された手ぶれ動きベクトルＶＣｄｉｖ＿０は、手ぶれベクトル出力部８３０を通じてそのまま出力されて、初期座標算出部７０に供給される。
【０３３０】
初期座標算出部７０は、処理対象のフレームの画像データの画像メモリからの読み出し位置の初期座標（ＳＸ、ＳＹ）を、手ぶれ動きベクトルＶＣｄｉｖ＿０から算出して、信号処理部５０に供給する。
【０３３１】
次に、信号処理部５０では、画像メモリ６０への画像データの書き込み／読み出しを制御しながら、後で詳述するようにして、速度ベクトル処理部８０からの水平補正速度成分と垂直補正速度成分とから、各水平ラインの水平補正変位量ＳＸ＿ＡＤＤおよび垂直補正変位量ＳＹ＿ＡＤＤを算出し、これら算出した水平補正変位量ＳＸ＿ＡＤＤおよび垂直補正変位量ＳＹ＿ＡＤＤを用いて、画像メモリ部４からの撮像画像データＤｉｎについての手ぶれによる画像歪みを補正処理し、出力画像データＤｏｕｔを得る。
【０３３２】
なお、前述したように、信号処理部５０では、電子ズーム（画像拡大・縮小）処理を行なうと共に、標準精細度、高精細度などに応じた解像度変換処理なども行なう。
【０３３３】
すなわち、手ぶれ補正処理や電子ズーム処理、出力データ生成処理のために、信号処理部５０は、水平処理ブロック５１と、垂直処理ブロック５２と、レジスタブロック５３とを含んでいる。そして、水平処理ブロック５１は、水平手ぶれ補正量積分部５１１と水平画像処理部５１２とを備え、また、垂直処理ブロック５２は、垂直手ぶれ補正量積分部５２１と垂直画像処理部５２２とを備える。
【０３３４】
水平処理ブロック５１は、入力される撮像画像データＤｉｎに対して水平方向の処理を施す。この水平方向の処理には、水平方向の手ぶれ補正処理を含む。垂直処理ブロック５２は、入力される撮像画像データＤｉｎに対して垂直方向の処理を施す。この垂直方向の処理には、垂直方向の手ぶれ補正処理を含む。また、レジスタブロック５３は、速度ベクトル処理部８０で検出された手ぶれ速度ベクトルの水平方向および垂直方向の情報を、水平処理ブロック５１および垂直処理ブロック５２に受け渡す処理を行なう。
【０３３５】
この場合に、この実施形態では、水平補正速度成分Ｘ＿ＳＴＢ＿０～Ｘ＿ＳＴＢ＿７および垂直補正速度成分Ｙ＿ＳＴＢ＿０～Ｙ＿ＳＴＢ＿７のそれぞれの値は、１水平ライン区間の時間長当たりの手ぶれ補正量として求められる。
【０３３６】
すなわち、水平補正速度成分Ｘ＿ＳＴＢ（Ｘ＿ＳＴＢ＿０～Ｘ＿ＳＴＢ＿７）は、

Ｘ＿ＳＴＢ＝（水平手ぶれ補正量／１水平ライン区間の時間長）
とされる。
【０３３７】
また、垂直補正速度成分Ｙ＿ＳＴＢ（Ｙ＿ＳＴＢ＿０～Ｙ＿ＳＴＢ＿７）
Ｙ＿ＳＴＢ＝（水平手ぶれ補正量／１水平ライン区間の時間長）
とされる。
【０３３８】
そして、水平補正速度成分Ｘ＿ＳＴＢ＿０～Ｘ＿ＳＴＢ＿７は、水平方向の画素ピッチ（画素の間隔）ｄｘの倍数値（小数点以下の値を取り得る。以下同じ）、換言すれば、水平方向の画素数（小数点以下を取り得る）で表わされる。また、垂直補正速度成分Ｙ＿ＳＴＢ＿０～Ｙ＿ＳＴＢ＿７は、垂直方向の画素ピッチ（画素の間隔）ｄｙの倍数値（小数点以下を取り得る）、換言すれば、垂直方向の画素数（小数点以下を取り得る。以下同じ）で表わされる。
【０３３９】
これは、速度ベクトル生成部８１からの手ぶれ速度出力の水平方向成分と、垂直方向成分のそれぞれに対して、「速度成分の値→画素数（小数点以下の値を取り得る。以下同じ）の値」の対応テーブルを用意し、速度成分の値を入力値として、画素数の値を得るようにすることで実現される。そして、得られた画素数の値について、手ぶれの速度成分の方向とは逆方向に対応する符号を付加して、それぞれ、水平補正速度成分Ｘ＿ＳＴＢ＿０～Ｘ＿ＳＴＢ＿７および垂直補正速度成分Ｙ＿ＳＴＢ＿０～Ｙ＿ＳＴＢ＿７とするものである。
【０３４０】
この実施形態では、このように水平補正速度成分Ｘ＿ＳＴＢ＿０～Ｘ＿ＳＴＢ＿７および垂直補正速度成分Ｙ＿ＳＴＢ＿０～Ｙ＿ＳＴＢ＿７の値を定めることにより、後述するライン単位の時間積分計算による水平手ぶれ補正量ＳＸ＿ＡＤＤおよび垂直手ぶれ補正量ＳＹ＿ＡＤＤを、水平補正速度成分および垂直補正速度成分の単純加算により行なえるようにしている。
【０３４１】
そして、これら水平補正速度成分検出部８２および垂直補正速度成分検出部８３で算出された水平補正速度成分Ｘ＿ＳＴＢ＿０～Ｘ＿ＳＴＢ＿７および垂直補正速度成分Ｙ＿ＳＴＢ＿０～Ｙ＿ＳＴＢ＿７は、例えばタイミング信号発生部１２からのタイミング信号Ｓｔに応じたタイミングで、信号処理部５０のレジスタブロック５３のＩＦ（インターフェース）レジスタ（図示は省略）に、順次取り込まれる。
【０３４２】
そして、取り込まれた水平補正速度成分Ｘ＿ＳＴＢ＿０～Ｘ＿ＳＴＢ＿７および垂直補正速度成分Ｙ＿ＳＴＢ＿０～Ｙ＿ＳＴＢ＿７は、水平処理ブロック５１および垂直処理ブロック５２での処理タイミングに対応して、レジスタブロック５３内の水平処理ブロック用レジスタおよび垂直処理ブロック用レジスタ（何れも図示は省略）に、前記ＩＦレジスタから、前記取り込まれたタイミングとは別のタイミングで、転送される。
【０３４３】
水平処理ブロック５１および垂直処理ブロック５２の水平手ぶれ補正量積分部５１１および垂直手ぶれ補正量積分部５２１は、レジスタブロック５３の水平処理ブロック用レジスタおよび垂直処理ブロック用レジスタに取り込まれている各分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７の水平補正速度成分Ｘ＿ＳＴＢ＿０～Ｘ＿ＳＴＢ＿７および垂直補正速度成分Ｙ＿ＳＴＢ＿０～Ｙ＿ＳＴＢ＿７を積分することにより、各分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７における各水平ラインに対する水平手ぶれ補正量ＳＸ＿ＡＤＤおよび垂直手ぶれ補正量ＳＹ＿ＡＤＤを算出する。
【０３４４】
水平処理ブロック５１の水平画像処理部５１２では、水平手ぶれ補正量積分部５１１で算出された水平手ぶれ補正量ＳＸ＿ＡＤＤを用いて、水平方向の手ぶれ成分を補正し、ま

た、垂直処理ブロック５２の垂直画像処理部５２２では、垂直手ぶれ補正量積分部５２１で算出された垂直手ぶれ補正量ＳＹ＿ＡＤＤを用いて、垂直方向の手ぶれ成分を補正する。この補正処理においては、水平処理ブロック５１の水平画像処理部５１２および垂直処理ブロック５２の垂直画像処理部５２２では、それぞれ画像メモリ６０に対して画像データの読み書きをしながら補正処理を実行するようにする。

【０３４５】

この実施形態では、画像メモリ６０は、水平処理用ＦＩＦＯラインメモリ６１と、垂直処理用小規模ラインメモリ６２とからなる。水平処理用ＦＩＦＯラインメモリ６１は、この例では、１ライン分の容量を備える。また、垂直処理用小規模ラインメモリ６２は、後述する垂直方向の補間処理用のＦＩＲ（Ｆｉｎｉｔｅ　Ｉｍｐｕｌｓｅ　Ｒｅｓｐｏｎｓｅ）フィルタのタップ数以上のライン数分の容量を有する。

【０３４６】

水平処理ブロック５１の水平画像処理部５１２には、水平処理用ＦＩＦＯラインメモリ６１への画像データの書き込み、読み出しを制御するＦＩＦＯメモリコントローラ（図３９では図示を省略）が設けられていると共に、算出した水平手ぶれ補正量ＳＸ＿ＡＤＤが小数点以下（補正後の画素位置が、水平方向の画素位置よりオフセットされた位置となる）の場合を想定し、補間処理を行なう水平方向補間処理部（図３９では図示を省略）が設けられている。この水平方向補間処理部は、後述するように、この実施形態では、水平方向のデジタルＦＩＲフィルタ（以下、水平ＦＩＲフィルタという）が用いられる。

【０３４７】

また、垂直処理ブロック５２の垂直画像処理部５２２には、垂直処理用小規模ラインメモリ６２への画像データの書き込み、読み出しを制御するメモリコントローラ（図３９では図示を省略）が設けられていると共に、算出した垂直手ぶれ補正量ＳＹ＿ＡＤＤが小数点以下（補正後の画素位置が、垂直方向の画素位置よりオフセットされた位置となる）の場合を想定し、補間処理を行なう垂直方向補間処理部（図３９では図示を省略）が設けられている。この垂直方向補間処理部は、後述するように、この実施形態では、垂直方向のデジタルＦＩＲフィルタ（以下、垂直ＦＩＲフィルタという）が用いられる。

【０３４８】

ここで、算出した水平手ぶれ補正量ＳＸ＿ＡＤＤおよび垂直手ぶれ補正量ＳＹ＿ＡＤＤが小数点以下の値を持つ場合の補間処理について説明する。

【０３４９】

例えば、手ぶれの水平方向の速度成分により、水平方向に図４２に示すような画像歪みが生じた場合を想定する。すなわち、図４２の例は、手ぶれが無ければ破線で示す直線上にあるべき画素Ｇ１１，Ｇ２１，Ｇ３１・・・、Ｇ１２，Ｇ２２，Ｇ３２・・・、Ｇ１３，Ｇ２３，Ｇ３３・・・、が、手ぶれの水平方向の速度成分のために実線で示すような斜め位置となっている場合である。

【０３５０】

この手ぶれを補正するためには、ずれている画素位置を本来の位置にあるようにずらせばよく、そのずれ量が、図４２に示すように、前述した水平手ぶれ補正量ＳＸ＿ＡＤＤに相当する。この水平手ぶれ補正量ＳＸ＿ＡＤＤが、画素ピッチｄｘの整数倍であれば、画素データを当該水平手ぶれ補正量ＳＸ＿ＡＤＤに応じた画素ピッチ分だけずらして読み出すようにすることで補正することができる。

【０３５１】

しかしながら、水平手ぶれ補正量ＳＸ＿ＡＤＤが、画素ピッチｄｘの整数倍ではなく、小数点以下のずれ量を含む場合には、そのずれ量に対応する位置は、画素データが存在する位置ではないので、その位置の近傍の複数個の画素データを用いた補間処理により、対応する画素位置の画素データを生成する必要がある。

【０３５２】

図４３は、その補間方法の一例を示すもので、求める画素データが、画素Ｇ１と画素Ｇ２との間の補間画素Ｇｓである場合に、この補間画素Ｇｓの位置と画素Ｇ１，Ｇ２の位置

との距離ｋ１，ｋ２に応じた割合で、画素Ｇ１のデータと画素Ｇ２のデータとを加算することにより、補間画素Ｇｓの画素データを生成するものである。この場合、ｋ１：ｋ２＝Ｗ：（１－Ｗ）としたとき、

補間画素Ｇｓ＝Ｇ１×Ｗ＋Ｇ２×（Ｗ－１）

なる補間演算により、補間画素Ｇｓの画素データを得る。

【０３５３】

また、図４３の例のような２個の画素データを用いるのではなく、図４４の例のように２個以上（図４４の例では４個）の近傍画素データを用いて、補間演算をして補間画素を求めることもできる。この場合、補間画素Ｇｓの位置に対する画素Ｇ０，Ｇ１，Ｇ２，Ｇ３の位置との距離ｋ０，ｋ１，ｋ２，ｋ３に応じた割合となる乗算係数Ｗ０，Ｗ１，Ｗ２，Ｗ３を定めて、画素Ｇ０，Ｇ１，Ｇ２，Ｇ３のデータを加算することにより、補間画素Ｇｓの画素データを得る。

【０３５４】

すなわち、

補間画素Ｇｓ＝Ｇ０×Ｗ０＋Ｇ１×Ｗ１＋Ｇ２×Ｗ２＋Ｇ３×Ｗ３

なる補間演算により、補間画素Ｇｓの画素データを得るようにする。

【０３５５】

この図４４の例の場合、乗算係数Ｗ０，Ｗ１，Ｗ２，Ｗ３の値の組は、画素Ｇ１または画素Ｇ２と補間画素Ｇｓとの距離ｄｓに対応するものとして関連付けられたテーブルを用意しておき、距離ｄｓを検索子として、上記乗算係数Ｗ０，Ｗ１，Ｗ２，Ｗ３の組をテーブルから読み出すようにすることで得ることができる。なお、距離ｄｓは、水平手ぶれ補正量ＳＸ＿ＡＤＤの小数部に相当するものである。

【０３５６】

この補間処理は、この実施形態では、水平ＦＩＲフィルタを用いて行なうようにする。すなわち、水平手ぶれ補正量ＳＸ＿ＡＤＤの小数部に対応する前記乗算係数の組をテーブルから読み出し、その乗算係数を水平ＦＩＲフィルタに対して供給することで上記補間処理を行なうことができる。

【０３５７】

以上は、水平方向の補間処理であるが、垂直方向についても、補間方向が垂直方向になるだけで、全く同様である。

【０３５８】

［水平処理ブロック５１における処理動作］

［水平手ぶれ補正量ＳＸ＿ＡＤＤの積分処理］

図４５に、水平手ぶれ補正量積分部５１１における積分値としての水平手ぶれ補正量ＳＸ＿ＡＤＤを得るための積分処理動作のフローチャートを示す。

【０３５９】

まず、初期ｙ座標（水平ライン位置に対応）ＳＹに、水平同期信号Ｈ－ＳＹＮＣ（ここでは、水平同期信号Ｈ－ＳＹＮＣは、一般的な水平走査信号のうち、有効画像領域以外をマスクしたタイミング信号を意味するものとする）毎に「ｓｔｅｐ」という値を加算して行く（ステップＳ２０１）。

【０３６０】

初期ｙ座標ＳＹは、出力画像の最初の1水平ライン目が、ＣＭＯＳイメージャ上の総画素の座標系（以後絶対座標系と呼ぶ）のどこに位置するかを表しており、一般的には「０」ではない。

【０３６１】

この初期座標（ＳＸ，ＳＹ）は、上述したような当該フレームの手ぶれに応じた初期的変位量を含むのみならず、次のような要因に基づいて定められるものである。すなわち、動画時には、通常、ＣＭＯＳイメージャ上の全画素が、信号処理部５０に入力されるのではなく、ＣＭＯＳイメージャ内で垂直方向に平均加算されてから、後段の信号処理部５０に入力されるからである。また、信号処理部５０に入力されてからでも、幾つかの処理を

経るため、1水平ライン目の絶対座標は、「０」でないことの方が多いからである。更に、入力画像データに対して、垂直方向の部分拡大処理を行なう場合には、初期ｙ座標ＳＹは、画面の中ほどの水平ライン位置となるので、非常に大きい値をとることもあり得るからである。

【０３６２】

ステップＳ２０１で、１水平同期信号Ｈ－ＳＹＮＣ毎に加算する「ｓｔｅｐ」という値は、ライン間隔の絶対座標系における垂直座標の増量を表わすが、上述したＣＭＯＳイメージャ内もしくは水平処理ブロックの前段における、垂直方向の複数ラインの画像データの平均加算により、１以外に２や４といった値をとる可能性がある。

【０３６３】

さて、ステップＳ２０１で、初期ｙ座標値をＳＹとして、１水平同期信号Ｈ－ＳＹＮＣ毎に、値ｓｔｅｐを加算することにより、現在処理している水平ラインの絶対座標が求まり、当該絶対座標の整数成分ｖｐ＿ｉ（ステップＳ２０２）が、水平補正速度成分Ｘ＿ＳＴＢ＿＊（＊は０～７のいずれかである。すなわち、水平補正速度成分Ｘ＿ＳＴＢ＿＊は、Ｘ＿ＳＴＢ＿０～Ｘ＿ＳＴＢ＿７のいずれかを意味する。以下、同じ。）を、現時点までに積分すべき積分回数に対応する。ここで、この積分回数は、１画面内における積算値である。

【０３６４】

次に、現在処理している水平ライン（以下、現処理ラインという）の絶対座標の整数成分ｖｐ＿ｉと、現処理ラインの1水平ライン前までに積分した回数（積分回数ｈｓｔｂ＿ｃｎｔ）とを比較する。これにより、現処理ラインにおいて、あと何回積分すれば良いかが分かり、その回数分だけ、処理単位サイクル（１水平周期よりも十分に短い）毎にトリガＴＧ１が発生する（ステップＳ２０３）。

【０３６５】

このトリガＴＧ１が発生する毎に、積分回数ｈｓｔｂ＿ｃｎｔは１づつインクリメントする（ステップＳ２０４、ステップＳ２０５）。

【０３６６】

そして、このトリガＴＧ１ごとに、水平補正速度成分Ｘ＿ＳＴＢ＿０～Ｘ＿ＳＴＢ＿７のうちの、現処理ラインが所在する分割画像区間についての水平補正速度成分を加算することにより、積分処理を行ない、水平手ぶれ補正量ＳＸ＿ＡＤＤを得る（ステップＳ２０６）。ここで、図４５のステップＳ２０６における「ｓｔｂ＿ｘ」は、水平補正速度成分Ｘ＿ＳＴＢ＿０～Ｘ＿ＳＴＢ＿７のいずれかの値である。すなわち、「ｓｔｂ＿ｘ」は、現処理ラインが所在する分割画像区間についての水平補正速度成分の値である。

【０３６７】

ステップＳ２０１で、１水平同期信号Ｈ－ＳＹＮＣ毎に加算される値「ｓｔｅｐ」＝１であって、画像の拡大などではない通常の画像出力の場合には、トリガＴＧ１は、１水平同期信号Ｈ－ＳＹＮＣ毎に発生するため、ステップＳ２０６では、１水平ライン毎に「ｓｔｂ＿ｘ」を加算（積分）するものである。

【０３６８】

こうして得られる水平手ぶれ補正量ＳＸ＿ＡＤＤは、前述もしたように、ＣＭＯＳイメージャ上での各水平ラインの水平方向の読み出し開始位置の補正量、すなわち、水平処理ブロック５１における水平方向画像歪み補正処理における各ラインの水平初期位置（水平オフセット）そのものとなる。

【０３６９】

上述したように、ステップＳ２０６では、現処理ラインが、１画面分（１フレーム＝１垂直同期周期）を垂直方向に8分割した分割画像区間のどこに属するかによって、水平補正速度成分Ｘ＿ＳＴＢ＿０～Ｘ＿ＳＴＢ＿７のうちの、いずれか１つを選択しなければならない。そのための仕組みが、図４５の残りの部分の処理である。

【０３７０】

この例では、分割画像区間分のライン数の加算値ｈｓｔｂ＿ｇｒｉｄが定義される（ス

テップＳ２０７）。この加算値ｈｓｔｂ＿ｇｒｉｄの初期値は、１分割画像区間のライン数ＤＩＶである。

【０３７１】

そして、この加算値ｈｓｔｂ＿ｇｒｉｄと、積分回数ｈｓｔｂ＿ｃｎｔとを比較し、積分回数ｈｓｔｂ＿ｃｎｔが加算値ｈｓｔｂ＿ｇｒｉｄを超える毎に、トリガＴＧ２を発生する（ステップＳ２０８）。そして、トリガＴＧ２が発生するごとに、積分回数ｈｓｔｂ＿ｃｎｔに、１分割画像区間のライン数ＤＩＶを加算する（ステップＳ２０９）。

【０３７２】

以上の処理により、現処理ラインが次の分割画像区間に移る毎に、トリガＴＧ２が発生する。したがって、現処理ラインが各分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７の何れに存在するかの分割区間値ＨＤＩＶ＿ＣＮＴを定義し（ステップＳ２１０）、この分割区間値ＨＤＩＶ＿ＣＮＴを、トリガＴＧ２が発生する毎に、１づつインクリメントする（ステップＳ２１１）。すると、分割区間値ＨＤＩＶ＿ＣＮＴは、現処理ラインが所在する各分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７のそれぞれに対応して「０」～「７」の値を取る。すなわち、分割区間値ＨＤＩＶ＿ＣＮＴは、水平処理の進捗度合いを示すものとなる。ステップＳ２０６では、この分割区間値ＨＤＩＶ＿ＣＮＴを参照して、「ｓｔｂ＿ｘ」を決定する。

【０３７３】

なお、この例では、分割区間値ＨＤＩＶ＿ＣＮＴ＝８のときには、１フレーム分の画像データについての処理が終了したことを意味する。そして、分割区間値ＨＤＩＶ＿ＣＮＴ＝８に、「１」が加わると、分割区間値ＨＤＩＶ＿ＣＮＴ＝０となるように計算される。

【０３７４】

この分割区間値ＨＤＩＶ＿ＣＮＴは、垂直処理ブロック５２およびレジスタブロック５３に、水平処理ブロック５１での現処理ラインが何れの分割画像区間に所在しているかを示すために、すなわち、水平処理の進捗状況を報知するために供給される。

【０３７５】

［水平画像処理部５１２の処理動作］

図４６は、水平画像処理部５１２の構成例を示すものである。この図４６の例では、前述の図４５のフローチャートにおける水平同期信号Ｈ－ＳＹＮＣ毎に値「ｓｔｅｐ」を初期ｙ座標ＳＹに加算する加算部５１０１が、この水平画像処理部５１２に含まれるものとしており、このため、この水平画像処理部５１２に初期ｙ座標ＳＹが入力される。さらに、この水平画像処理部５１２には初期ｘ座標ＳＸが入力される。これら初期ｙ座標ＳＹおよび初期ｘ座標ＳＸは、１フレームの先頭で、例えば制御部を構成するＣＰＵ１で設定されて入力される。

【０３７６】

そして、加算部５１０１からの、水平同期信号Ｈ－ＳＹＮＣ毎に値「ｓｔｅｐ」が加算されたｙ座標値ＳＹ′は、水平手ぶれ補正量積分部５１１に供給される。すると、この水平手ぶれ補正量積分部５１１からは、現処理ラインの先頭のｘ座標のオフセット値として、水平手ぶれ補正量ＳＸ＿ＡＤＤが返ってくる。

【０３７７】

水平画像処理部５１２では、加算部５１０２において、各水平ラインの先頭で一回だけ、水平手ぶれ補正量積分部５１１からの水平手ぶれ補正量ＳＸ＿ＡＤＤを、初期ｘ座標ＳＸに加算する。

【０３７８】

そして、この加算部５１０２の出力値に対して、画素処理クロックＣＬＫごとに、水平拡大・縮小のパラメータである値「ｈｍａｇ」を、加算部５１０３により加算する。この加算部５１０３の出力値Ｘ＿ＡＤＤは、比較部５１０４および５１０５、また、小数部抽出部５１０６および整数部抽出部５１０７に供給される。

【０３７９】

比較部５１０４は、加算部５１０３からの出力値Ｘ＿ＡＤＤを、１画素処理クロックＣ

ＬＫだけ前の出力値Ｘ＿ＡＤＤの整数成分と比較し、差が１以上であるとき、すなわち、画素位置が１画素分以上ジャンプしたときに、そのジャンプを示す信号ＳＫＩＰを出力する。この例では、このジャンプを示す信号ＳＫＩＰを参照することで、次に画素処理クロックＣＬＫのタイミングで処理すべきジャンプした画素位置を知ることができる。この信号ＳＫＩＰは、ＦＩＦＯメモリコントローラ５１０８に供給される。

【０３８０】

また、比較部５１０５は、加算部５１０３からの出力値Ｘ＿ＡＤＤを、１画素処理クロックＣＬＫだけ前の出力値Ｘ＿ＡＤＤの整数成分と比較し、差が１より小さいときに、その旨を示す信号ＨＯＬＤを出力する。したがって、この信号ＨＯＬＤを参照することで、読み出し画素を、１画素処理クロックＣＬＫだけ前と同じであることを知ることができる。この信号ＨＯＬＤは、ＦＩＦＯメモリコントローラ５１０８に供給される。

【０３８１】

小数部抽出部５１０６は、加算部５１０３からの出力値Ｘ＿ＡＤＤの小数部を、画素処理クロックＣＬＫ毎に抽出し、抽出した小数部の値Ｘ＿ＰＨＡＳＥを出力する。この少数部の値Ｘ＿ＰＨＡＳＥは、水平ＦＩＲフィルタ５１１０を備える水平補間処理部５１０９に供給される。水平補間処理部５１０９では、この小数部の値Ｘ＿ＰＨＡＳＥに基づいて乗算係数テーブルを検索し、水平ＦＩＲフィルタ５１１０に供給する乗算係数を得るようにする。

【０３８２】

整数部抽出部５１０７は、加算部５１０３からの出力値Ｘ＿ＡＤＤの整数部を、水平同期信号Ｈ－ＳＹＮＣにより各水平ラインの先頭で抽出し、抽出した整数部の値ＳＴ＿ＰＯＳを出力する。この整数部の値ＳＴ＿ＰＯＳは、現処理ラインの初期ｘ座標としてＦＩＦＯメモリコントローラ５１０８に供給される。

【０３８３】

ＦＩＦＯメモリコントローラ５１０８は、入力画像データＤｉｎについて、水平処理用ＦＩＦＯラインメモリ６１を用いて、水平手ぶれ補正量ＳＸ＿ＡＤＤに基づいた水平手ぶれ補正処理を行なうと共に、水平拡大・縮小処理を行なう。ここで、水平処理用ＦＩＦＯラインメモリ６１が、実際に１水平ライン分相当のサイズを必要とするのは、後述するように、画像の部分拡大処理をサポートすることを考慮する場合のみである。

【０３８４】

すなわち、ＦＩＦＯメモリコントローラ５１０８は、入力画像データＤｉｎについて、前記整数部の値ＳＴ＿ＰＯＳを用いて、出力画像データとして読み出す各水平ラインの画素データの初期ｘ座標を決める。また、ＦＩＦＯメモリコントローラ５１０８は、信号ＳＫＩＰおよび信号ＨＯＬＤを画素処理クロックＣＬＫごとに参照し、水平処理用ＦＩＦＯラインメモリ６１から読み出す画素データを決定するようにする。

【０３８５】

そして、ＦＩＦＯメモリコントローラ５１０８は、以上のようにして読み出した水平ラインの画素データを出力画像データとして、データイネーブル信号ＥＮと共に、水平補間処理部５１０９に供給する。

【０３８６】

水平補間処理部５１０９は、データイネーブル信号ＥＮがイネーブル状態となっているときに入力される画像データを有効データとして、水平ＦＩＲフィルタ５１１０を用いて水平方向の補間処理を実行する。前述したように、このときに水平補間処理部５１０９に入力される小数部の値Ｘ＿ＰＨＡＳＥを用いて、水平ＦＩＲフィルタ５１１０に供給する乗算係数が決定されている。

【０３８７】

こうして、水平補間処理部５１０９からは、水平手ぶれ補正量ＳＸ＿ＡＤＤに応じて水平手ぶれ補正処理がなされると共に、水平方向の拡大・縮小処理がなされた画像データＤＨｏｕｔが得られる。このＤＨｏｕｔは、垂直処理ブロック５２に供給される。

【０３８８】

ところで、上述の説明では、水平手ぶれ補正量積分部５１１の水平手ぶれ補正量ＳＸ＿ＡＤＤの積分処理と、水平画像処理部５１１の水平拡大・縮小画像処理の、互いの処理タイミングについて特に触れなかったが、水平手ぶれ補正量ＳＸ＿ＡＤＤの積分処理を、遅延無しで、水平拡大・縮小画像処理との単純なシーケンス処理で行なうことが好都合なのは、図４７の上側の図４７（Ａ）に示すように、垂直方向の部分切り出し及び部分拡大をサポートしない場合にほぼ限られる。図４７で、Ｖ－ＳＹＮＣは垂直同期信号を示し、また、ＳＴ＿ＴＧは、有効走査ラインの開始時点を示している。また、図４７で、網掛けを付して示した四角で囲まれた区間は、それぞれの処理区間を示している。

【０３８９】

一方、垂直方向の部分切り出し、もしくは部分拡大をサポートする場合、初期ｙ座標ＳＹが非常に大きな値を取り得るため、フレーム先頭の積分処理に、相当長いサイクル数（画素処理サイクル数）を要してしまう可能性がある。図４７下側の（Ｂ）のシーケンスにおける期間ｔｍがそのための積分期間である。

【０３９０】

このようになると、図４７上側の（Ａ）のシーケンスでは、最初の拡大・縮小処理が水平同期信号Ｈ－ＳＹＮＣの１周期期間をオーバーしてしまうため、次の水平同期信号Ｈ－ＳＹＮＣで入力されて来る２ライン目の画像データを保持しておくためのラインメモリが必要となる。

【０３９１】

そこで、この実施形態では、図４７下側の（Ｂ）のシーケンスに示すように、水平拡大・縮小画像処理が開始される１水平周期前から積分処理の起動を行ない、常に画像処理に対して、１水平周期だけ先行して積分処理を行なうタイミングおよびシーケンスを採用する。このため、水平処理用ＦＩＦＯラインメモリ６１は、１水平周期分のサイズのものがこの実施形態では用いられている。

【０３９２】

上述した問題は、画像の部分拡大等に関わらず、水平処理は常に全入力ラインに対して処理を施す、というアーキテクチャにすれば避けられる。しかし、いずれにしても、図４７上側のシーケンスを実現しようとすると、積分処理の終了信号を新たに設け、それを受けて画像処理の起動をかけなければならず、やや煩雑である。つまり、換言すれば、上述したこの実施形態で採用した手法は、既存の画像処理の回路に新たな手を加えないで済む、というメリットがあるのである。

【０３９３】

なお、上述した実施形態で採用した手法は、水平同期信号Ｈ－ＳＹＮＣの１周期期間内の画素処理サイクル数よりも、絶対座標系の垂直サイズが大きい場合には採用できない。その場合には、水平処理は常に全入力ラインに対して処理を施すという、上述したアーキテクチャにすれば良い。

【０３９４】

［垂直処理ブロック５２における処理動作］

［垂直手ぶれ補正量ＳＹ＿ＡＤＤの積分処理］

次に、垂直手ぶれ補正量積分部５２１における積分値としての垂直手ぶれ補正量ＳＹ＿ＡＤＤを得るための積分処理動作について説明する。図４８は、この垂直手ぶれ補正量ＳＹ＿ＡＤＤを得るための積分処理動作のフローチャートを示すものである。

【０３９５】

この垂直手ぶれ補正量ＳＹ＿ＡＤＤを得るための積分処理動作は、図４５を用いて上述した水平手ぶれ補正量ＳＸ＿ＡＤＤを得るための積分処理動作と類似しているが、次の点が異なる。

【０３９６】

すなわち、水平同期信号Ｈ－ＳＹＮＣ毎に初期ｙ座標ＳＹに加算する値が「ｖｍａｇ」であることと、水平処理の進捗度合いを表すＨＤＩＶ＿ＣＮＴを見て、ウエイト（ＷＡＩＴ）をかけることと、各ラインの積分処理が終了したことを表わす積分完了信号ＳＴＢ＿

ＲＤＹを生成して、垂直信号処理モジュールへ伝達すること、の以上３点である。

【０３９７】

この実施形態では、垂直画像処理において垂直方向の拡大・縮小を想定しており、値「ｖｍａｇ」は垂直拡大・縮小の倍率パラメータを表わしている。したがって、垂直処理ブロック５２では、水平処理ブロック５１から出力される画像データを全て処理するのではなく、拡大・縮小処理後に出力するラインの垂直座標を基準として処理を行なう。

【０３９８】

そのため、この垂直方向の積分処理においても、実際に、この垂直方向の積分処理後の垂直画像処理の対象となる座標における積分値を出力することが求められることから、垂直方向の座標増分パラメータを水平同期信号Ｈ－ＳＹＮＣ毎に加算するのである。なお、画像の拡大・縮小を伴わずに、ＣＭＯＳ手ぶれの補正のみを行なうのであれば、ｖｍａｇ＝1として良い。

【０３９９】

水平方向の積分処理とは異なる二つ目の、分割区間値ＨＤＩＶ＿ＣＮＴを見て、水平処理の進捗度合いを観測する必要性は、水平処理を垂直処理が追い越すことがあってはならないからである。一般的な画像の垂直拡大・縮小処理においては、垂直処理が水平処理を追い越す、というのは、まだ生成されていない画像に対して処理を施すことであるため、自ずとウエイトをかける仕組みが必要となる。同様に、この垂直方向の積分処理においても、画像を直接扱う訳ではないが、水平処理を追い越さない仕掛けが必要なのである。

【０４００】

もう少し詳細に説明する。水平および垂直方向の積分パラメータである水平補正速度成分Ｘ＿ＳＴＢ＿＊（＊は０～７のいずれかであり、Ｘ＿ＳＴＢ＿＊は、Ｘ＿ＳＴＢ＿０～Ｘ＿ＳＴＢ＿７のいずれかを意味する。以下同じ）と、垂直補正速度成分Ｙ＿ＳＴＢ＿＊（＊は０～７のいずれかであり、Ｙ＿ＳＴＢ＿０～Ｙ＿ＳＴＢ＿７のいずれかを意味する。以下、同じ）は、処理を行っている水平ラインのサンプリング時刻（厳密に言うと露光期間の中心時点であることは既に説明）付近の手ぶれ速度ベクトル（手ぶれの速度と方向）の、それぞれ水平成分、垂直成分の逆符号である。したがって、露光時間が極めて短い時間の場合まで考慮すると、水平補正速度成分Ｘ＿ＳＴＢ＿＊と垂直補正速度成分Ｙ＿ＳＴＢ＿＊は、対応する水平ラインの画像データが入力される直前まで確定できない場合がある。

【０４０１】

後で詳述するレジスタブロック５３では、この事情を考慮し、水平方向の積分処理において現在処理しているラインが属する分割画像区間を意味する分割区間値ＨＤＩＶ＿ＣＮＴの値を見て、水平補正速度成分Ｘ＿ＳＴＢ＿＊と垂直補正速度成分Ｙ＿ＳＴＢ＿＊を、水平処理ブロック用レジスタと、垂直処理ブロック用レジスタにラッチする。したがって、水平処理ブロック５１では、特別な仕組みを設けなくとも、入力された画像を順次処理して行けば良いのである。

【０４０２】

一方、垂直処理ブロック５２での積分処理では、当該積分処理を行なった結果を、後段の垂直方向画像処理の初期オフセット座標とするため、水平処理からの画像が未だ入って来ていないことに後から気付いても、その時点では既にそのラインの積分処理は終わってしまっている。つまり、そのときには、未確定の垂直補正速度成分が用いられて積分がなされてしまったことになる。

【０４０３】

そのため、積分処理の段階で、処理しようとするラインに対応した積分パラメータである垂直補正速度成分Ｙ＿ＳＴＢ＿＊が、まだ確定前であることを、水平処理の進捗度合いから判断する必要があるのである。

【０４０４】

そのための仕組みとして、図４８のように、垂直処理の進捗度合いを表わす分割区間値ＶＤＩＶ＿ＣＮＴが、水平処理の進捗度合いを表わす分割区間値ＨＤＩＶ＿ＣＮＴを超え

てしまっていれば、ウエイトを発行し、図４８の左下で、垂直処理における現処理ラインが所在する分割画像区間についての垂直補正速度成分の値である「ｓｔｂ＿ｙ」が未確定のままの垂直補正速度成分Ｙ＿ＳＴＢ＿＊が選択された状態で、垂直手ぶれ補正量ＳＹ＿ＡＤＤに加算されるのを防ぐ。

【０４０５】

垂直方向の積分処理（図４８）が水平方向の積分（図４５）と異なる、３つ目の点は、各ラインでの垂直積分が完了したことを表す積分完了信号ＳＴＢ＿ＲＤＹの生成である。各ライン内で積分を繰り返して行なった結果、そのラインの絶対座標の整数成分ｖｐ＿ｉと、そのフレーム内での累積積分回数ｖｓｔｂ＿ｃｎｔとが一致したときに、後段の垂直画像処理モジュールに渡すべき垂直オフセットＳＹ＿ＡＤＤが求められ、それと同時に、後段の垂直画像処理の起動をかけるのが目的である。

【０４０６】

既に説明した水平方向の積分処理においては、有効画像が入力されるタイミングよりも１水平同期信号Ｈ－ＳＹＮＣ（１水平周期）前から積分処理を開始しておけば、有効画像に対する水平画像処理が始まるまでに積分処理を完了できることが、殆どの場合、システム的に保証可能であるが、垂直方向の積分処理では、前述のウエイトが発生する可能性があるために、直前１水平周期の時間で積分が完了することを、全ての場合において保証するのは不可能だからである。すなわち、ウエイトが発生したときには、垂直積分が完了したことを表す積分完了信号ＳＴＢ＿ＲＤＹが発生したときに、垂直手ぶれ補正および垂直拡大・縮小処理を含む垂直画像処理を実行するようにする。

【０４０７】

以上の機能を有する、この垂直方向の積分回路を適用した垂直方向の積分処理を図４８のフローチャートを参照しながら説明する。

【０４０８】

まず、初期ｙ座標（水平ライン位置に対応）ＳＹに、水平同期信号Ｈ－ＳＹＮＣ毎に前述した値「ｖｍａｇ」を加算して行く（ステップＳ３０１）。

【０４０９】

ステップＳ３０１で、初期ｙ座標値をＳＹとして、１水平同期信号Ｈ－ＳＹＮＣ毎に、値ｖｍａｇを加算することにより、現在処理している水平ラインの絶対座標が求まり、当該絶対座標の整数成分ｖｐ＿ｉ（ステップＳ３０２）が、垂直補正速度成分Ｙ＿ＳＴＢ＿＊（Ｙ＿ＳＴＢ＿０～Ｙ＿ＳＴＢ＿７のいずれか）を、現時点までに積分すべき積分回数に対応する。ここで、この積分回数は、１画面内における積算値である。

【０４１０】

次に、現在処理している水平ライン、すなわち、現処理ラインの絶対座標の整数成分ｖｐ＿ｉと、現処理ラインの1水平ライン前までに積分した回数（積分回数ｖｓｔｂ＿ｃｎｔ）とを比較する。これにより、現処理ラインにおいて、あと何回積分すれば良いかが分かり、その回数分だけ、処理単位サイクル（１水平周期よりも十分に短い）毎にトリガＴＧ３が発生する（ステップＳ３０３）。

【０４１１】

このトリガＴＧ３が発生する毎に、積分回数ｖｓｔｂ＿ｃｎｔは１づつインクリメントする（ステップＳ３０４、ステップＳ３０５）。

【０４１２】

そして、このトリガＴＧ３ごとに、垂直補正速度成分Ｙ＿ＳＴＢ＿０～Ｙ＿ＳＴＢ＿７のうちの、現処理ラインが所在する分割画像区間についての垂直補正速度成分を加算することにより、積分処理を行ない、垂直手ぶれ補正量ＳＹ＿ＡＤＤを得る（ステップＳ３０６）。ここで、図４８のステップＳ３０６における「ｓｔｂ＿ｙ」は、垂直補正速度成分Ｙ＿ＳＴＢ＿０～Ｙ＿ＳＴＢ＿７のいずれかの値である。すなわち、「ｓｔｂ＿ｙ」は、現処理ラインが所在する分割画像区間についての垂直補正速度成分の値である。

【０４１３】

ステップＳ３０１で、１水平同期信号Ｈ－ＳＹＮＣ毎に加算される値「ｖｍａｇ」＝１

であって、画像の拡大などではない通常の画像出力の場合には、トリガＴＧ３は、１水平同期信号Ｈ－ＳＹＮＣ毎に発生するため、ステップＳ３０６では、１水平ライン毎に「ｓｔｂ＿ｙ」を加算（積分）するものである。

【０４１４】

こうして得られる垂直手ぶれ補正量ＳＹ＿ＡＤＤは、前述もしたように、ＣＭＯＳイメージャ上での各水平ラインの垂直方向の読み出し開始位置の補正量、すなわち、垂直処理ブロック５２における垂直方向画像歪み補正処理における各ラインの垂直位置（垂直オフセット）そのものとなる。

【０４１５】

上述したように、ステップＳ３０６では、現処理ラインが、１画面分（１フレーム＝１垂直同期周期）を垂直方向に8分割した分割画像区間のどこに属するかによって、垂直補正速度成分Ｙ＿ＳＴＢ＿０～Ｙ＿ＳＴＢ＿７のうちの、いずれか１つを選択しなければならない。

【０４１６】

この例では、分割画像区間分のライン数の加算値ｖｓｔｂ＿ｇｒｉｄが定義される（ステップＳ３０７）。この加算値ｖｓｔｂ＿ｇｒｉｄの初期値は、１分割画像区間のライン数ＤＩＶである。

【０４１７】

そして、この加算値ｖｓｔｂ＿ｇｒｉｄと、積分回数ｖｓｔｂ＿ｃｎｔとを比較し、積分回数ｖｓｔｂ＿ｃｎｔが加算値ｖｓｔｂ＿ｇｒｉｄを超える毎に、トリガＴＧ４を発生する（ステップＳ３０８）。そして、トリガＴＧ４が発生するごとに、加算値ｖｓｔｂ＿ｃｎｔに、１分割画像区間のライン数ＤＩＶを加算する（ステップＳ３０９）。

【０４１８】

以上の処理により、現処理ラインが次の分割画像区間に移る毎に、トリガＴＧ４が発生する。したがって、現処理ラインが各分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７の何れに存在するかの分割区間値ＶＤＩＶ＿ＣＮＴを定義し（ステップＳ３１０）、この分割区間値ＶＤＩＶ＿ＣＮＴを、トリガＴＧ４が発生する毎に、１づつインクリメントする（ステップＳ３１１）。すると、分割区間値ＶＤＩＶ＿ＣＮＴは、現処理ラインが所在する各分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７のそれぞれに対応して「０」～「７」の値を取る。すなわち、分割区間値ＶＤＩＶ＿ＣＮＴは、垂直処理の進捗度合いを示すものとなる。

【０４１９】

なお、この例では、分割区間値ＶＤＩＶ＿ＣＮＴ＝８のときには、１フレーム分の画像データについての処理が終了したことを意味する。そして、分割区間値ＶＤＩＶ＿ＣＮＴ＝８に、「１」が加わると、分割区間値ＶＤＩＶ＿ＣＮＴ＝０となるように計算される。

【０４２０】

この分割区間値ＶＤＩＶ＿ＣＮＴは、水平処理ブロック５１からの、水平処理の進捗度合いを示す分割区間値ＨＤＩＶ＿ＣＮＴと比較され、分割区間値ＶＤＩＶ＿ＣＮＴが分割区間値ＨＤＩＶ＿ＣＮＴよりも大きくなったときには、水平処理を垂直処理が追い越してしまう状態を意味するので、ウエイト（ＷＡＩＴ）を発生して、トリガＴＧ３の発生を待機させる。すなわち、トリガＴＧ３は、ウエイトが解除された後に発生するように遅延させる（ステップＳ３１２）。

【０４２１】

そして、この実施形態では、ステップＳ３０２で得られるｙ座標の整数成分ｖｐ＿ｉと、積分回数ｖｓｔｂ＿ｃｎｔとが等しいかどうかをチェックし、等しければ現処理ラインについての垂直方向の積分が完了したことを示す積分完了信号ＳＴＢ＿ＲＤＹを生成して出力する（ステップＳ３１３）。この積分完了信号ＳＴＢ＿ＲＤＹは、垂直画像処理部５２２に供給される。垂直画像処理部５２２では、この積分完了信号ＳＴＢ＿ＲＤＹにより、現処理ラインについての垂直方向の手ぶれ補正処理を開始してよいことを認識し、次の水平同期信号Ｈ－ＳＹＮＣのタイミングで垂直画像処理を再開する。

【０４２２】

［垂直画像処理部５２２の処理動作］
図４９は、垂直画像処理部５２２の構成例を示すものである。この図４９の例では、前述の図４８のフローチャートにおける水平同期信号Ｈ－ＳＹＮＣ毎に値「ｖｍａｇ」を加算する加算部５２０１が、この垂直画像処理部５２２に含まれるものとしており、この垂直画像処理部５２２に初期ｙ座標ＳＹが入力される。
【０４２３】
そして、加算部５２０１からの、垂直同期信号Ｈ－ＳＹＮＣ毎に値「ｖｍａｇ」が加算されたｙ座標値ＳＹ´は、垂直手ぶれ補正量積分部５２１に供給される。すると、この垂直手ぶれ補正量積分部５２１からは、現処理ラインの先頭のｙ座標のオフセット値として、垂直手ぶれ補正量ＳＹ＿ＡＤＤが返ってくる。
【０４２４】
垂直画像処理部５２２では、加算部５２０２において、各水平ラインの先頭で一回だけ、垂直手ぶれ補正量積分部５２１からの垂直手ぶれ補正量ＳＹ＿ＡＤＤを、初期ｙ座標ＳＹに加算する。この加算部５２０２の出力値Ｙ＿ＡＤＤは、小数部抽出部５２０３および整数部抽出部５２０４に供給される。
【０４２５】
小数部抽出部５２０３は、加算部５２０２からの出力値Ｙ＿ＡＤＤの小数部を、水平同期信号Ｈ－ＳＹＮＣにより各水平ラインの先頭で抽出し、抽出した小数部の値Ｙ＿ＰＨＡＳＥを出力する。この少数部の値Ｙ＿ＰＨＡＳＥは、垂直ＦＩＲフィルタ５２０７を備える垂直補間処理部５２０６に供給される。
【０４２６】
垂直ＦＩＲフィルタ５２０７は、垂直方向に並ぶ複数ラインの画素データを用いて補間処理を行なうもので、垂直補間処理部５２０６では、この小数部の値Ｙ＿ＰＨＡＳＥに基づいて乗算係数テーブルを検索し、垂直ＦＩＲフィルタ５２０７に供給する乗算係数を得るようにする。これは、補間方向が垂直方向になるだけで、図４２～図４４を用いて説明した水平ＦＩＲフィルタを用いる水平方向の補間処理とほぼ同様である。
【０４２７】
整数部抽出部５２０４は、加算部５２０２からの出力値Ｙ＿ＡＤＤの整数部を、水平同期信号Ｈ－ＳＹＮＣにより各水平ラインの先頭で抽出し、抽出した整数部の値Ｙ＿ＰＯＳを出力する。この整数部の値Ｙ＿ＰＯＳは、現処理ラインのｙ座標としてメモリコントローラ５２０５に供給される。
【０４２８】
メモリコントローラ５２０５は、水平処理ブロック５１からの画像データＤＨｏｕｔについて、垂直処理用小規模ラインメモリ６２を用いて、垂直手ぶれ補正量ＳＹ＿ＡＤＤに基づいた垂直手ぶれ補正処理を行なうと共に、垂直拡大・縮小処理を行なう。ここで、垂直処理用小規模ラインメモリ６２は、少なくとも、垂直ＦＩＲフィルタ５２０７における補間処理に用いるライン数分、すなわち、垂直ＦＩＲフィルタのタップ数分の容量を有する。
【０４２９】
メモリコントローラ５２０５は、その入力画像データＤＨｏｕｔを垂直処理用小規模ラインメモリ６２に一時保存すると共に、前記整数部の値Ｙ＿ＰＯＳを用いて、出力画像データＤＹｏｕｔ＿０、ＤＹｏｕｔ＿１、・・・、ＤＹｏｕｔ＿ｎとして出力する複数の水平ラインのデータを決める。そして、メモリコントローラ５２０５は、以上のようにして決定した複数水平ラインの画素データＤＹｏｕｔ＿０、ＤＹｏｕｔ＿１、・・・、ＤＹｏｕｔ＿ｎを、小規模ラインメモリ６２から読み出して垂直補間処理部５２０６に供給する。
【０４３０】
垂直補間処理部５２０６は、入力される画像データＤＹｏｕｔ＿０、ＤＹｏｕｔ＿１、・・・、ＤＹ＿ｎを、垂直ＦＩＲフィルタ５２０７を用いて垂直方向の補間処理を実行する。前述したように、このときに垂直補間処理部５２０６に入力される小数部の値Ｙ＿Ｐ

ＨＡＳＥを用いて、垂直ＦＩＲフィルタ５２０７に供給する乗算係数が決定されている。

【０４３１】

こうして、垂直補間処理部５２０６からは、垂直手ぶれ補正量ＳＹ＿ＡＤＤに応じて垂直手ぶれ補正処理がなされると共に、垂直方向の拡大・縮小処理がなされた画像データＤｏｕｔが得られる。

【０４３２】

次に、図５０に、上説明した垂直方向の積分処理と、その後段の垂直画像処理のタイミング関係を示す。図５０の上の図５０（Ａ）のように、垂直方向の部分切り出し、垂直方向の部分拡大・縮小のいずれもサポートしない場合には、前述したウエイトは発生せず、現処理ラインについての垂直方向の積分回数の上限は高々数回である。このため、同一のラインに対する積分処理と画像処理の両方を、同じ１水平同期信号Ｈ－ＳＹＮＣ期間内でシーケンス化可能である。したがって、この場合には、わざわざ積分完了信号ＳＴＢ＿ＲＤＹを導入する必要はない。

【０４３３】

一方、図５０の下の図５０（Ｂ）のように、垂直方向の部分切り出し、もしくは垂直方向の部分拡大をサポートする場合には、1ライン目の積分処理の途中でウエイトが発生する可能性があり、縮小をサポートする場合には、どこか途中の水平ラインでウエイトが発生する可能性がある。

【０４３４】

いずれの場合でも、同一ラインの積分処理が、ウエイトによって数水平同期信号Ｈ－ＳＹＮＣ区間もしくはそれ以上に渡って中断されるため、積分処理が終了したことを後段に知らせるための積分完了信号ＳＴＢ＿ＲＤＹが必要となる。そして、図５０（Ｂ）の下２段に示すように、垂直拡大・縮小処理は、ウエイトが終了し、積分完了信号ＳＴＢ＿ＲＤＹが発生した後の水平同期信号Ｈ－ＳＹＮＣのタイミングから開始される。

【０４３５】

また、同時に、あるラインの垂直画像処理が完了していない場合には、次の水平同期信号Ｈ－ＳＹＮＣを、図５０（Ｂ）において点線で示すように、マスクするようにする。すなわち、ウエイトが発生している間の水平同期信号Ｈ－ＳＹＮＣは、マスクするようにする。

【０４３６】

以上説明した水平処理ブロック５１と垂直処理ブロック５２を含む信号処理部５における処理の流れは、図５１に示すように表わすことができる。すなわち、データ変換部３からの画像データＤｉｎは、水平処理用ＦＩＦＯラインメモリ６１に一旦書き込んで読み出し、水平画像処理部５１２および水平手ぶれ補正量積分部５１１を用いて水平手ぶれ補正および水平拡大・縮小処理を行なう。そして、水平画像処理部５１２の処理結果は、垂直処理用小規模ラインメモリ６２に書き込む。

【０４３７】

垂直処理用小規模ラインメモリ６２に書き込まれた画像データは、垂直画像処理部５２２および垂直手ぶれ補正量積分部５２１を用いて垂直手ぶれ補正および垂直拡大・縮小処理を行なう。そして、垂直画像処理部５２２の処理結果は、垂直処理用小規模ラインメモリ６２に書き戻す。そして、さらに、この垂直処理用小規模ラインメモリ６２から、出力画像データＤｏｕｔを読み出して出力する。

【０４３８】

以上の処理の結果、図５２の右下図のように、ＣＭＯＳ手ぶれに起因する歪み補正がなされた出力画像が得られる。

【０４３９】

ここでの垂直処理用小規模ラインメモリ６２としては、最低限、垂直ＦＩＲフィルタ５２０７のタップ数分のラインメモリが必要であることを述べたが、現実的には、垂直方向の部分拡大処理を行なう場合には、垂直処理のスピードよりも水平処理のスピードの方が早くなるため、拡大対象となる水平処理結果を溜めておくだけのラインメモリが必要とな

る。
【０４４０】
また、部分拡大をしない場合でも、垂直補正速度成分Ｙ＿ＳＴＢ＿＊が鉛直上向きである場合には、部分拡大の場合と動作上同じになるため、小規模ラインメモリ６２としては、手ぶれが上下方向に変動する分を考慮したラインメモリが必要である。
【０４４１】
なお、上述の実施形態の場合、１垂直周期が１／６０秒であるとすると、分割画像区間Ｐｄｉｖの時間長ＤＩＶは、１／６０÷８＝１／４８０秒であるが、実験によれば、この程度の時間長ＤＩＶで、手ぶれ速度ベクトルを求めて手ぶれ補正をすることにより、必要十分な手ぶれ補正効果が得られ、補正後の出力画像としては、良好なものとなることが確認されている。
【０４４２】
次に、以上の水平処理および垂直処理におけるレジスタブロック５３におけるレジスタへのラッチタイミングを、図５３に示す。
【０４４３】
すなわち、この実施形態では、垂直同期信号Ｖ－ＳＹＮＣの１周期について、各分割画像区間Ｐｄｉｖ＿０～Ｐｄｉｖ＿７の先頭ラインの時点で割り込み（マイコン割り込み）が発生する。
【０４４４】
この割り込みにより起動されて、速度ベクトル処理部８０で、前述したようにして、水平補正速度成分Ｘ＿ＳＴＢ＿＊および垂直補正速度成分Ｙ＿ＳＴＢ＿＊が生成され、その生成結果である水平補正速度成分Ｘ＿ＳＴＢ＿＊および垂直補正速度成分Ｙ＿ＳＴＢ＿＊がレジスタブロック５３のＩＦレジスタに取り込まれる。
【０４４５】
水平補正速度成分Ｘ＿ＳＴＢ＿＊および垂直補正速度成分Ｙ＿ＳＴＢ＿＊の値を、ＣＰＵ１のソフトウエアで計算する場合、図５２に示すような定期的な割り込みを発生させる方法が有効である。
【０４４６】
また、レジスタブロック５３内では、分割区間値ＨＤＩＶ＿ＣＮＴの値の変化を見てラッチパルスを生成し、前述のようにＣＰＵ１が設定した、対応する水平補正速度成分Ｘ＿ＳＴＢ＿＊と垂直補正速度成分Ｙ＿ＳＴＢ＿＊とを、ＩＦレジスタから、水平処理用レジスタおよび垂直処理用レジスタに、この生成したラッチパルスのタイミングでラッチする。つまり、水平処理ブロック５１で用いる水平補正速度成分Ｘ＿ＳＴＢ＿＊を、その使用直前でラッチするのである。このとき、垂直処理ブロック５２で用いる垂直補正速度成分Ｙ＿ＳＴＢ＿＊も一緒にラッチしておく。
【０４４７】
垂直処理ブロック５２で使用する、垂直補正速度成分Ｙ＿ＳＴＢ＿＊は、基本的には、上記の水平処理ブロック５１用の水平補正速度成分Ｘ＿ＳＴＢ＿＊と一緒にラッチした、Ｙ＿ＳＴＢ＿＊の値を常にラッチし続ける。
【０４４８】
ただし、それは、垂直処理ブロック５２のフレーム処理起動パルスＶＬ１がアクティブ（ハイ状態がアクティブ）になってから、水平処理ブロック５１のフレーム処理起動パルスＶＬ０がアクティブ（ハイ状態がアクティブ）になるまでの期間（図５２のＲＶラッチイネーブルのアクティブ（ハイ状態がアクティブ）期間）に限られる。
【０４４９】
これは、垂直処理の時間は長引くことがあり、入力画像の垂直同期信号Ｖ－ＳＹＮＣだけでなく、次のフレームの水平処理ブロックのフレーム起動パルスＶＬ０さえも跨ぐことがあるからである。
【０４５０】
［この発明による画像信号の歪み補正装置の第２の実施形態］

上述した画像信号の歪み補正装置の第１の実施形態としての撮像装置における手ぶれ動きベクトル検出部１５においては、図３０に示したように、画像メモリ部４は、２枚の画像、つまり元フレームの画像と、参照フレームの画像とが、両方共、フレームメモリ４２，４２に格納されていることを前提にしていた。このため、動きベクトルの検出タイミングは、１フレーム分遅延されることとなる。

【０４５１】

これに対して、この第２の実施形態では、撮像素子１１からの垂れ流し画像データを参照フレームとする構成として、ラスタスキャンのストリームデータに対して、リアルタイムでＳＡＤ値を演算することができるようにしている。

【０４５２】

図５３に、この第２の実施形態の場合における撮像装置の構成例のブロック図を示す。この図５３から分かるように、撮像信号処理系１０の構成ブロックおよびその他の構成ブロックは、図３０に示した第１の実施形態と全く同様であるが、この第２の実施形態においては、図５３に示すように、画像メモリ部４は１個のフレームメモリ４３からなる。

【０４５３】

この第２の実施形態では、元フレームがフレームメモリ４３に格納されており、参照フレームは、データ変換部１４からストリームで入力されて来るものとされる。手ぶれ動きベクトル検出部１５は、第１の実施形態では、２個のフレームメモリ４１，４２に格納された２枚の画像データを用いて、参照ブロックについてのＳＡＤ値を求める処理をするようにした。これに対して、この第２の実施形態では、図５３に示すように、データ変換部１４からのストリーム画像データを参照フレームの画像データとすると共に、フレームメモリ４３に格納されている画像データを元フレームの画像データとして、参照ブロックについてのＳＡＤ値を求めるようにする。

【０４５４】

そして、歪み補正解像度変換部１６は、フレームメモリ４３からの画像データの切り出しを、手ぶれ動きベクトル検出部１５で検出された動きベクトルに基づいて行なうことで、手ぶれの無い画像データを出力するようにしている。その他の構成および動作は、第１の実施形態と同様である。

【０４５５】

上述したように、この第２の実施形態では、データ変換部１４からのストリーム画像データを参照フレームの画像データとする。このため、ある入力画素に対して、この画素を要素とする参照ブロックが、参照フレーム上に同時に複数存在することになる。図５４は、そのことを説明するための図である。

【０４５６】

すなわち、参照フレーム１０２上のサーチ範囲１０５における入力画素Ｄｉｎは、例えば、参照ベクトル１０７１が対応する参照ブロック１０６１の左側に位置する画素であると共に、参照ベクトル１０７２が対応する参照ブロック１０６２の右上に位置する画素となっていることが、この図５４から分かる。

【０４５７】

したがって、入力画素Ｄｉｎが参照ブロック１０６１に属するとした場合には、ターゲットブロック１０３の画素Ｄ１を読み出して、その差分を算出する必要がある。また、入力画素Ｄｉｎが参照ブロック１０６２に属するとした場合には、ターゲットブロック１０３の画素Ｄ２を読み出して、その差分を算出する必要がある。

【０４５８】

図５４および後述の図５５では簡単のため、２つの参照ブロックのみを図示しているが、実際上は、入力画素Ｄｉｎを、その参照ブロック内の画素とする参照ブロックは多数となる。

【０４５９】

この第２の実施形態の場合のＳＡＤ演算は、入力画素Ｄｉｎの輝度値Yと、各々の参照ブロック内の入力画素Ｄｉｎの位置に対応した、ターゲットブロック内の画素の輝度値Ｙ

との差分絶対値を算出し、その算出した差分絶対値を、それぞれの参照ブロックに対応した参照ベクトルに従って、ＳＡＤテーブルに加算してゆくようにして行なう。

【０４６０】

例えば、入力画素Ｄｉｎが参照ブロック１０６１に属するとした場合における、ターゲットブロック１０３の画素Ｄ１と入力画素Ｄｉｎとの差分絶対値は、図５５に示すように、ＳＡＤテーブル１０８の参照ベクトル１０７１が対応するＳＡＤテーブル要素１０９１のＳＡＤ値に加算して書き込むようにする。また、入力画素Ｄｉｎが参照ブロック１０６２に属するとした場合における、ターゲットブロック１０３の画素Ｄ２と入力画素Ｄｉｎとの差分絶対値は、図５５に示すように、ＳＡＤテーブル１０８の参照ベクトル１０７２が対応するＳＡＤテーブル要素１０９２のＳＡＤ値に加算して書き込むようにする。

【０４６１】

したがって、サーチ範囲内の全ての領域の入力画素が入力されて処理が終了したときには、ＳＡＤテーブルが完成することになる。

【０４６２】

図５５の説明は、従来手法に、リアルタイムＳＡＤ算出処理を適用した場合である。この第２の実施形態においては、図５５において、ＳＡＤテーブル１０８の参照ベクトル１０７１または１０７２が対応するＳＡＤテーブル要素１０９１または１０９２のＳＡＤ値として、算出した差分絶対値のそれぞれを、加算して書き込むのではなく、前述した第１の実施形態のように、参照ベクトル１０７１，１０７２を縮小倍率１／ｎで縮小した参照縮小ベクトルを算出し、その参照縮小ベクトルの近傍の複数の参照ベクトルに、前記算出した差分絶対値から、それぞれを分散加算するための分散加算値を求め、求めた分散加算値を、前記近傍の複数の参照ベクトルに対応するＳＡＤ値に加算するようにするものである。

【０４６３】

ＳＡＤテーブル（縮小ＳＡＤテーブル）が完成した後の正確な動きベクトルを検出するための処理は、この第２の実施形態においても、前述した第１の実施形態で述べた手法と全く同様にして、２次曲面や、水平方向および垂直方向の３次曲線を用いた手法を用いることができる。

【０４６４】

この第２の実施形態における手ぶれ動きベクトル検出部１５における動きベクトルの検出処理動作のフローチャートを図５６および図５７に示す。

【０４６５】

先ず、手ぶれ動きベクトル検出部１５では、入力画像のフレーム（参照フレーム）の任意の位置（ｘ，ｙ）の画素データＤｉｎ（ｘ，ｙ）を受け取る（ステップＳ４０１）。次に、当該画素の位置（ｘ，ｙ）を含む複数の参照ブロックの一つに対応する参照ベクトル（ｖｘ，ｖｙ）を設定する（ステップＳ４０２）。

【０４６６】

次に、設定された参照ベクトル（ｖｘ，ｖｙ）の参照ブロックＩｉの当該画素値Ｉｉ（ｘ，ｙ）と、これに対応するターゲットブロックＩｏ内の画素値Ｉｏ（ｘ－ｖｘ，ｙ－ｖｙ）との差分の絶対値αを算出する（ステップＳ４０３）。すなわち、差分絶対値αは、

$$\alpha = |Io(x - vx, y - vy) - Ii(x, y)| \quad \cdots \text{（式３）}$$

として算出される。

【０４６７】

次に、縮小倍率を１／ｎとして、参照ベクトル（ｖｘ，ｖｙ）を１／ｎに縮小した参照縮小ベクトル（ｖｘ／ｎ，ｖｙ／ｎ）を算出する（ステップＳ４０４）。

【０４６８】

次いで、参照縮小ベクトル（ｖｘ／ｎ，ｖｙ／ｎ）の近傍の複数の参照ベクトル、この例では、上述したように４個の近傍参照ベクトルを検知する（ステップＳ４０５）。そして、検知した４個の近傍参照ベクトルのそれぞれに対応するテーブル要素として分散加算すべき値（差分絶対値）を、前述したように、参照縮小ベクトルと近傍参照ベクトルとが

それぞれ示す位置の関係に基いて、ステップＳ４０３で求めた差分絶対値αから、線形加重分散値として求める（ステップＳ４０６）。そして、求めた４個の線形加重分散値を、近傍参照ベクトルのそれぞれに対応するＳＡＤテーブル要素値に加算する（ステップＳ４０７）。

【０４６９】

次に、入力画素Ｄｉｎ（ｘ，ｙ）を含む参照ブロックの全てについての上記ステップＳ４０２～ステップＳ４０７の演算を行なったか否か判別し（ステップＳ４０８）、当該入力画素Ｄｉｎ（ｘ，ｙ）を含む他の参照ブロックがあると判別したときには、ステップＳ４０２に戻り、当該入力画素Ｄｉｎを含む他の参照ブロック（ｖｘ，ｖｙ）を設定し、このステップＳ４０２～ステップＳ４０７の処理を繰り返す。

【０４７０】

また、ステップＳ４０８で、入力画素Ｄｉｎ（ｘ，ｙ）を含む参照ブロックの全てについての上記ステップＳ４０２～ステップＳ４０７の演算を行なったと判別したときには、サーチ範囲内の全ての入力画素Ｄｉｎについて、上記の演算ステップの処理を終了したか否か判別し（図５７のステップＳ４１１）、終了していないと判別したときには、ステップＳ４０１に戻り、サーチ範囲内の次の入力画素Ｄｉｎを取り込み、このステップＳ４０１以降の処理を繰り返す。

【０４７１】

そして、ステップＳ４１１で、サーチ範囲内の全ての入力画素Ｄｉｎについて、上記の演算ステップの処理を終了したと判別すると、縮小ＳＡＤテーブルが完成したとして、当該完成した縮小ＳＡＤテーブルにおいて、最小値となっているＳＡＤ値を検出する（ステップＳ４１２）。

【０４７２】

次に、当該最小値となっているテーブル要素アドレス（ｍｘ，ｍｙ）のＳＡＤ値（最小値）と、その近傍の複数個、この例では、上述したように１５個の近傍テーブル要素のＳＡＤ値を用いて２次曲面を生成し（ステップＳ４１３）、その２次曲面の最小値のＳＡＤ値が対応する小数精度の位置を示す最小値ベクトル（ｐｘ，ｐｙ）を算出する（ステップＳ４１４）。この最小値ベクトル（ｐｘ，ｐｙ）は、小数精度の最小テーブル要素アドレスに対応している。

【０４７３】

そして、算出した小数精度の位置を示す最小値ベクトル（ｐｘ，ｐｙ）をｎ倍することにより、求めるべく動きベクトル（ｐｘ×ｎ，ｐｙ×ｎ）を算出する（ステップＳ４１５）。

【０４７４】

なお、この例においても、小数精度の位置を示す最小値ベクトル（ｐｘ，ｐｙ）を算出する方法としては、前述した水平方向および垂直方向の３次曲線を用いる方法を用いても良いことは前述の例と同様である。

【０４７５】

また、前述の第１の実施形態の第３の例と同様にして、この第２の実施形態においても、サーチ範囲を絞りながら、かつ、必要に応じて縮小倍率を変更しながら、２段階以上、縮小ＳＡＤテーブルを用いた動きベクトル検出処理を繰り返すようにしても、勿論良い。

【０４７６】

この第２の実施形態のメリットは、フレームメモリを、第１の実施形態に比べて１枚分削減できることと、フレームメモリに入力画像を格納する時間を短縮できることである。メモリ削減の効果は言うまでもないが、処理時間の短縮も、近年、重要視されて来ている。特に動画を扱う場合、そのままシステム遅延の短縮に繋がるため、システム遅延が原因で生じる、実際の被写体とパネル表示画像の間に生じる違和感をなるべく無くすことは、ユーザへの訴求効果が高い。

【０４７７】

［その他の実施形態および変形例］

上述の実施の形態では、分割画像区間のそれぞれでは、水平方向に領域を複数個に分割してそれぞれの分割領域にターゲットブロックを設定し、それぞれのターゲットブロックについて検出した動きベクトルの平均化処理により、それぞれの分割画像区間の手ぶれ動きベクトルを求めるようにしたが、分割画像区間のそれぞれについて一つのターゲットブロックを設定して、一つの動きベクトルを算出するようにしても勿論良い。

【０４７８】

また、上述の実施形態では、水平補正速度成分Ｘ＿ＳＴＢ＿＊および垂直補正速度成分Ｙ＿ＳＴＢ＿＊は、１水平ライン区間の時間長当たりの手ぶれ補正量として、水平手ぶれ補正量ＳＸ＿ＡＤＤおよび垂直手ぶれ補正量ＳＹ＿ＡＤＤの時間積分演算を、水平補正速度成分Ｘ＿ＳＴＢ＿＊および垂直補正速度成分Ｙ＿ＳＴＢ＿＊の単純加算とするようにしたが、それぞれの検出した速度成分と、各ラインの時間とを乗算するようにしても良いことは言うまでもない。

【０４７９】

また、上述の実施形態の説明は、撮像装置を操作するユーザにより生起される手ぶれによる画像歪みを補正する場合であったが、ユーザにより生起される手ぶれのみではなく、撮影時に、撮像素子に対して、その撮像画像の水平方向および／または垂直方向の位置的変化を生じるように加わった振動などの偏倚力に起因して生じる画像歪みをも補正することができることは言うまでもない。

【０４８０】

また、上述の実施形態は、Ｘ－Ｙアドレス型の固体撮像素子としてＣＭＯＳイメージャを用いた場合であるが、撮像素子としてはＣＭＯＳイメージャに限定されるものでないことは勿論である。

【０４８１】

また、この発明は、撮像装置（カメラ）にのみ適用されるものではなく、例えば携帯電話端末、情報携帯端末などに撮像素子が取付けられて、画像を撮影する場合にも適用できるものである。さらに、ユーザが手で持って撮影を行なう装置にのみに適用されるものではなく、パーソナルコンピュータやテレビ電話装置などの固定的に設置される装置に対して外力が加わって振動等が生じる場合、また、自動車などに撮像素子が取付けられて、画像を撮影する場合にも適用可能である。

【図面の簡単な説明】

【０４８２】

【図１】この発明による画像歪み補正方法の実施形態の概要を説明するための図である。

【図２】この発明による画像歪み補正方法の実施形態の概要の要部を説明するための図である。

【図３】ブロックマッチングにより動きベクトルを検出する処理を説明するための図である。

【図４】ブロックマッチングにより動きベクトルを検出する処理を説明するための図である。

【図５】ブロックマッチングにより動きベクトルを検出する処理を説明するための図である。

【図６】この発明の実施形態における動きベクトルを検出する処理の概要を説明するための図である。

【図７】この発明の実施形態における動きベクトルを検出する処理の概要を説明するための図である。

【図８】この発明の実施形態における動きベクトルを検出する処理の概要を説明するための図である。

【図９】この発明の実施形態における動きベクトルを検出する方法において、正確な動きベクトルを検出するための処理の第１の例を説明するための図である。

【図１０】この発明の実施形態における動きベクトルを検出する方法において、正確な動きベクトルを検出するための処理の第１の例を説明するための図である。

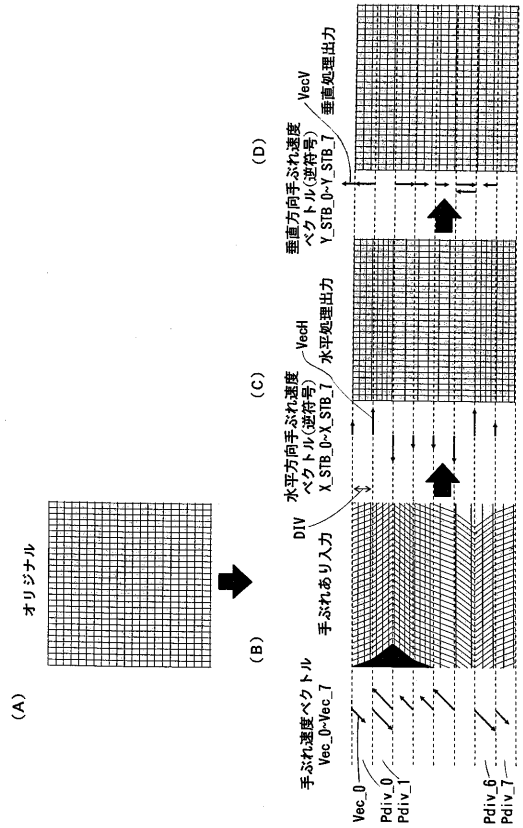
【図１１】この発明の実施形態における動きベクトルを検出する処理の概要を説明するための図である。
【図１２】この発明の実施形態における動きベクトルを検出する処理の概要を説明するための図である。
【図１３】この発明の実施形態における動きベクトルを検出する方法において、正確な動きベクトルを検出するための処理の第１の例を説明するための図である。
【図１４】この発明の実施形態における動きベクトルを検出する方法において、正確な動きベクトルを検出するための処理の第１の例を説明するための図である。
【図１５】この発明の実施形態における動きベクトルを検出する方法において、正確な動きベクトルを検出するための処理の第１の例を説明するための図である。
【図１６】この発明の実施形態における動きベクトルを検出する方法において、正確な動きベクトルを検出するための処理の第１の例を説明するための図である。
【図１７】この発明の実施形態における動きベクトルを検出する方法において、正確な動きベクトルを検出するための処理の第１の例を説明するための図である。
【図１８】この発明の実施形態における動きベクトルを検出する方法において、正確な動きベクトルを検出するための処理の第１の例を説明するための図である。
【図１９】この発明の実施形態における動きベクトルを検出する方法において、正確な動きベクトルを検出するための処理の第１の例を説明するための図である。
【図２０】この発明の実施形態における動きベクトルを検出する方法において、正確な動きベクトルを検出するための処理の第２の例を説明するための図である。
【図２１】この発明の実施形態における動きベクトルを検出する方法において、正確な動きベクトルを検出するための処理の第２の例を説明するための図である。
【図２２】この発明の実施形態における動きベクトルを検出する方法において、正確な動きベクトルを検出するための処理の第２の例を説明するための図である。
【図２３】この発明の実施形態における動きベクトルを検出する方法において、正確な動きベクトルを検出するための処理の第２の例を説明するための図である。
【図２４】この発明の実施形態における動きベクトルを検出する方法の処理性能を説明するための図である。
【図２５】この発明の実施形態における動きベクトルを検出する方法の特徴を、従来の手法と比較して説明するために用いる図である。
【図２６】この発明の実施形態における動きベクトルを検出する方法の特徴を、従来の手法と比較して説明するために用いる図である。
【図２７】この発明の実施形態における動きベクトルを検出する方法の特徴を、従来の手法と比較して説明するために用いる図である。
【図２８】この発明の実施形態における動きベクトルの検出方法を説明するための図である。
【図２９】この発明の実施形態における動きベクトルの検出処理の要部を説明するための図である。
【図３０】この発明による画像信号の歪み補正方法の実施形態が適用された撮像装置の第１の実施形態の構成例を示すブロック図である。
【図３１】この発明による撮像装置の第１の実施形態における動きベクトル検出処理の第１の例を説明するためのフローチャートの一部を示す図である。
【図３２】この発明による撮像装置の第１の実施形態における動きベクトル検出処理の第１の例を説明するためのフローチャートの一部を示す図である。
【図３３】この発明による撮像装置の第１の実施形態における動きベクトル検出処理の第２の例を説明するためのフローチャートの一部を示す図である。
【図３４】この発明による撮像装置の第１の実施形態における動きベクトル検出処理の第２の例を説明するためのフローチャートの一部を示す図である。
【図３５】この発明による撮像装置の第１の実施形態における動きベクトル検出処理の第２の例を説明するためのフローチャートの一部を示す図である。

【図３６】この発明による撮像装置の第１の実施形態における動きベクトル検出処理の第３の例を説明するためのフローチャートの一部を示す図である。
【図３７】この発明による撮像装置の第１の実施形態における動きベクトル検出処理の第３の例を説明するためのフローチャートの一部を示す図である。
【図３８】この発明による撮像装置の第１の実施形態における動きベクトル検出処理の第３の例を説明するためのフローチャートの一部を示す図である。
【図３９】図３０の撮像装置の第１の実施形態の一部のブロックの、より詳細構成例のブロック図である。
【図４０】図３０の撮像装置の第１の実施形態の一部のブロックの、より詳細構成例のブロック図である。
【図４１】図３０の撮像装置の第１の実施形態の一部のブロックの、処理動作例をセル瞑するためのフローチャートである。
【図４２】この発明による画像歪み補正方法の実施形態において、画像歪みを補正するために用いる補間処理を説明するための図である。
【図４３】この発明による画像歪み補正方法の実施形態において、画像歪みを補正するために用いる補間処理を説明するための図である。
【図４４】この発明による画像歪み補正方法の実施形態において、画像歪みを補正するために用いる補間処理を説明するための図である。
【図４５】実施形態の撮像装置における水平手ぶれ補正量積分部の動作の流れを説明のためのフローチャートである。
【図４６】実施形態の撮像装置における水平画像処理部の詳細構成例を示すブロック図である。
【図４７】図４６の水平画像処理部の動作タイミングを説明するための図である。
【図４８】実施形態の撮像装置における垂直手ぶれ補正量積分部の動作の流れを説明のためのフローチャートである。
【図４９】実施形態の撮像装置における垂直画像処理部の詳細構成例を示すブロック図である。
【図５０】図４９の垂直画像処理部の動作タイミングを説明するための図である。
【図５１】図３０の画像歪み補正装置の実施形態における要部の処理動作を説明するためのブロック図である。
【図５２】図３０の画像歪み補正装置の実施形態における信号処理部内のレジスタブロックにおけるレジスタ動作のタイミングを説明するための図である。
【図５３】この発明による画像歪み補正方法の実施形態が適用された撮像装置の第２の実施形態の構成例を示すブロック図である。
【図５４】撮像装置の第２の実施形態における動きベクトル検出処理を説明するための図である。
【図５５】撮像装置の第２の実施形態における動きベクトル検出処理を説明するための図である。
【図５６】撮像装置の第２の実施形態における動きベクトル検出処理の例を説明するためのフローチャートの一部を示す図である。
【図５７】撮像装置の第２の実施形態における動きベクトル検出処理の例を説明するためのフローチャートの一部を示す図である。
【図５８】従来の手ぶれ補正を説明するために用いる図である。
【図５９】ＣＭＯＳイメージャにおける、手ぶれによるフレーム内画像歪みの原因を説明するための図である。
【図６０】ＣＭＯＳイメージャにおける、手ぶれによるフレーム内画像歪みを説明するために用いる図である。
【図６１】従来のＣＭＯＳイメージャについての、手ぶれによるフレーム内画像歪みの補正方法を説明するための図である。
【符号の説明】

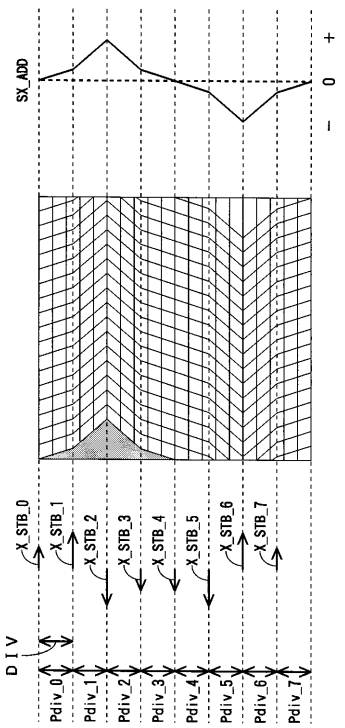
【０４８３】

１０１…元フレーム、１０２…参照フレーム、１０３…ターゲットブロック、１０５…サーチ範囲、１０６…参照ブロック、１０７…参照ベクトル、１５…手ぶれ動きベクトル検出部、１６…歪み補正解像度変調幹部、４１～４３…フレームメモリ、ＴＢＬｓ…縮小ＳＡＤテーブル、ＴＢＬｏ…従来のＳＡＤテーブル、ＲＶ…参照ベクトル、ＣＶ…参照縮小ベクトル、ＮＶ１～ＮＶ４…近傍参照ベクトル

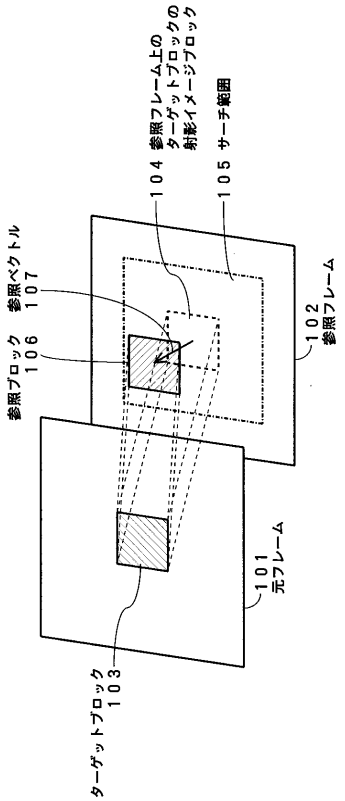
【図１】



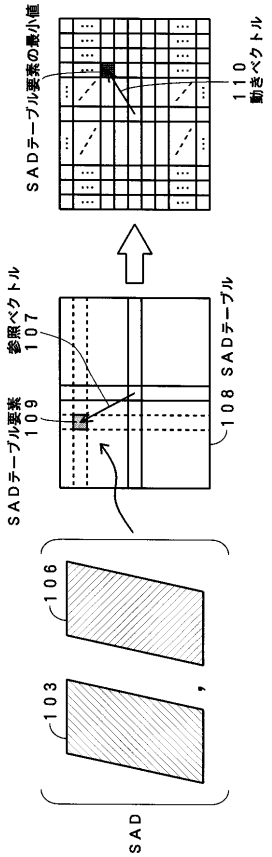
【図２】



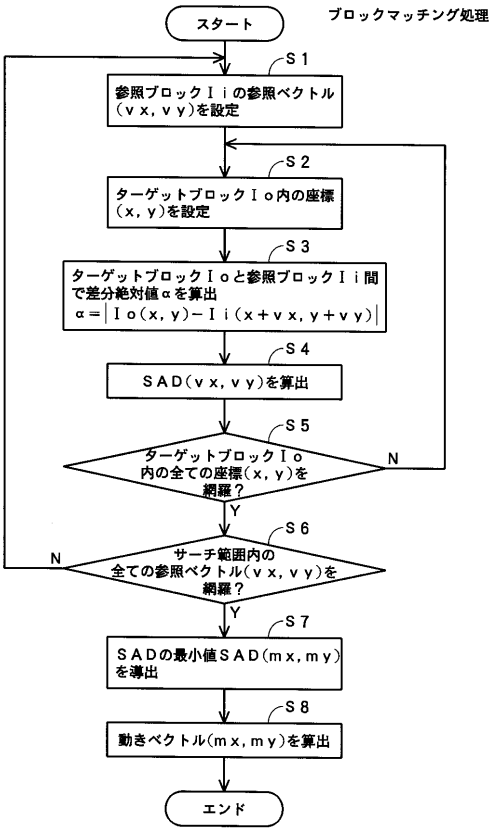
【図３】


１０４ 参照フレーム上のターゲットブロックの射影イメージブロック
１０５ サーチ範囲
参照ベクトル １０７
参照ブロック １０６
１０２ 参照フレーム
１０１ 元フレーム
ターゲットブロック １０３

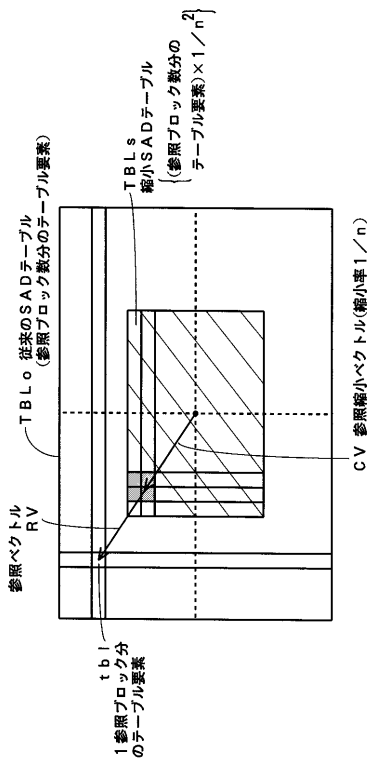
【図４】


SADテーブル要素の最小値
１１０ 動きベクトル
参照ベクトル １０７
SADテーブル要素 １０９
１０８ SADテーブル
１０６
１０３
SAD

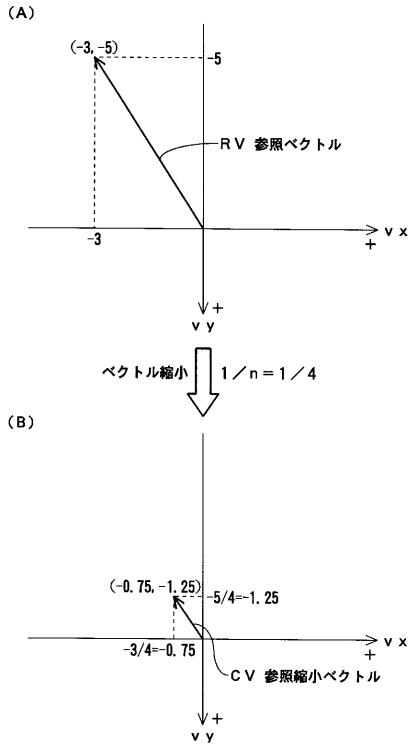
【図５】


ブロックマッチング処理
スタート
S１
参照ブロックＩｉの参照ベクトル(ｖｘ,ｖｙ)を設定
S２
ターゲットブロックＩｏ内の座標(ｘ,ｙ)を設定
S３
ターゲットブロックＩｏと参照ブロックＩｉ間で差分絶対値αを算出
α＝|Ｉｏ(ｘ,ｙ)－Ｉｉ(ｘ＋ｖｘ,ｙ＋ｖｙ)|
S４
ＳＡＤ(ｖｘ,ｖｙ)を算出
S５
ターゲットブロックＩｏ内の全ての座標(ｘ,ｙ)を網羅？
N
Y
S６
サーチ範囲内の全ての参照ベクトル(ｖｘ,ｖｙ)を網羅？
N
Y
S７
ＳＡＤの最小値ＳＡＤ(ｍｘ,ｍｙ)を導出
S８
動きベクトル(ｍｘ,ｍｙ)を算出
エンド

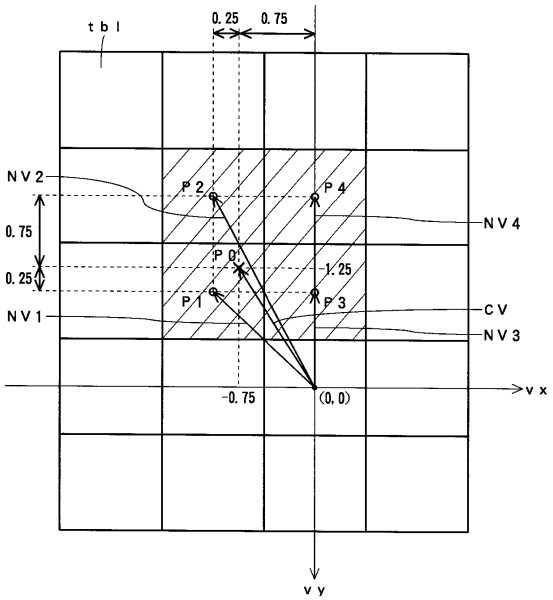
【図６】


ＴＢＬｓ 縮小ＳＡＤテーブル {(参照ブロック数分のテーブル要素)×1／n²}
ＴＢＬｏ 従来のＳＡＤテーブル (参照ブロック数分のテーブル要素)
ＣＶ 参照縮小ベクトル(縮小率1／n)
参照ベクトル RV
ｔｂｌ 1参照ブロック分のテーブル要素

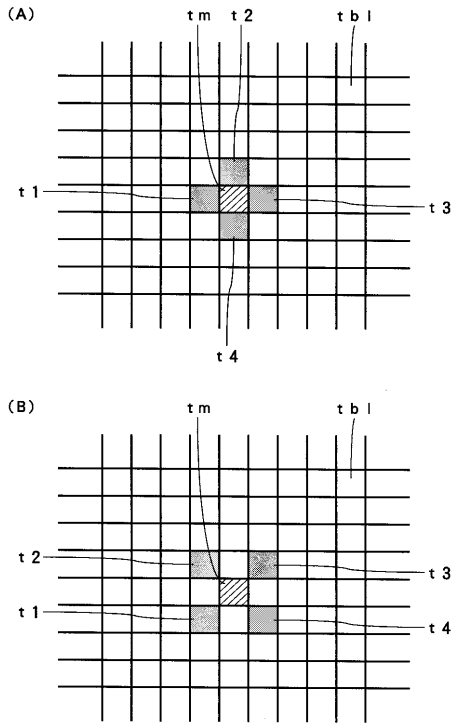
【図７】


(A)
(-3, -5)
-5
RV 参照ベクトル
-3
v x
+
v y
+
ベクトル縮小
1／n＝1／4
(B)
(-0.75, -1.25)
-5/4=-1.25
-3/4=-0.75
v x
+
CV 参照縮小ベクトル
v y
+

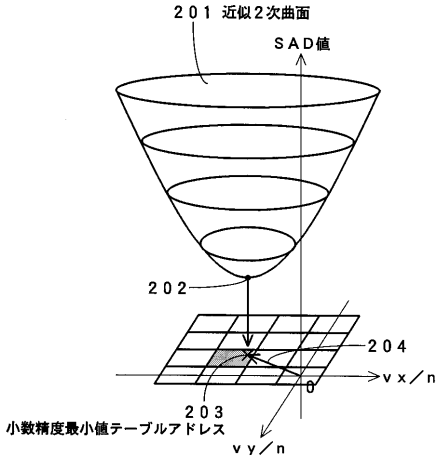
【図８】


t b l
0.25
0.75
NV2
P 2
P 4
NV4
0.75
P 0
0.25
1.25
P 1
P 3
CV
NV1
NV3
v x
-0.75
(0,0)
v y

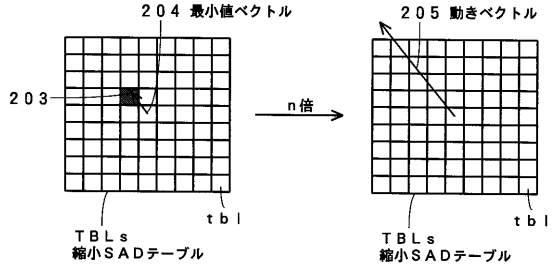
【図９】


(A)
t m
t 2
t b l
t 1
t 3
t 4
(B)
t m
t b l
t 2
t 3
t 1
t 4

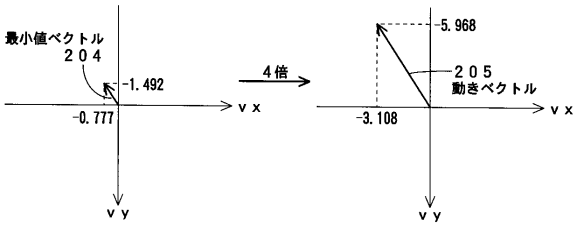
【図１０】


２０１ 近似２次曲面
SAD値
２０２
２０４
v x／n
０
２０３
小数精度最小値テーブルアドレス
v y／n

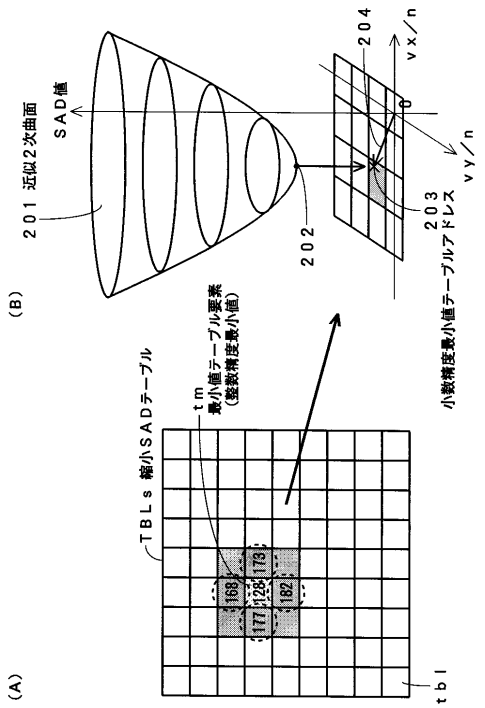
【図１１】



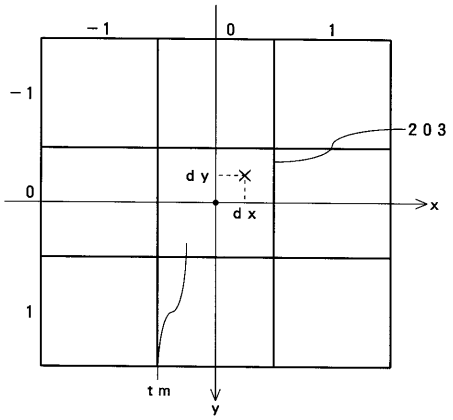
【図１２】



【図１３】



【図１４】



【図１５】

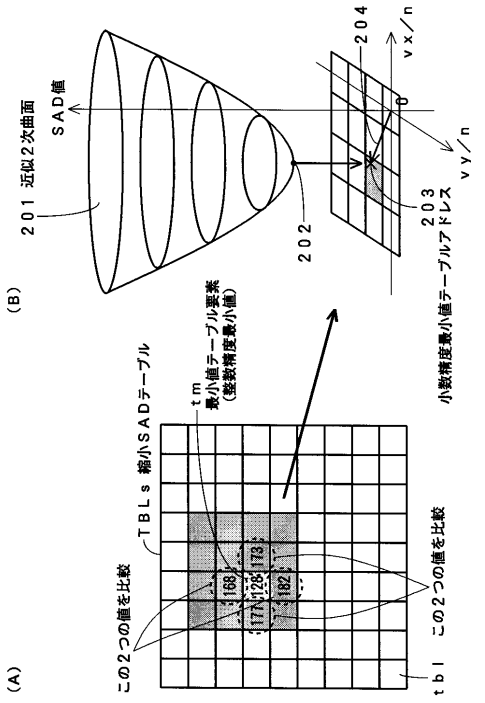
$$dx = \frac{\sum (Sxy \cdot Kx)}{4\sum Sxy - 6\sum (Sxy \cdot Kx^2)} \quad \cdots (式A)$$

$$dy = \frac{\sum (Sxy \cdot Ky)}{4\sum Sxy - 6\sum (Sxy \cdot Ky^2)} \quad \cdots (式B)$$

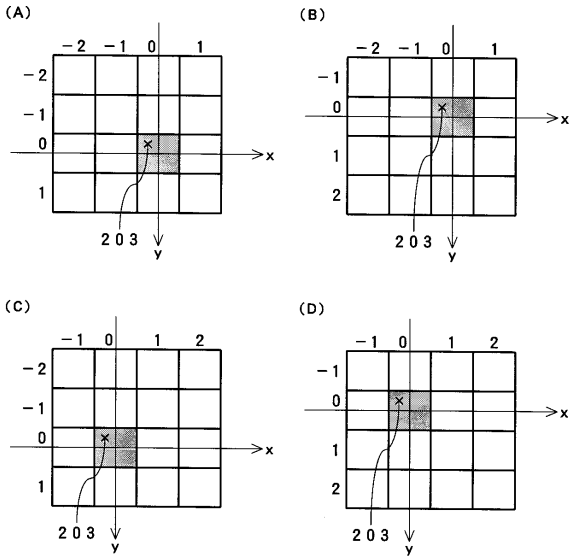
ただし

ｘ＝－１のときＫｘ＝－１	ｙ＝－１のときＫｙ＝－１
ｘ＝０のときＫｘ＝０	ｙ＝０のときＫｙ＝０
ｘ＝１のときＫｘ＝１	ｙ＝０のときＫｙ＝１

【図１６】



【図１７】

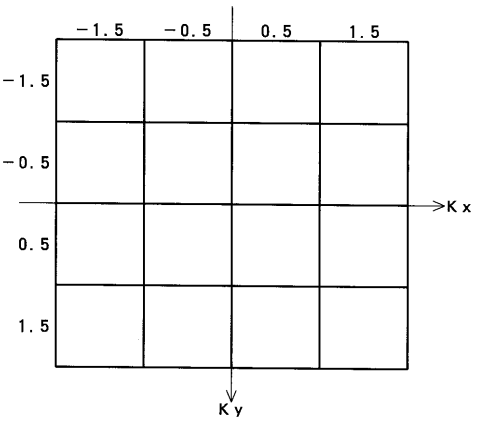


【図１８】

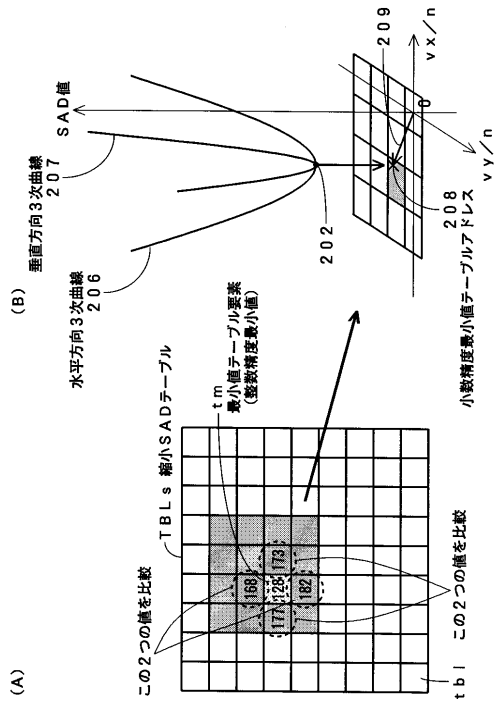
$$dx = \frac{8\sum(Sxy \cdot Kx)}{25\sum Sxy - 20\sum(Sxy \cdot Kx^2)} + \Delta x \quad \cdots (式C)$$

$$dx = \frac{8\sum(Sxy \cdot Ky)}{25\sum Sxy - 20\sum(Sxy \cdot Ky^2)} + \Delta y \quad \cdots (式D)$$

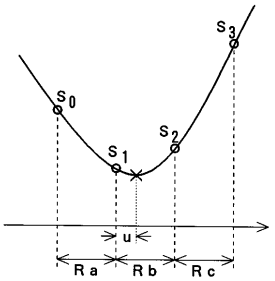
【図１９】



【図２０】



【図２１】



【図２２】

区間Ra

$$u = \frac{2S_3 - 11S_2 + 16S_1 - 7S_0 + \sqrt{(4S_3 - 7S_2 + 2S_1 + S_0)^2 + 15(S_3 - 3S_2 + 3S_1 - S_0)(S_1 - S_3)}}{3(S_3 - 3S_2 + 3S_1 - S_0)} \quad \cdots (式E)$$

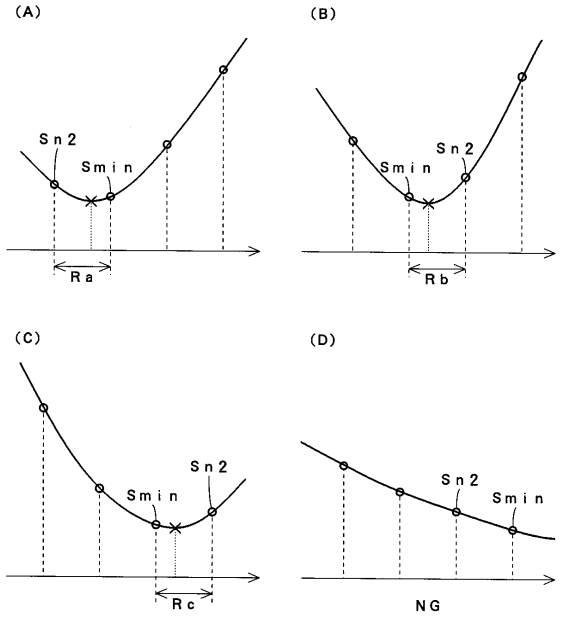
区間Rb

$$u = \frac{S_3 - 4S_2 + 5S_1 - 2S_0 + \sqrt{(S_3 - 5S_2 + 5S_1 - S_0)^2 + (S_2 - S_0)(S_1 - S_3)}}{3(S_3 - 3S_2 + 3S_1 - S_0)} \quad \cdots (式F)$$

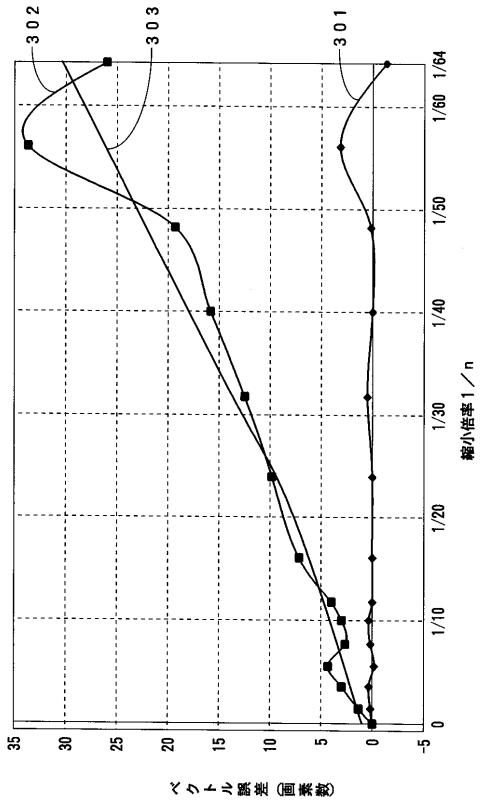
区間Rc

$$u = \frac{-4S_3 + 7S_2 - 2S_1 - S_0 + \sqrt{(4S_3 - 7S_2 + 2S_1 + S_0)^2 + 15(S_3 - 3S_2 + 3S_1 - S_0)(S_1 - S_3)}}{3(S_3 - 3S_2 + 3S_1 - S_0)} \quad \cdots (式G)$$

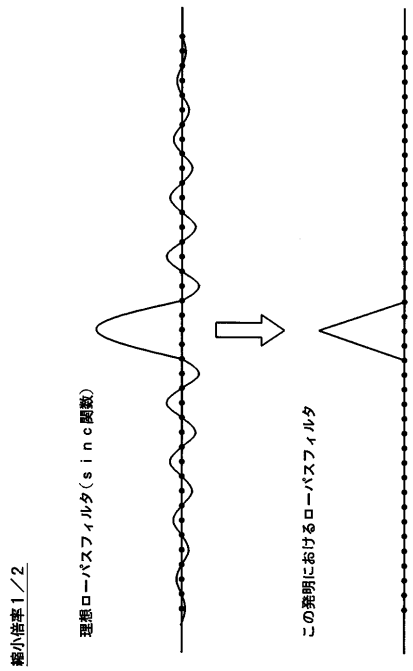
【図２３】



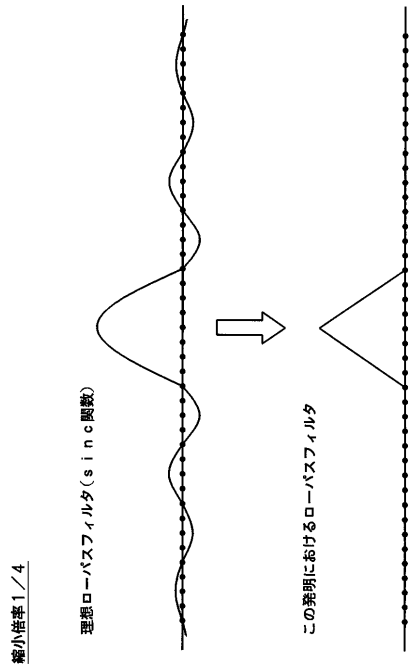
【図２４】



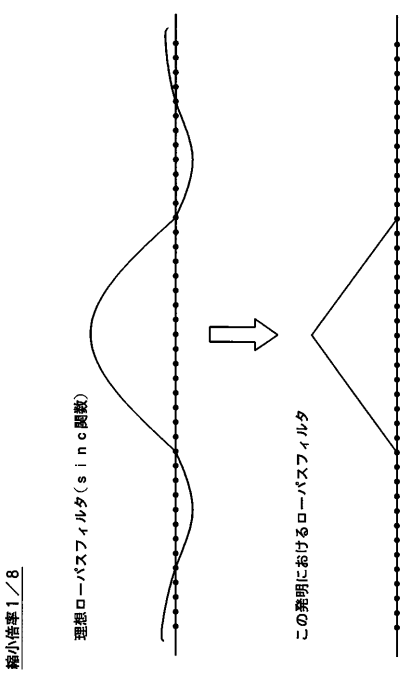
【図２５】



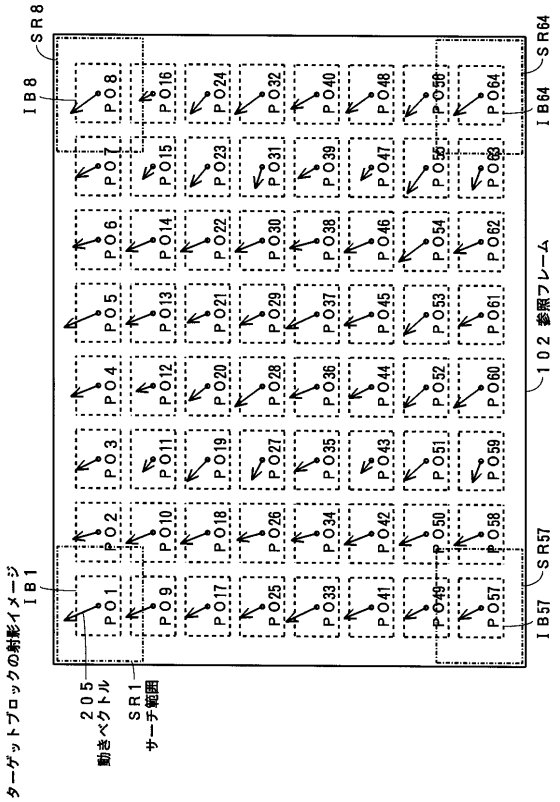
【図２６】



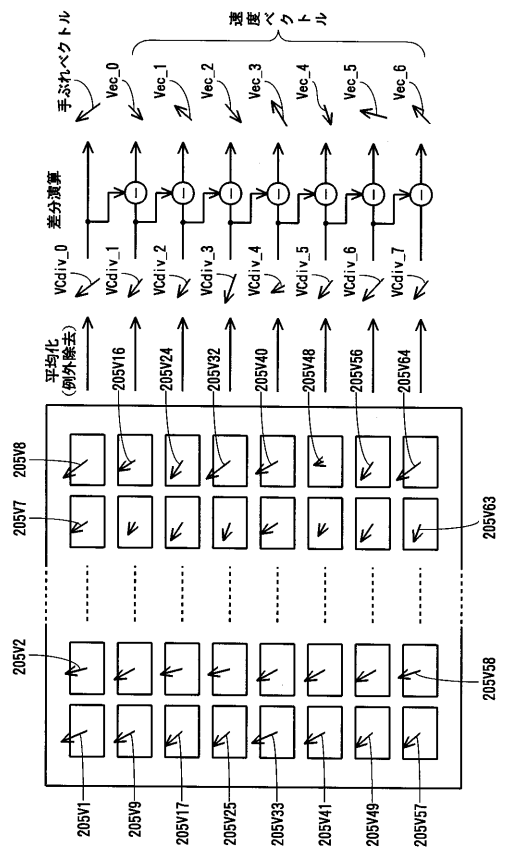
【図２７】



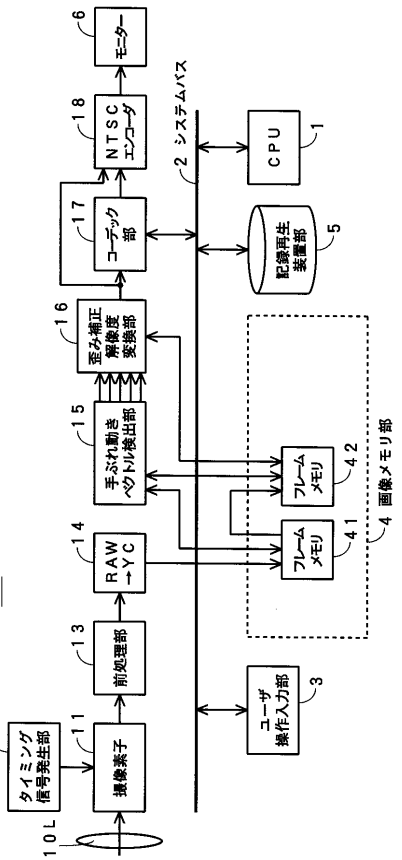
【図２８】



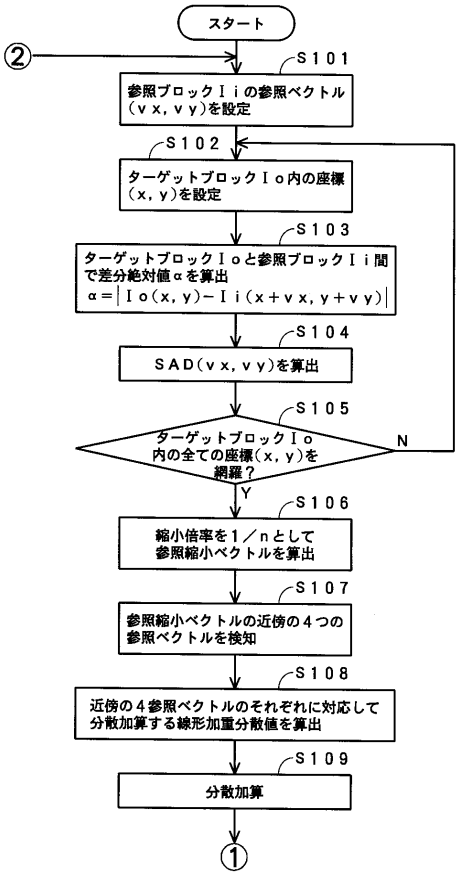
【図２９】



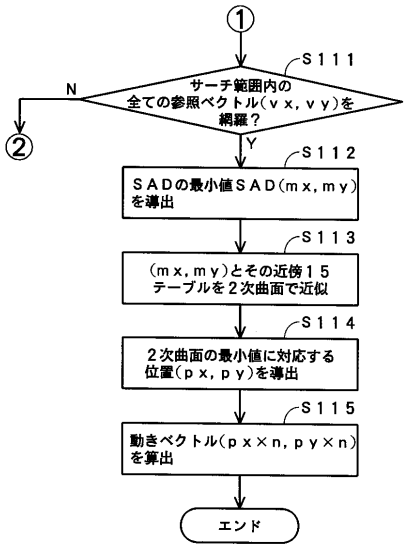
【図３０】



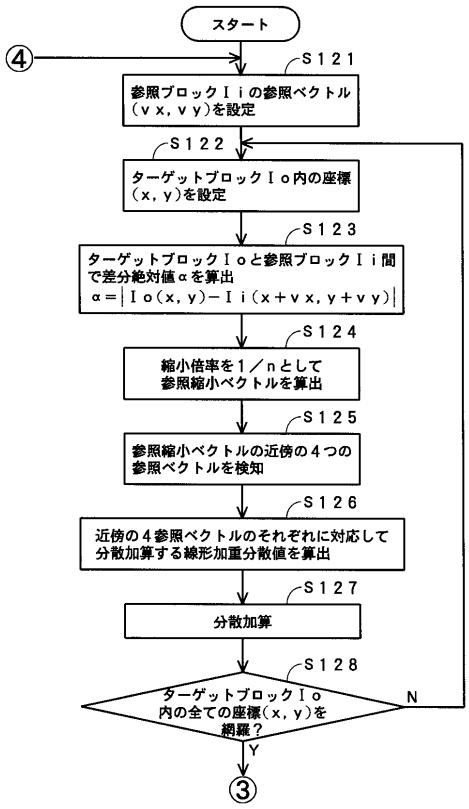
【図３１】



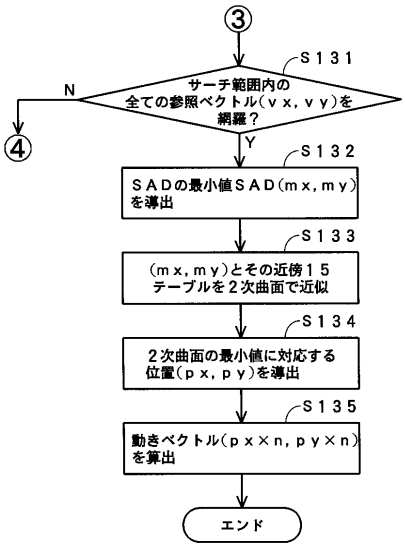
【図３２】



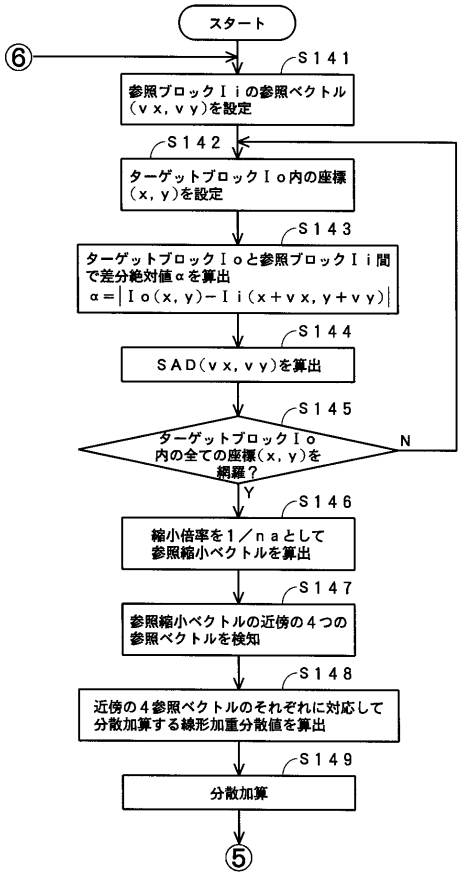
【図３３】



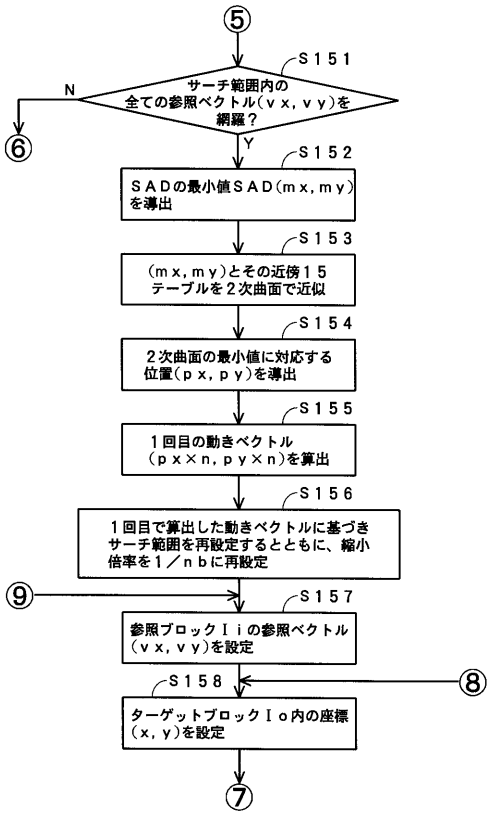
【図３４】



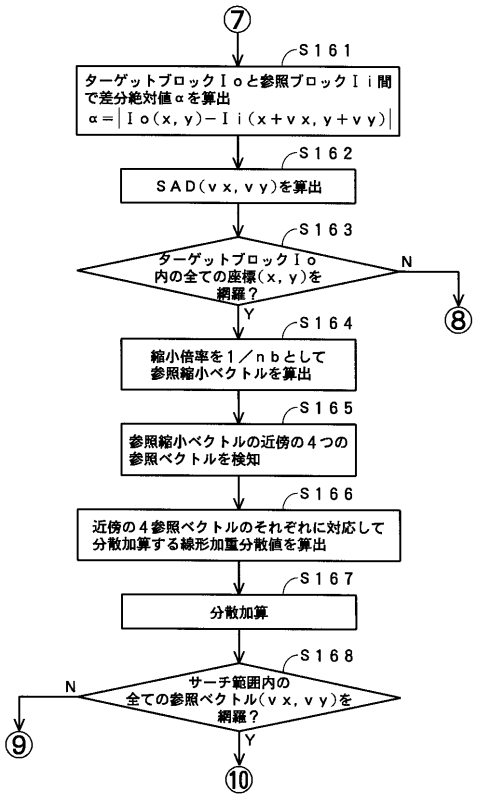
【図３５】



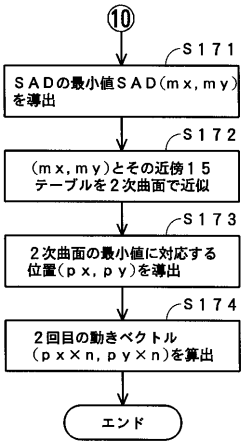
【図３６】



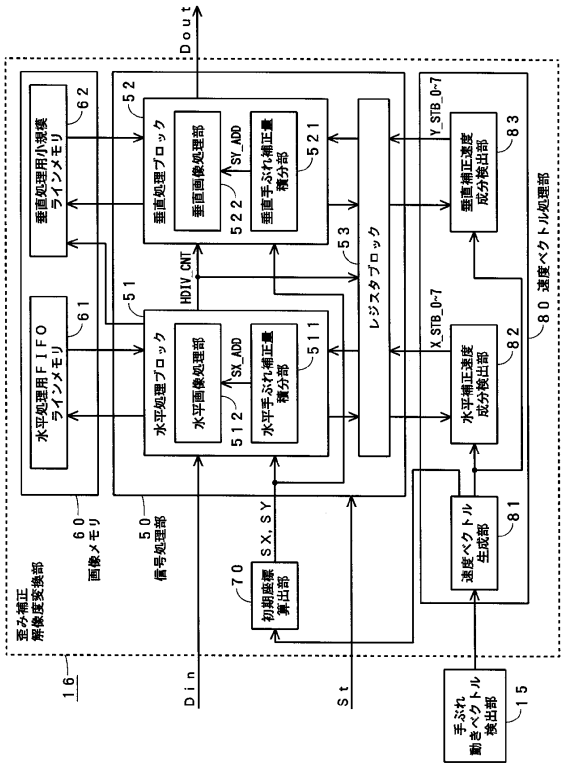
【図３７】



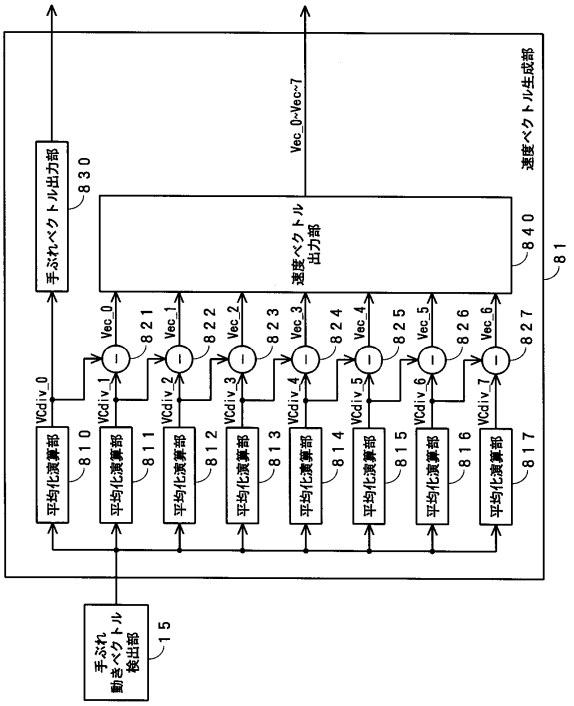
【図３８】



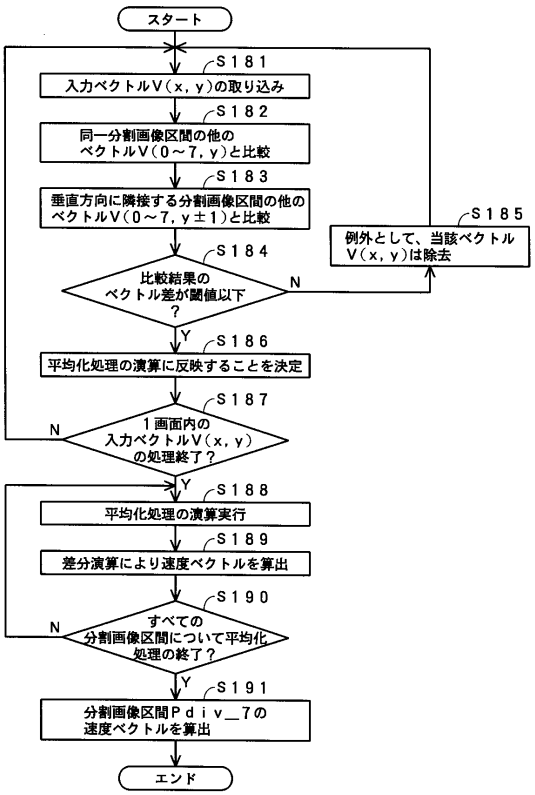
【図３９】



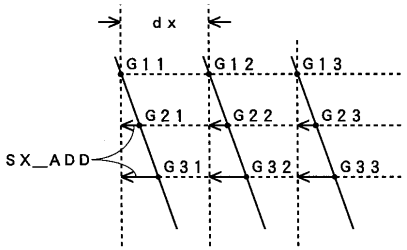
【図４０】



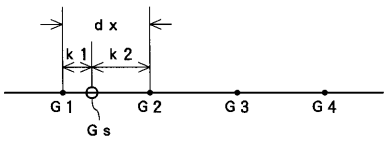
【図４１】



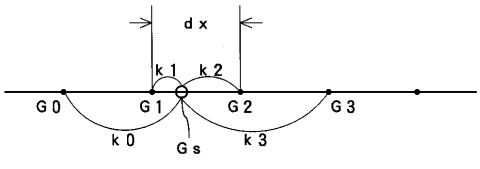
【図４２】



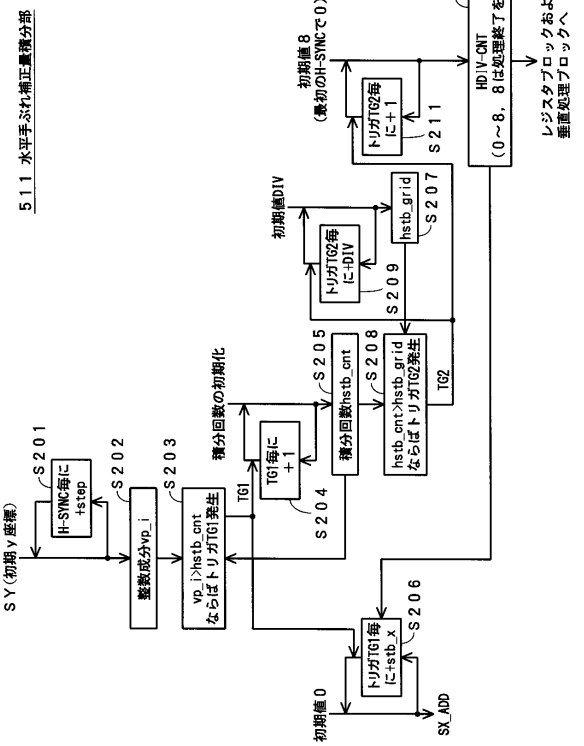
【図４３】



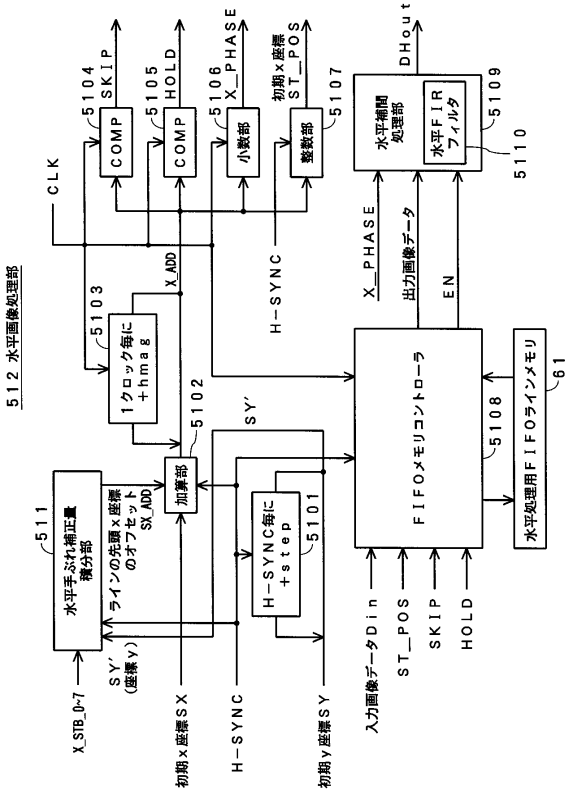
【図４４】



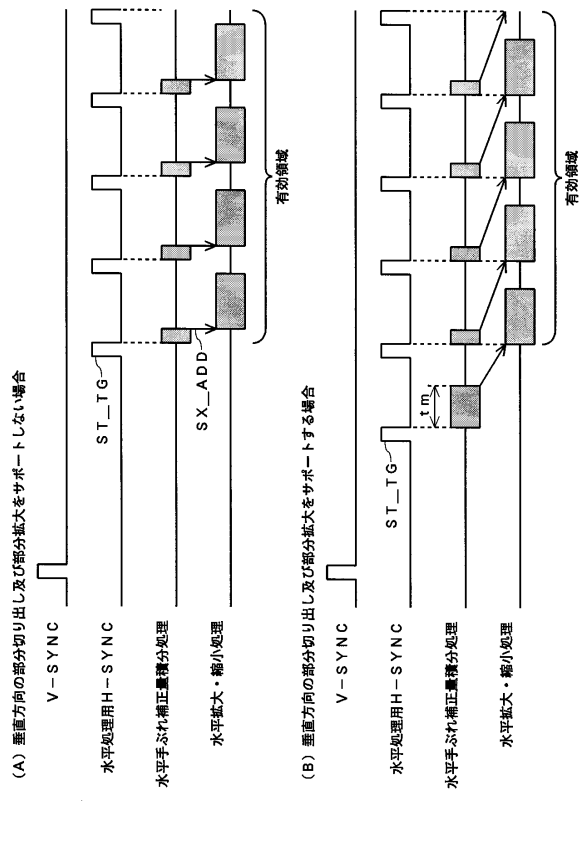
【図４５】



【図４６】

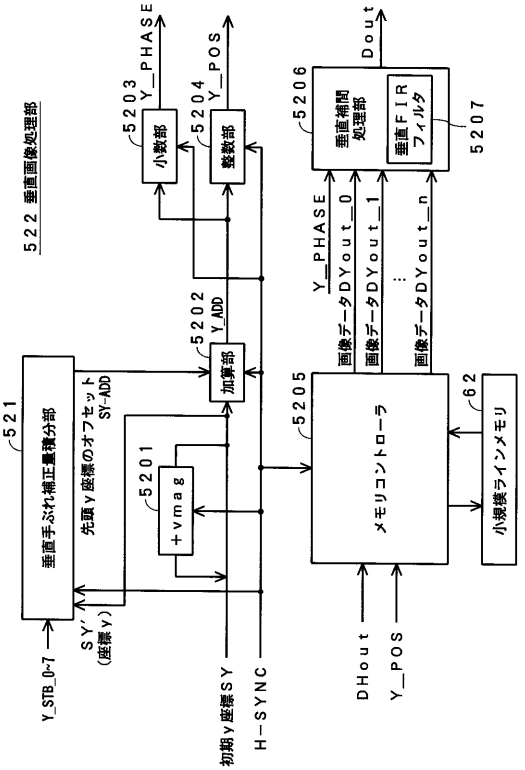


【図４７】

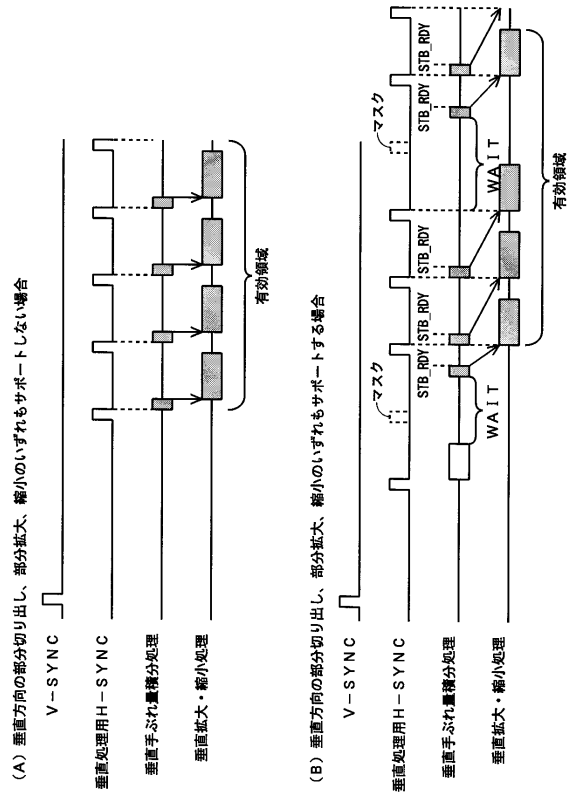


【図４８】

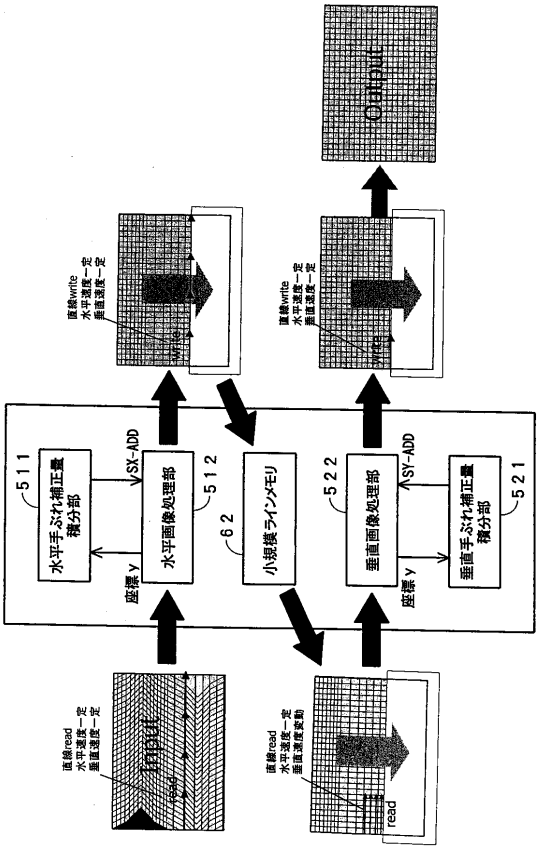
【図４９】



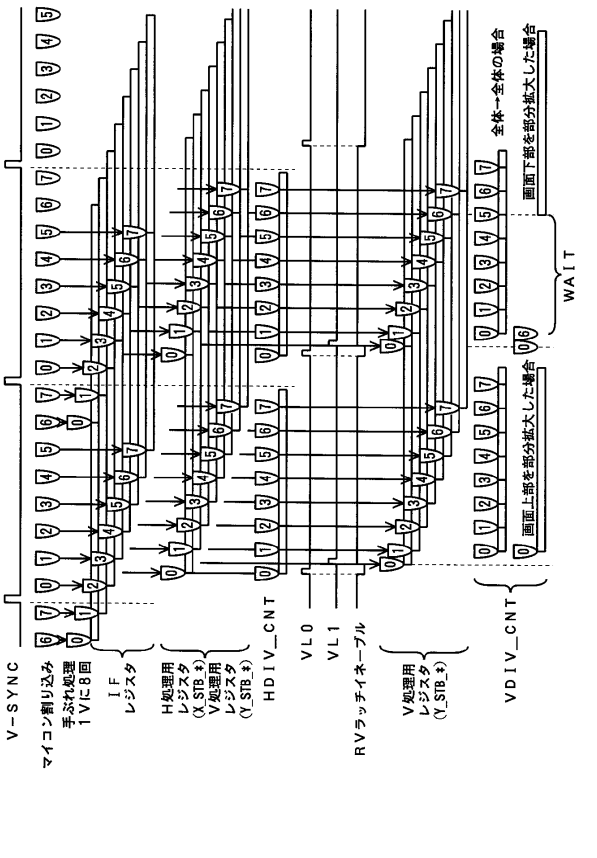
【図５０】



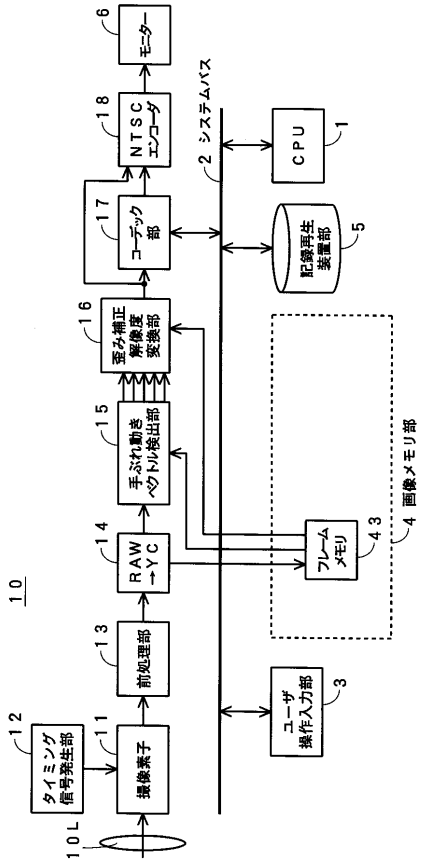
【図５１】



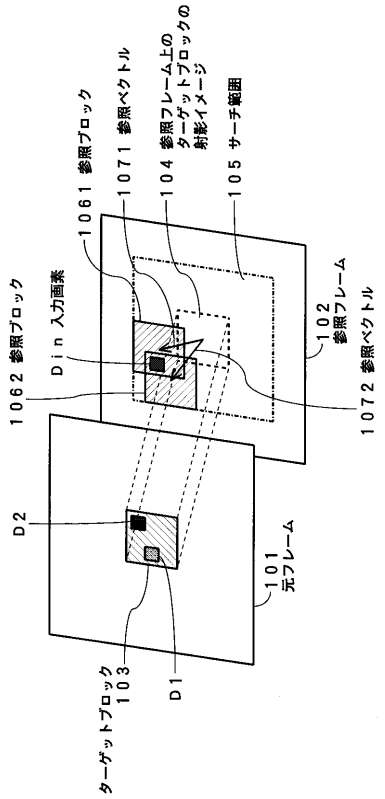
【図５２】



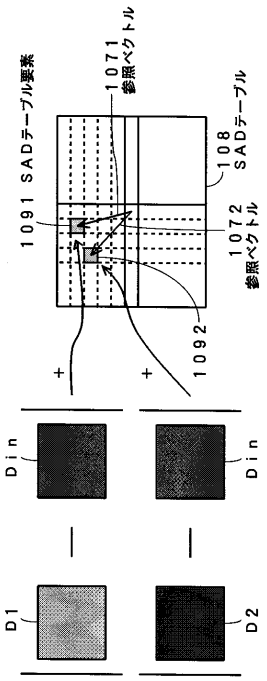
【図５３】



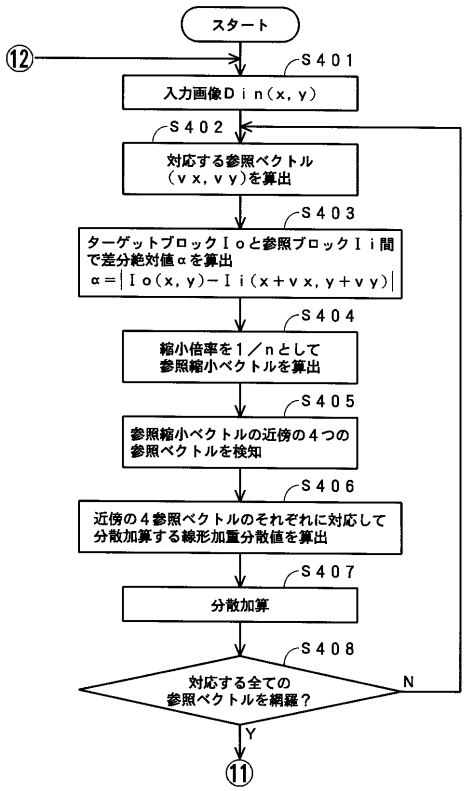
【図５４】



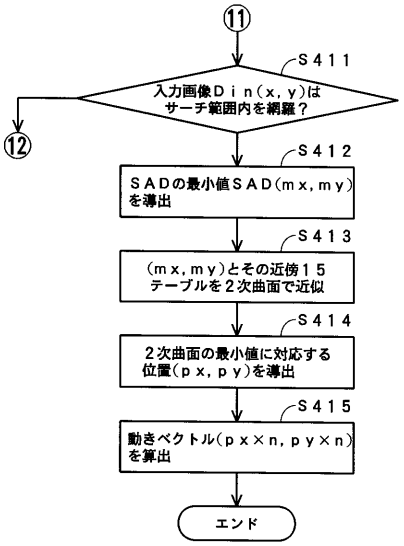
【図５５】



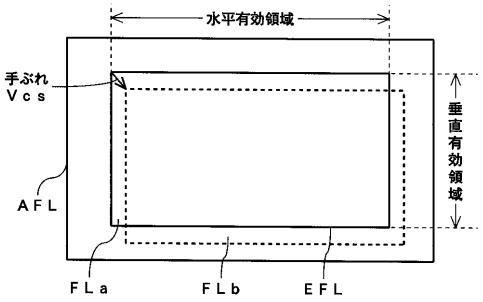
【図５６】



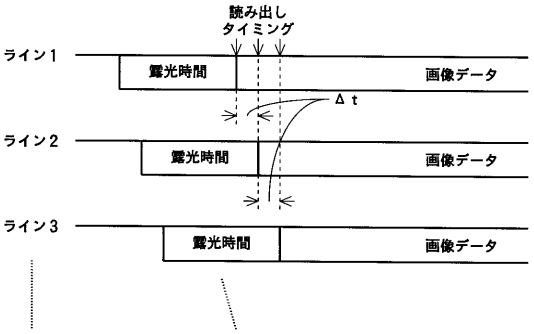
【図５７】



【図５８】

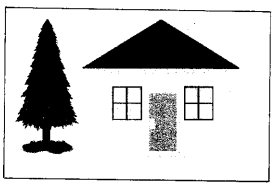


【図５９】

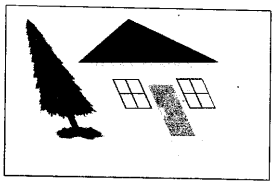


【図６０】

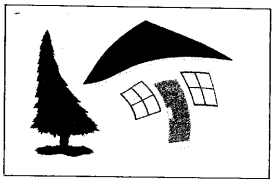
（Ａ）本来の撮像イメージ



（Ｂ）走る電車の中から外の景色を撮影した場合



（Ｃ）（ＣＭＯＳイメージャでの）手ぶれが発生した場合



【図６１】