2J-2

SW/HW 協調処理による船舶自動見張りシステム –航海画像安定化処理の高速化に関する初期検討–

植竹 大地 † 大川 猛 † 松本 洋平 †† 横田 隆史 † 大津 金光 †
† 宇都宮大学大学院工学研究科 †† 東京海洋大学海洋工学部

1 はじめに

海上交通において，主な事故の原因は見張り不十分による人為的要因である．船舶の見張りは 1 日 3 交代制で 24 時間行なわれるため，人的負担が大きい．そのため，海上での事故を減らすために，コンピュータビジョンによる船舶監視システムの導入が望まれている．

船舶監視システムは船に搭載したカメラから画像を取得し，取得した画像から船を認識して船の位置推定を行なう．しかし，船に搭載したカメラが取得した画像には動揺があるため，船の認識処理を高速に行なうことができない問題がある．そのため，認識処理を高速に行なうには，カメラからの入力画像に対して画像の動揺を取り除く安定化処理を行なう必要がある．

先行研究 [2] による安定化処理は，水平画素 1,024，垂直画素 128，総画素数 13 万画素の画像に対して，画像一枚あたり 83ms の処理であった．13 万画素の画像では遠方の船を認識することができないので，遠方の船を認識するために必要な画素数は，先行研究の画像に対して縦横の画素数を 4 倍にした 209 万画素が必要である．また，海の風景は劇的に変化することがないので 10frame/sec で処理を行なうものとする．209 万画素は，先行研究の総画素数の 16 倍であるため，ソフトウェアによる処理では 10frame/sec で処理することができない．

本稿は安定化処理の時間を分析して時間のかかる処理を FPGA のハードウェア処理により高速に処理する方法について検討を行なう．

2 安定化処理プログラムの分析

FPGA 化を検討するにあたり，[2] の安定化処理のプログラムの分析を行なった．安定化処理のプログラムは，カメラからの入力画像に対して，グレイフィルタ，ガウスフィルタ，安定化フィルタを画像に順番に適用することで，動揺を取り除いた画像の生成を行なう．

画像の安定化方法として，[2] の安定化プログラムでは逆合成法のアルゴリズム用いて処理を行なう．逆合成法は Lucas-Kanade 法 [3] と比較するとヘシアンを繰り返し計算する必要がないため計算量が少ない点で優れている．

安定化処理の時間がかかる処理を分析するために

表 1: pgomgr での測定結果

関数名	total	run total
c_imgalign::rgd_itr()	36.9%	36.9%
c_imgalign::calc_H_rgd()	21.0%	57.9%
c_imgalign::get_warped_pix_val()	17.8%	75.7%
cv::MatConstIterator::operator++()	5.3%	81.0%
c_imgalign::calc_warp()	5.2%	86.2%

Microsoft Visual Studio Professional 2010 のプロファイリングツールである pgomgr を使用した．プロファイル結果を表 1 に示す．関数名はプログラム内で使用している関数の名前である．total は全体における処理時間の割合である．また，run total は累積値となっており，各関数の処理時間を加算した値である．

それぞれ，c_imgalign::rgd_itr() には 36.9%，c_imgalign::calc_H_rgd() には 24.0%の時間がかかる．上位 2 つの関数は安定化フィルタ内で動作しており，この処理に全体の約 6 割の時間がかかっている．そのため，処理時間が長い 2 つの関数を高速化の対象とする．

rgd_itr() はワープ関数の値を求める演算を行なっている．ワープ関数とは，テンプレート画像から平行移動・回転を行なった画像を生成する関数である．また関数 calc_H_rgd() は，安定化処理の中でヘシアンの演算を行なう．

安定化処理を行なう関数のフローチャートを図 1 に示す．まず，ヘシアンの演算を行なう calc_H_rgd() の関数の処理をしたあとで，rgd_itr() の処理を行なう．rgd_itr() は 4 重ループを含んでいる．

外側 2 つのループはワープ先の画素をスキャンする処理であり，内側 2 つのループはワープ先の画素値の計算に必要なワープ元の画像をスキャンする処理である．画素の位置設定を行なうループはテンプレート領域の画素数回分ループが回るので，画素数分の並列度がある．rgd_itr() の計算結果が収束条件を満たせば処理が終了するが，収束条件を満たさない場合はもう一度 rgd_itr() の処理を行なう．

上位二つの関数は，calc_H_rgd() の関数で求めたヘシアンを元に，rgd_itr() が収束するまで処理を行なう．

1 フレームあたりの安定化処理の実行時間の総和を図 2 に示す．一番時間のかかる関数は rgd_itr() である．それぞれの関数の実行時間は，一定でないことがわかる．実行時間が一定でない理由として，rgd_itr() の関数の収束するまでの実行回数に差があるためである．その結果，それぞれの関数は 1 フレームあたり rgd_itr()

Automatic Watch System for Ship by using HW/SW Co-operative Processing –Preliminary Study on High-Speed Stabilization of Navigation Images–
†Daichi Uetake, Takeshi Ohkawa, Takashi Yokota and Kanemitsu Ootsu
††Yohei Matsumoto
Graduate School of Engineering, Utsunomiya University (†)
Tokyo University of Marine Science and Technology (††)

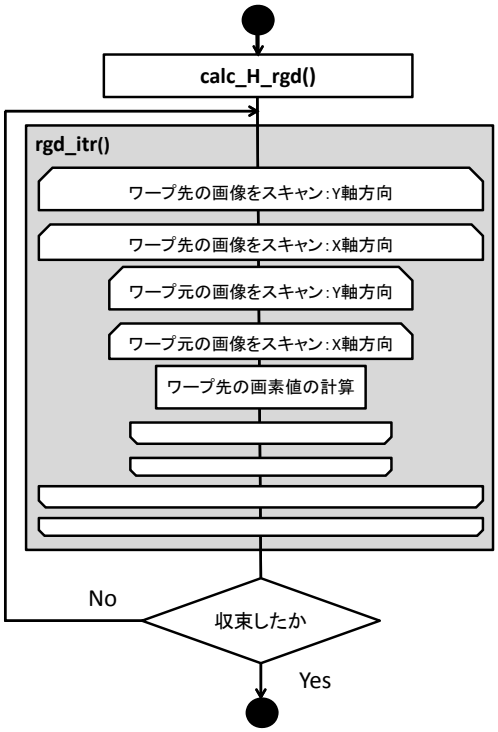

図 1: 安定化処理のフローチャート

関数で平均 8.6 回，calc_H_rgd() 関数で平均 3.0 回実行する．

上記の分析の結果より，1 フレームあたりの実行時間が多い rgd_itr() を FPGA を用いてハードウェア化することによって，安定化処理全体の実行時間を短くすることが期待できる．そのため，時間がかかっている関数である rgd_itr() についての FPGA を用いた高速化手法を議論する．

3 FPGA 安定化処理の開発方針

安定化処理の一部を FPGA 化により高速に処理を行なう．FPGA でハードウェアを開発するには，ハードウェア記述言語を使用しなければならないため設計生産性が低い問題がある．そのため，我々は高位合成手法を用いて開発を行なう．

FPGA 安定化処理システムの動作環境として，Xilinx 社製の Zynq-7000 を搭載ボードである ZedBoard[1] を選定した．Zynq-7000 は ARM のコアと FPGA がひとつのチップ上にあるので，ソフトウェアとハードウェアの処理を効率よく行なうことが可能である．

ループで処理をする rgd_itr() をパイプライン処理することで高速化を図ろうとしたが，rgd_itr() は前の出力結果を用いて，次の演算を行なうのでパイプライン実行することができない．そのため，rgd_itr() 内の処理を並列化する必要がある．

高速化の対象とした rgd_itr() に関して 4 重ループ内のデータ依存を分析した結果，データの依存がないため，処理に用いるテンプレート領域画素分の並列度があることがわかった．テンプレート領域の画素数は 3 パターンあり，1024*64，512*32，256*16 である．テンプレート領域は，[2] から収束率の高かったものを使用した．

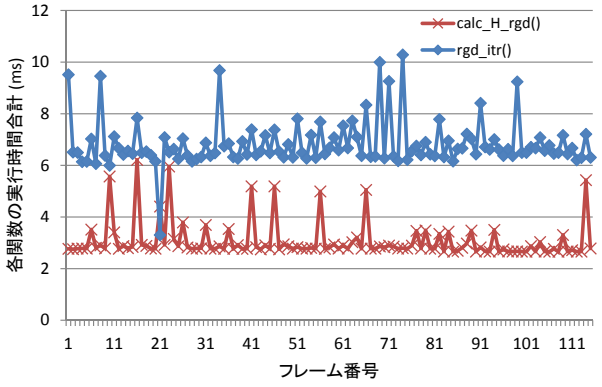

図 2: 1 フレームに対する各関数の総実行時間

並列化の検討を行なった結果，我々はテンプレート領域の画素数分の並列度がある 4 重ループ内の処理をハードウェア化することで安定化を高速に処理することとした．

また，FPGA 化する際の問題として，SW 側から HW 側に送信するデータ量が大きくなると，データ送信がオーバーヘッドとなり高速化できない問題がある．そのため，SW と HW 間のデータ通信量や通信方法についても調査・検討をする必要がある．

4 まとめ

本稿では，ソフトウェア動作している安定化処理の時間を測定し，時間がかかっている処理の分析を行なった．その結果，安定化フィルタの関数の一部に時間がかかっていたため，その部分に対して FPGA を用いた並列処理方法の検討について述べた．

今後は検討した内容の処理を FPGA 化することで，安定化処理における FPGA 導入効果を評価する予定である．

謝辞

本研究は，一部日本学術振興会科学研究費補助金（基盤研究 (C)24500055，同 (C)24500054，同 (C)25330055，若手研究 (B)25730026）の援助による．

参考文献

[1] http;//www.xilinx.com

[2] 松本 洋平: “逆合成法を用いた航海画像の安定化,” 日本航海学会, Vol.127, No.SIG 70, pp.205-214, 2012.

[3] SIMON BAKER, IAIN MATTHEWS: “Lucas-Kanade 20 Years On:A Unifying Framework,” International Journal of Computer Vision, Vol.56, No.3, pp.221-255, 2004.