ステレオカメラと GPU を用いた手の輪郭と爪からの高速な手指形状推定

旭川工業高等専門学校　　○平沢　拓己，以後　直樹，戸村　豊明

要　旨

市販の USB カメラを平行に配置したステレオカメラを用いて，掌側から見た手指画像内の手の輪郭と爪領域を用いて手の形状を推定する場合，爪検出処理と逆運動学計算処理がボトルネックとなる．そこで，本研究では，並列化手法の改善及び並列化区画の拡大によって爪検出処理の高速化を図り，高速で高精度な手の形状推定を行う手法を提案する．

1. はじめに

市販の USB カメラを平行に配置したステレオカメラは，カメラと赤外線カメラを組み合わせたセンサや 2 台の赤外線カメラを組み合わせたセンサといったものより，単純かつ小型かつ安価で高解像度なセンサとなり得る．

このようなステレオカメラを用いて，掌側から撮影した手指画像から手の形状推定を行う場合，指の関節角度が大きい状況等では，手の輪郭のみから正しい指先・指間位置を検出するのが困難となり，輪郭以外の情報が必要となる．

そこで, 以前の研究[1], [2]においてはステレオカメラの手指画像における指先・指間位置と爪領域を用いることにより，正しい指先・指間位置を検出し，高精度な手形状推定を行う手法を提案した．さらには，ボトルネックとなる爪検出処理を GPU の利用により高速化し, 高速で高精度な手の形状推定を行う手法を提案するために，一部区間の並列化を行った．

本研究では, NVIDIA が提供する GPU 向け統合開発環境である CUDA を利用して，以前の研究での並列化手法の改善及び並列化区画の拡大を図り，高速で高精度な手の形状推定を行うための，爪検出処理の並列化手法を提案する．

2. 既存の爪検出手法

本研究では，爪検出を行うために藤嶋・星野らの手法[3]を用いている．本手法は，図 1 のフローに示すような手順によって実現される．それぞれの手順を以下に記す．

(1) 爪色と肌色の分布領域の違いを利用するために，RGB 色空間において手領域の主成分分析を行い，第三主成分を爪尤度として，手領域内の各画素について爪尤度を計算する．
(2) 手領域の各画素について，爪尤度が条件を満たす画素を抽出し，二値画像を生成する．

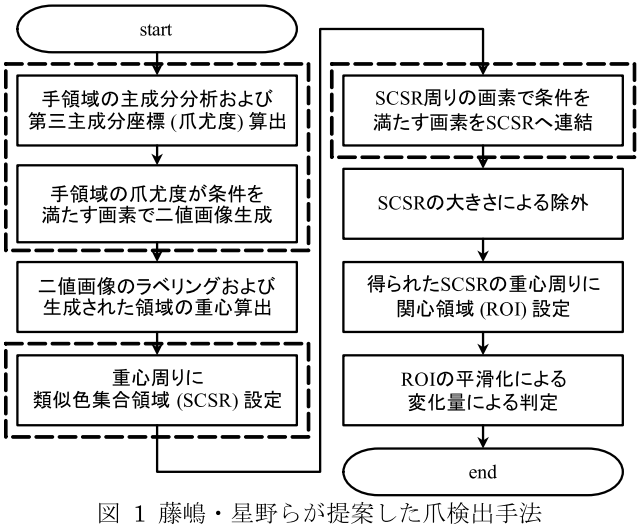

図 1 藤嶋・星野らが提案した爪検出手法

(3) 手領域内の各画素について，爪尤度が条件を満たす画素を抽出し，二値画像を生成する．
(4) 生成した二値画像にラベリングを行い，ラベリングによって生成された各領域の爪尤度の上位 100 画素で重心を算出する．
(5) 算出された重心周りに類似色集合領域（Similar Color Set Region: SCSR）を設定する．
(6) 設定した SCSR の外周を注目画素として，注目画素内で色の類似性の条件を満たす画素をすべて検出し，検出した画素を SCSR に連結する．すべての注目画素の探索終了後，条件を更新する．これを注目画素内に色の類似性の条件を満たす画素がなくなるまで繰り返す．
(7) 連結処理終了後に，SCSR の大きさが爪として仮定した矩形領域を超えていない領域を爪候補とする．
(8) 得られた爪候補とみられる SCSR の重心周りに関心領域（Region of Interest: ROI）を設定する．ROI を平滑化し，平滑化による変化量を求め，爪かどうか判定する．

上記の手順において，以前の研究では，手順 (4), (5) の連結処理を GPU に置換することを主目的とし, 処理の実装および実験を行った．本研究ではさらに手順 (1), (2) を GPU に置換することにする．

3. GPU による並列化のための手法

3.1. 以前の研究における手法

GPU による並列化を行うにあたって，2 章の手順 (4) および手順 (5) を以下のように置き換える．

(1) 重心G の膨張処理により，二値画像I_{SCSR}^{tmp} を生成する．
(2) 重心G が含まれる領域の画素を抽出することにより，二値画像I_{label} を生成する．
(3) I_{SCSR}^{tmp} とI_{label} との論理和により，SCSR の二値画像I_{SCSR} を生成する．
(4) I_{SCSR} の膨張処理により，二値画像I_{SCSR}^{dilate} を生成する．
(5) I_{SCSR}^{dilate} とI_{SCSR} との排他的論理和により，注目画素を含む二値画像I_{poi} を生成する．
(6) 色の類似性の条件を満たす画素を抽出することにより，二値画像I_{sif} を生成する．
(7) I_{poi} とI_{sif} との論理積により，連結画素を含む二値画像I_{SCSR}^{add} を生成する．
(8) I_{SCSR}^{add}に白色画素が存在する場合は手順 (9) へと進み，白色画素が存在しなければ処理を終了する．
(9) I_{SCSR}^{add} とI_{SCSR} との論理和により，二値画像I_{SCSR}^{new} を生成する．
(10) 計算量を削減するために, 重心を中心として ROI の対角線の長さを 1 辺に持つ正方形領域に含まれない画素を抽出することにより，除去用の二値画像I_{ROI}^{del}を生成する．
(11) I_{ROI}^{del}とI_{SCSR}^{new} の論理積により判定用二値画像を生成し，

判定用二値画像に白色画素が存在すれば処理を終了する．

(12) I_{SCSR}^{new}をI_{SCSR}と置き直し，色の類似性の条件を更新する．

(13) 手順 (4) へと戻り，処理を繰り返す．

高精度な爪検出を維持するために，各爪領域に対し本手法を実行する．

3.2. 本研究における手法

GPU による並列化を行うにあたって，本研究では 2 章の手順 (1) および手順 (2) を以下のように置き換える．

(1) 画像データから色情報P_{RGB}，位置情報P_{xy}を取得する．

(2) P_{RGB}内で黒以外の画素数を総画素数として計算する．

(3) P_{RGB}と総画素数から各要素の平均を算出する．

(4) P_{RGB}と各要素の平均から各要素の偏差を算出する．

(5) 各要素の偏差から分散共分散行列を作成する．

(6) 分散共分散行列の主成分分析により，第三主成分軸を算出する．

(7) P_{RGB}と第三主成分軸により，第三主成分座標（爪尤度）を計算する．

(8) 爪尤度が条件を満たす画素により，二値画像を生成する．

4. 実験結果

以前の研究では，左右のカメラから取得した入力画像から生成した動画を用いて，CPU 単体または CPU と GPU の組み合わせで手の形状推定を行った．このとき，Visual Studio の Debug モードと Release モードにおける 1 フレームあたりの平均処理時間を評価していた．本研究では，同様の評価手法をとり，CPU 単体または CPU と GPU の組み合わせで手の形状推定を行い，Visual Studio の Debug モードと Release モードにおける 1 フレームあたりの平均処理時間を評価する．それにより，並列化手法の改善及び並列化区画の拡大による速度の変化を考察する．

評価実験を行った環境を表 1 に示す．また以前の手法を用いて得られた結果を表 2，提案手法を用いて得られた結果を表 3 に示す．また，例として図 2 に示される入力画像から爪領域を検出する．以前の手法を用いて得られた結果を図 3 に示す．提案手法を用いて得られた結果を図 4 に示す．

5. 考察

表 2 と表 3 では，評価区間が 2 章の手順 (4) および手順 (5) のみである点と手順 (1)，手順 (2)，手順 (4)および手順 (5) のすべてが含まれるといった点から異なっているが，それを考慮しても GPU の全処理時間においては CPU の処理時間のような急激な増加が見られない．さらには，実際にカーネル関数内での処理時間に相当する実処理時間が CPU での処理に匹敵するまでの劇的な改善を見せた．しかし，いまだに Release モードでは CPU の処理時間を上回る速度を見せることができていない．問題は，3.1 節の処理が前段階の処理を必要とし，並列化が GPU にとって不可能ともいえる処理を含んでいたが，並列化が可能な各爪に対する処理の実装を見送っていた．そのため，GPU による高速化が十分に行われなかったという点と，GPU を効率的に使用できていないという点にある．

6. まとめ

本研究では，GPU を使用する処理区間を拡大することによる処理時間の変化を評価した．その結果，実処理時間の劇的な向上を行うことができた．しかし，3.1 節の手法において並列化が可能である爪単体での処理の CUDA での実装の見送りの影響を受けて，GPU が CPU に勝る結果を得ることができなかった．

今後の課題として，爪単体での処理を CUDA で実装し，データの転送処理などのボトルネックとなる部分を最適化することで，さらなる高速化を実現したい．

表 1 評価実験を行った環境

要素	詳細
USB カメラ	Logicool(R) HD Pro Webcam C920t
画像サイズ	640 x 480 (VGA)
CPU	Intel(R) Core(TM) i7-4770K @ 3.50GHz
GPU	NVIDIA GeForce GTX 760
CUDA	Version 6.5
カメラ間距離	150 mm

表 2 以前の手法の 1 フレームあたりの平均処理時間（単位: ms）

	CPU	GPU	
		全処理時間	実処理時間
Debug モード	609	36997	8113
Release モード	129	16778	7741

表 3 提案手法の 1 フレームあたりの平均処理時間（単位: ms）

	CPU	GPU	
		全処理時間	実処理時間
Debug モード	1202	37967	463
Release モード	173	10798	191

図 2 入力画像

図 3 以前の手法による結果

図 4 提案手法による結果

参考文献

[1] 平沢拓己，戸村豊明: 手指画像における輪郭と爪を用いた手の形状推定, 2014 年度精密工学会北海道支部学術講演会講演論文集, pp.47-48, 2014.

[2] 平沢拓己，戸村豊明: GPUを用いた輪郭と爪からの高速な手指形状推定, 2015年度精密工学会春季大会学術講演会, 2015.

[3] 藤嶋教彰，星野聖: 肌領域における色の連続性を利用した爪検出の高精度化，社団法人映像メディア学会技術報告, Vol.37 No.12, pp.5-8, 2013.