私の研究開発ツール ――第53回――

MATLABによる映像処理システム開発

てれびさろん my recommendations on research & development tools.

(正会員) 村 松 正 吾†

キーワード：映像処理，Computer Vision System Toolbox，Image Acquisition Toolbox，MATLAB Coder，組込みビジョン，FPGA開発

1．ま え が き

小学校6年生の長男がサッカーを始めて早6年．私もお父さん審判としてデビューしてから4年が経ちました．子どもの試合の審判とはいえ，試合中は緊張感をもって時々刻々と変化する状況に対応し，瞬時に正しい判定を下さなければなりません．試合が終わってから「ゆっくり考える」などあり得ない現場重視の世界です．

映像情報メディアにも多様かつ高度な要求が増えています．さまざまな環境の下，映像情報に関する瞬時の処理や判断機能が必須となっています．ところで，本稿において紹介するMathWorks社のMATLAB[1]ですが，映像処理分野に携わる研究開発者にとって，これまでは「ゆっくり考える」タイプのツールであったと思います．しかし最近では，映像関係の拡張オプションも充実し，高速化も図られ，並列処理や実時間処理も可能となっています．すなわち，瞬時の処理や判断が必要な現場においても充分に使えるツールへと進化しています．

今後数年の間に，組込みビジョン技術[2]が実世界に浸透し，急増すると予想されています．MATLABは組込みビジョンの開発環境としても拡張できます．アルゴリズム性能のほか，電力，速度，価格といった厳しい制約を満たすようにトレードオフを探すことや設計検証にも利用できます．MATLABについては，以前筆者が執筆した著書や本誌記事もご参照下さい[3][4]．本稿では，MATLABによる映像処理システム開発に焦点を絞り，便利な機能について簡単にご紹介します．

2．MATLABによる映像処理

MATLABは高級プログラミング言語であり，アルゴリズム開発，データ解析，可視化，数値計算のインタラクティブな環境を与えます．拡張オプションとしてImage Processing Toolboxを加えることで画像処理や解析にも対応します[3]．さらに，拡張オプションComputer Vision System ToolboxとImage Acquisition Toolboxを加えると，映像処理や解析にも便利に利用できます．これらオプションについて紹介します．

```
%% 映像データ読込の準備
reader = vision.VideoFileReader;
reader.Filename = '16SIF_Whale_Show.avi';

%% 映像データ表示の準備(コード生成対応)
viewer = vision.DeployableVideoPlayer;

%% 映像データの読込と表示
while ~isDone(reader)
   frame = step(reader); % フレームの読込
   step(viewer,frame)   % フレームの表示
end

%% リソースの解放
release(viewer)
release(reader)
```

図1　映像ファイルの読込と表示の例

2.1　Computer Vision System Toolbox

Computer Vision System Toolboxは，MATLAB R2011aより登場した拡張オプションで，映像処理やコンピュータビジョンの設計，シミュレーションに利用できます．**図1**に，映像データの読込と表示の例を示します．

図1のコードでは，フレーム1枚毎に読込と表示を行っています．長時間の映像でもメモリーを圧迫することなく読込と表示が可能です．**図2**に，実行の様子を示します．本オプションは映像入出力のほか，特徴点抽出や動き解析，ステレオビジョンなどの関数も備えています．

2.2　Image Acquisition Toolbox

Image Acquisition Toolboxは，カメラからの映像を直接MATLABに取込む機能を提供し，MATLAB上での実時間映像ストリーミング処理を可能とします．**図3**に，Webカメラからの映像の取込みと表示の例を示します．本オプションは，ループ処理内でのトリガーによる取込やバックグラウンドでの取込，複数デバイスへの対応など，現場でも使える機能も提供します．

† 新潟大学　自然科学系　工学部
"Video Processing System Development via MATLAB" by Shogo Muramatsu (Faculty of Engineering, Niigata University, Niigata)

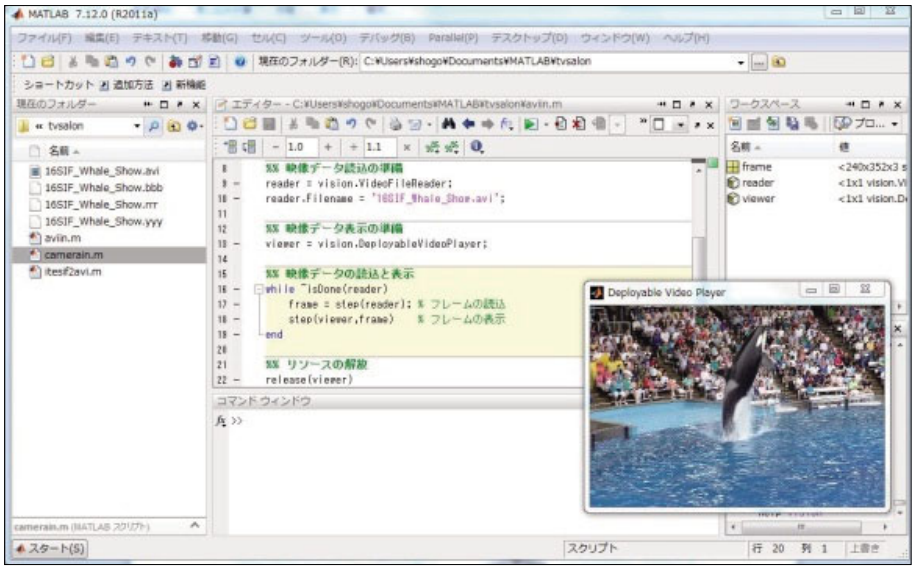

図2　図1のコードの実行の様子

```
%% カメラの準備
camera = videoinput('winvideo',1,'RGB24_320x240');
set(camera,'TriggerRepeat',Inf)

%% 映像データ表示の準備
viewer = vision.VideoPlayer;

%% 撮影と表示
nFrames = 600;
iFrame = 0;
start(camera);
while (iFrame<nFrames)
    if get(camera,'FramesAvailable')
        frame = getdata(camera,1); % フレームの取込
        step(viewer,frame) % フレームの表示
        flushdata(camera)
        iFrame=iFrame+1;
    end
end
stop(camera);

%% リソースの解放
release(viewer)
delete(camera)
```

図3　カメラ映像の取込と表示の例

3．映像処理システムの高速化

膨大な量のデータを瞬時に扱う映像処理には，処理速度にシビアな要求があります．本章では，MATLAB上での処理速度の解析に利用できるプロファイラーを紹介します．さらに，実行処理の高速化と大規模データ処理を可能とする拡張オプションMATLAB Coderと，Parallel Computing Toolboxについて紹介します．

3.1　プロファイラー

プロファイラーは，実行時間を解析するユーティリティで，コードのデバッグや最適化を支援します．プロファイラーの利用はとても簡単です．まず，実行時間を解析したいコードの前でコマンド"profile on"を実行します．続いて，解析したいコードを実行後，コマンド"profile off"を実行します．最後に，コマンド"profile viewer"を実行すると累積実行時間の解析結果が表示されます．

例として筆者が作成した2次元対称直交離散ウェーブレット変換（sowavedec2）と逆変換（sowaverec2）の実行時間の解析例を示しましょう．ただし，frameにはRGB形式の静止画像が予め保存されていると仮定します．

```
>> X=rgb2gray(frame);
>> profile on %解析開始
>> L=3;
>> [C, S]=sowavedec2(X, L);
>> Y=sowaverec2(C, S);
>> profile off %解析終了
>> profile viewer %解析結果表示
```

図4に，プロファイラーの表示例を示します．関数の呼び出し回数や実行時間が表示されます．時間を要するブロックを追跡し，重点的にコードを改変（リファクタリング）することで，処理速度を改善することができます．

3.2　MATLAB Coder

MATLABのコード上でも，配列操作や線形演算等を工夫することで，数倍の高速化が見込めます．しかしながら，更なる実行速度の高速化が必要な場合，C/C++コードの自動生成を行う拡張オプションMATLAB Coderを利用するとよいでしょう．コマンド

```
>> coder
```

を実行するとMATLAB Coderが立ち上がります．プロジェクト作成後，**図5**に示すような，データ型等の設定を行うウィンドウが画面右側に現れます．

MATLAB Coderは，MATLAB上で利用できる実行形式のMEX関数も生成できます．今回，3.1節で紹介したコードの一部をMEX関数化したところ，筆者の環境では，10

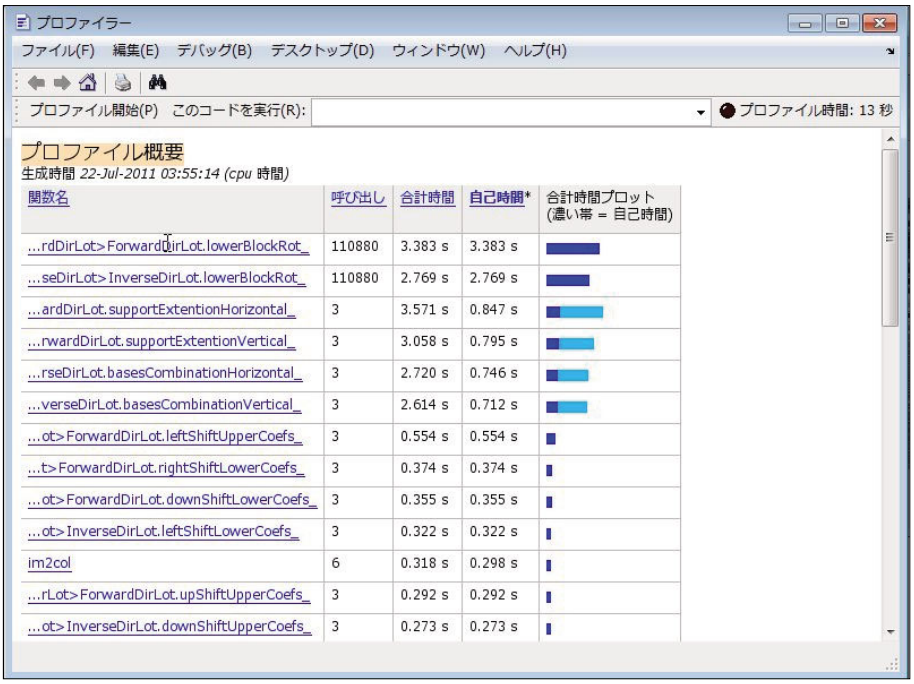

図4　プロファイラーの結果表示例

図5　MATLAB Coderの設定例

倍以上の速度向上が実現できました．ただし，筆者の経験から注意があります．MEX関数化は万能ではありません．MATLABが得意とする線形演算や，細かい単位でのMEX関数化は逆効果となることがあります．配列サイズが可変の場合，プロジェクトの設定において動的メモリー割当てを「行わない」とするのが良いでしょう．可能ならば配列サイズを固定化することをお勧めします．

3.3　Parallel Computing Toolbox

実行速度の高速化の別の手段として，並列・分散処理があります．拡張オプションParallel Computing Toolbox（PCT）の導入により，容易に並列処理を実現できます．R2011aでは，8コアまでのマルチコア，マルチCPU処理のほか，GPGPU処理も可能です．さらに，MATLAB Distributed Computing Serverとの併用により，8並列を超えるクラスタ計算にも対応します．MATLABメニューバー上にメニュー「Parallel」が存在すれば，同メニューから「Manage Configurations ...」を選択しましょう．すると，**図6**に示すConfiguration Managerが立ち上がります．右下の「Start Validation」ボタンを押すことで並列・分散処理の有効性を確認できます．

8フレームを8並列処理する例を以下に示します．ただし，8フレーム分のグレースケール画像が3次元配列Xに保存されていると仮定しています．

```
>> matlabpool local 8 %並列化
>> profile on
>> L = 3;
>> parfor iframe = 1:8
       [C, S] = sowavedec2 (X (:, :, iframe), L);
       Y (:, :, iframe) = sowaverec2 (C, S);
   end
>> profile off
>> matlabpool close %終了
```

上記のコードを実行した結果，並列化前と比べ並列化後は6倍以上の速度向上がみられました．また，MEX関数の利用により，さらに5倍以上の速度向上が実現できました．

4．MATLABによるFPGA開発

今後，映像処理やコンピュータビジョンの組込み機器への搭載が急速に進むと予想されています．MATLABも他の多くの組込み機器開発ツールと連携が強化されており，組込みビジョン装置の普及を推し進めるでしょう．

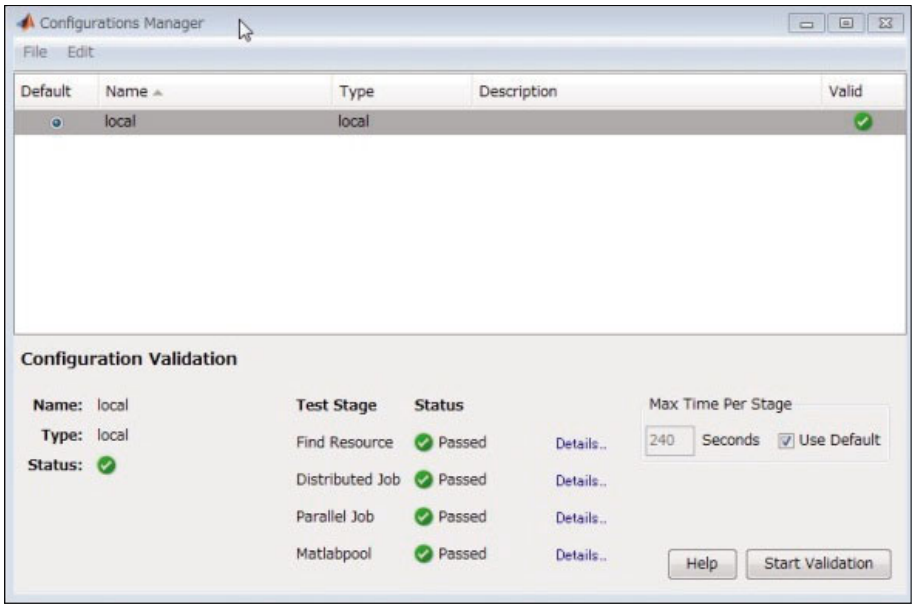

図6　並列・分散処理の有効性の確認

筆者らも，映像処理システムのFPGAへ実装を行っています．便利な拡張オプションを紹介しましょう．

4.1　Fixed-Point Toolbox

組込み機器をターゲットとしたアルゴリズム実装では，データと演算の固定小数点数化が望まれます．拡張オプションFixed-Point ToolboxはMATLAB上での固定小数点数表現の設定を可能とします．語長選択によるアルゴリズムの振る舞いへの影響を検討するのに便利です．

固定小数点数演算を利用するためには，コンストラクタfiに，数値，符号，語長，小数部のビット数を指定して固定小数点数オブジェクトを生成します．例えば，符号なし，語長8ビット，小数部6ビットで円周率π (pi)を扱うには，コマンド

```
>> piu8_6=fi(pi, false, 8, 6);
```

を実行します．

4.2　Simulink/Stateflow

FPGAの設計では語長選択の問題を無視して並列動作や状態遷移に興味を持つ場合が多くあります．このような場合，ブロック・ダイヤグラムに基づくモデリングツールのSimulink /Stateflowが便利です．システムアーキテクチャの浮動小数点数モデルを作成できます．

Simulink/Stateflowで作成した浮動小数点数モデルはSimulink Fixed Pointにより固定小数点数化できます．さらに，拡張オプションSimulink HDL Coderを利用すれば，FPGAに実装可能なVerilog/VHDLコードを生成できます．ただし，HDLコード化が可能なブロックは限られています．利用可能なブロックは予めコマンド

```
>> hdllib('html')
```

により確認できます．

最後にEDA Simulator Linkを紹介します．この拡張オプションは，映像処理システムのFPGA開発を飛躍的に効率化します．手動，自動によらず，作成したHDLの論理シミュレーションを，ブロックの一部として組み入れることが可能です．Computer Vision System Toolbox等を利用して作成したシステムモデルを，そのままテストベンチとして利用できます．映像データを利用したテスト検証も可能です．さらに，R2011aでは，実際のFPGA処理をブロックの一部としてリンクする，FPGA-in-the-loopシミュレーション機能も備わりました．

5. むすび

本稿では，MATLABによる映像処理システム開発について役立つ機能を紹介しました．現場での瞬時の処理にもMATLABを利用してみてはいかがでしょうか？ 紙面の都合上，ツールの入手方法，インストール方法，具体的実装コードの多くを割愛しましたが，補足事項については本学会誌Webサイトをご参照下さい．

(2011年7月25日受付)

〔文　献〕

1) MATLAB, http://www.mathworks.co.jp/products/matlab/
2) Embedded Vision Alliance, http://www.embedded-vision.com/what-is-embedded-vision
3) 村松正吾：“MATLABによる画像＆映像信号処理”，CQ出版社(2007)
4) 村松正吾：“私の研究開発ツール（第24回）MATLABでユニットテスティング”，映情学誌，63，6，pp.777-779（2009）

村松　正吾（むらまつ　しょうご）　1995年，東京都立大学大学院工学研究科修士課程修了．現在，新潟大学自然科学系（工学部）准教授．映像信号処理，多次元フィルタバンクの研究に従事．博士（工学）．正会員．