情報処理学会論文誌　コンピュータビジョンとイメージメディア　Vol. 2　No. 1　63–74 (Mar. 2009)

解像度非依存型動画像処理ライブラリ RaVioli の提案と実装

岡田　慎太郎[†1]　桜井　寛子[†1]
津邑　公暁[†1]　松尾　啓志[†1]

リアルタイム動画像処理システムへの要求から，これまでさまざまな専用チップが開発されてきた．しかし近年の汎用 PC の高性能化にともない，今後汎用 PC および汎用 OS を用いたリアルタイム動画像処理の需要が高まっていくと予想される．本研究では，使用可能な演算リソース量の変動によりリアルタイム性の保証が難しい汎用 OS 上でも，解像度（時間解像度および空間解像度）を自動的に変動させることによりこれを実現する動画像処理ライブラリ RaVioli を提案する．また，解像度の変動に対応したプログラミングは困難であることから，RaVioli は動画像処理における解像度をプログラマから完全に隠蔽するプログラミングパラダイムを提供することでこの問題を解消する．顔認識およびフレーム間差分検出プログラムを用いた検証の結果，RaVioli が十分な記述能力を持つこと，およびプログラムの変更なく解像度が変更可能であり，また汎用環境において，実時間で動画像が処理できるよう負荷が自動調整されることを確認した．

RaVioli: A Resolution-Independent Video/Image Processing Library

Shintaro Okada,[†1] Hiroko Sakurai,[†1]
Tomoaki Tsumura[†1] and Hiroshi Matsuo[†1]

Demands for real-time video processing are increasing and several ASICs are developed. On the other hand, the performance of general-purpose PC is growing up rapidly. Hence, real-time video processing on PCs will be demand in the near future. But it is difficult to guarantee realtime video processing on general-purpose systems. This paper discribes about a video processing library RaVioli which guarantees pseudo real-time video processing by adjusting resolutions of video automatically. Generally, writing the program which follows resolution change is very complex. RaVioli solves this problem by providing the new programing paradigm which hides the framerate and the image resolution from programmers. The result of the experiment with face-detection program shows that our library has good capability for writing programs and the program works correctly with different resolutions without any modification. Through the experiment with the program of detecting interframe difference, it is found that the resolutions are automatically modified appropriately.

1. はじめに

動画像処理能力の高いチップが多く開発され，空港や工場などの侵入者検知システムや，自動車走行中の前方車両もしくは障害物の認識による衝突回避システム[1] など，処理のリアルタイム性を重要視した動画像処理システムの開発がさかんに行われている．一方で計算機の高性能化により，顔認識アルゴリズムなどに代表される処理量の多い画像処理を汎用 PC 上で行うことが可能となりつつある．したがって，高度な認識アルゴリズムを実装したリアルタイム動画像処理を，比較的安価な汎用 PC および汎用 OS 上で行うことも今後多くなると予想される．

一方，Linux に代表される汎用 OS 上においてリアルタイム性を保証するためには，処理に必要な CPU のリソース量（以下 CPU リソース）を確保することが重要な課題となる．複数プロセスの並行実行によって使用可能な CPU リソースが時々刻々と変化したり，他プロセスと CPU リソースが共有されたりするため，1/30 もしくは 1/60 秒ごとに 1 フレーム分の CPU リソースの確保を保証するのは困難である．Linux をリアルタイム OS に拡張するプロジェクトも存在するが[2]，一般に毎フレームの演算量が変動する認識処理では，ワーストケースに基づく 1 フレーム単位の演算量の確保が必要となる．

そこで我々は，汎用システム上でも擬似的にリアルタイム性を保証した動画像処理を動作させることを可能とするライブラリ **RaVioli**（Resolution-Adaptable Video and Image Operating Library）を提案する．一般に，動画像処理プログラムにおいては出力フレームレートと 1 フレームの処理に割り当てることのできる演算量はトレードオフの関係にあり，使用可能な CPU リソースが限られる場合には**空間解像度**（1 フレームにおける画素数）および**時間解像度**（フレームレート）を低減させる必要がある．これに対し RaVioli は，ユーザが指定した優先度に応じて処理対象の空間解像度および時間解像度を自動的に変動させ，

†1 名古屋工業大学
Nagoya Institute of Technology

演算量を軽減させる機能を提供する．

また解像度を変動させる場合，画像の幅や高さの画素数や動画像におけるフレーム数，画素配列やフレーム配列にアクセスする際のイテレーション回数の変動に対応したプログラムを記述する必要があるが，これは困難である．プログラムが複雑化することでバグの温床となる可能性があり，開発プロセスにおけるコストの増大にもつながる．そこで RaVioli では，動画像処理における解像度の変動ひいては解像度そのものを，プログラマから完全に隠蔽するプログラミング環境を提供する．この環境では，動画像処理における繰返し処理単位を任意に設定し，その単位に対する繰返し方法を記述するのみでよいため，プログラマは構成画素数やフレームレート，繰返し文など解像度を意識したプログラムの記述を省略でき，また RaVioli 側では解像度を自由に変動させることができる．

解像度を隠蔽したプログラミング環境は，プログラマに意識させずに演算量を変動させることができるだけでなく，人間が本来持つ，画像に対する処理のイメージをそのまま画像に反映できるといった利点がある．人間の脳内における認識過程には，視覚情報である画像に対して「これは画素の集合である」という意識は存在しない．しかし量子的に情報を扱う計算機では，画像を点の集合として扱わざるをえなかった．RaVioli は，動画像における解像度を隠蔽することで人間と計算機の動画像の扱い方の違いを吸収し，人間が動画像に対して持つ直感的な処理手順を，ほぼそのままの形でプログラムに記述できる新しいプログラミングパラダイムを提供するライブラリであるといえる．

本論文では，2 章で RaVioli の基本的な概念について述べ，3 章で具体的な実装方針と RaVioli の仕様について述べる．次に 4 章では画像および動画に対する処理を通じた評価結果を示し，5 章で関連研究との比較から RaVioli の有用性を示す．最後に 6 章で本論文全体をまとめる．

2. 設計思想

2.1 演算量の自動調整

一般に汎用 PC および汎用 OS 上では，1 フレームあたりの演算量の変動や他プロセスの動作による使用可能な CPU リソースの変動から，動画像をリアルタイムに処理することは困難である．この状況を擬似的に解決するために，動画像のフレームあたりの演算量に応じて，動的に画素数を低減させることが考えられる．逆に演算に使用できる CPU リソースが十分ある場合には，十分高い画素数を維持することができる．このようにフレームレートを維持しつつ処理できる最大の画素数となるように自動調整することで，使用可能な CPU リソースが変動する環境においても，つねに実時間に近い形で処理結果を出力することが可能になる．

一方で顔認識のような，厳密なリアルタイム処理よりも画像の精度が重要であるような処理も存在する．そのような場合には，画素数を高い精度で維持しつつフレームレートを自動的に低減させることで演算量を調整する．

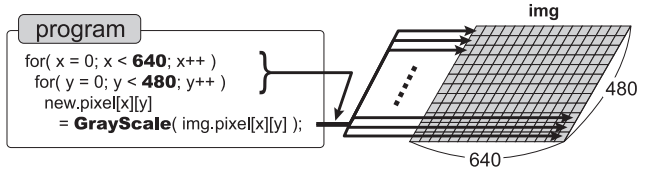
しかしながら，一般にはこの演算量の自動調整機能を実現するために，変動する解像度を意識したプログラミングが必要となる．この問題と問題に対する提案を次節に示す．

2.2 動画像処理の抽象化

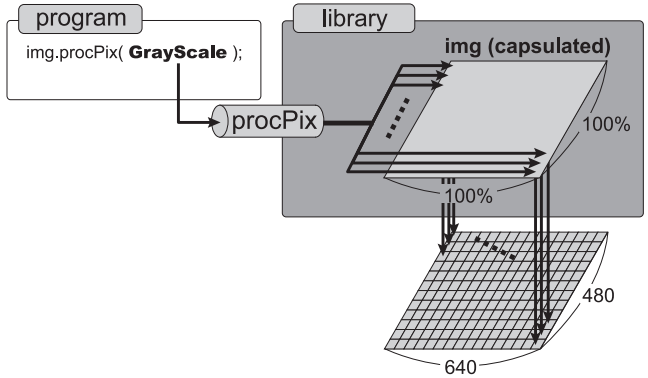
解像度を自動的に調節するという方針に基づいて動画像処理プログラムを実装する際，プログラマは解像度の変動を考慮してプログラムを記述する必要がある．これはプログラムの可読性が低くなったり，デバッグの際にバグ特定が困難になったりするなどの問題につながる．

この問題に対し RaVioli は，プログラマから解像度の概念を隠蔽するプログラミングパラダイムを提供する．動画像処理プログラムから空間解像度と時間解像度を示す変数を排除し，RaVioli 側ですべて管理することで，プログラマは解像度を意識しなくても動画像処理を記述できる．しかしその代償として動画像処理の自由度が低下する可能性がある．RaVioli はこの問題を解決するために，動画像に対するさまざまなインタフェースを提供する．

一般に画像処理は，小さな単位に対する処理を画像全体または任意の範囲に繰り返し適用するものが多い．たとえばカラーからモノクロへの変換や色の反転などの処理では処理単位は画素であり，ぼかしやエッジ強調などの近傍処理では，処理単位は画素およびその近傍である．また，テンプレートマッチングなどの処理では処理単位は小さなウィンドウである．そしてこれらの処理は，**図 1** (a) のように通常ループイテレーションで記述され，画像全体もしくは特定の範囲に対して繰り返し適用する形で行われる．ここで空間解像度の変動の影響を受けるのは，640，480 といったイテレーション回数や，イテレーション変数のインクリメント幅である．RaVioli は，処理単位に対する処理のみをプログラマに記述させ，その処理をライブラリ側で自動的に全画素や特定範囲の画素に適用することで，空間解像度を隠蔽する．(図 1 (b)) また空間解像度と同じく，時間解像度も隠蔽する．プログラマは当該フレームまたは当該および隣接フレームに対する処理のみを記述し，RaVioli が必要なフレームに処理を適用する．



(a) Traditional image processing.



(b) Image processing with RaVioli.

図 1　画像処理の処理イメージ
Fig. 1　Image processing flow.

3. ライブラリ仕様

3.1 基 本 設 計

3.1.1 パラメータに基づく処理量調整

汎用 PC が提供可能な CPU リソースより 1 フレームの処理に必要な演算量が多く，処理結果をリアルタイムに近い時間間隔で出力できない場合に，RaVioli は処理解像度を制御する**解像度ストライド**を変更させることで演算量を調整する．この解像度ストライドは処理画素数と処理フレームレートのそれぞれを制御する 2 つの値であり，これらの値に応じて，動的に空間解像度および時間解像度を変更する．**図 2** は，処理画素数の解像度ストライド S_I と処理フレームレートの解像度ストライド S_T がそれぞれ 4 と 3 の場合の処理対象となる部分を示した図である．薄いグレーのフレームが処理対象となるフレームであり，濃いグレーの画素が処理対象となる画素である．まず S_T が 3 であるため，キャプチャされたフレームを 2 枚ずつスキップしたものを処理対象フレームとする．さらに S_I が 4 であるため，処理対象フレーム内の画素を 3 つずつスキップしてアクセスする．つまり実際に処理されるのは濃いグレーの画素になる．このように解像度ストライドを変更し，通常の 1/3 のフレーム数と 1/16 の画素数のみに対し処理を適用することにより，演算量を低減させる．

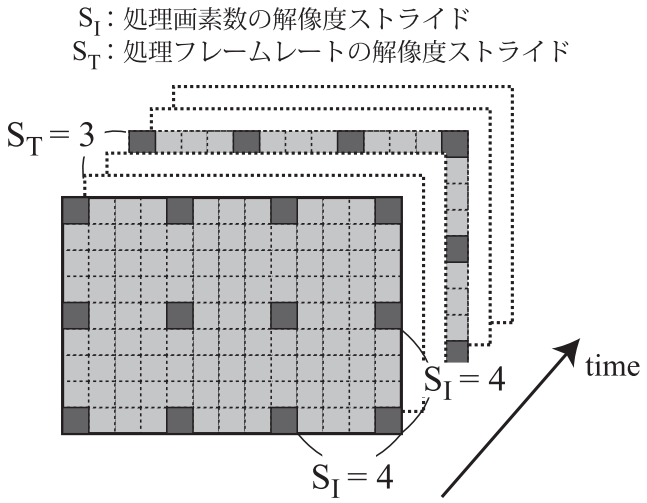


図 2　解像度ストライドに基づいたアクセス位置の指定手法
Fig. 2　Pixel-access method based on resolution strides.

またプログラマが処理画素数と処理フレームレートのどちらをどれだけ維持させるかを指定するために，維持させる割合を指示するための**優先度パラメータ**を導入する．この優先度パラメータには，処理画素数の優先度を表すパラメータ P_I と処理フレームレートの優先度を表すパラメータ P_T があり，パラメータの値が大きいほど優先する割合が高くなる．またどちらかのパラメータ値を 0 とした場合には，0 と設定された要素がいっさい優先されなくなる．たとえば (P_I, P_T) を (1, 0) とした場合は，S_I をつねに 1 とする代わりに S_T を増加させて処理フレームレートを低減させる．逆に (0, 1) とすれば，S_T を 1 で維持する代わりに S_I を増加させて処理画素数を低減させる．また (P_I, P_T) を (3, 7) と設定した場合は，S_I と S_T が 7 : 3 に近い値で維持されるようにする．こうすることで優先度パラメータに基づいた割合で処理画素数および処理フレームレートを削減する．

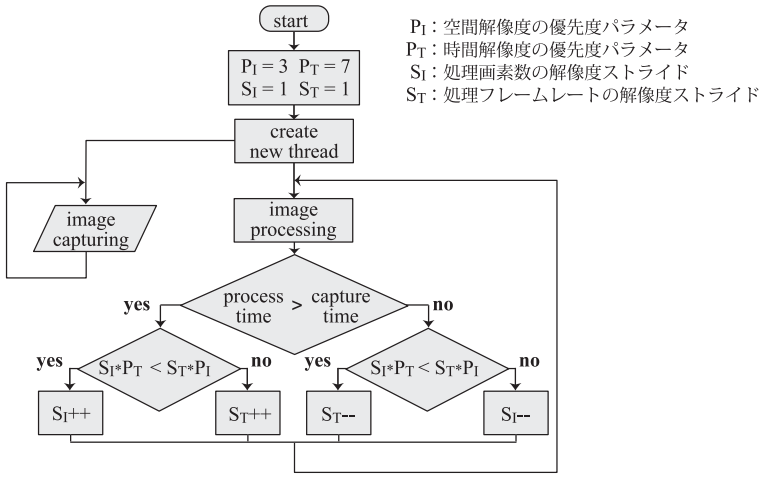


図 3　処理量を制御するパラメータのフローチャート
Fig. 3　Flowchart of image processing with parameter control.

以上に述べた優先度パラメータに基づく演算量調整の流れを**図 3** に示す．図はそれぞれの優先度パラメータ (P_I, P_T) を (3, 7) とした場合とし，処理開始後にそれぞれのパラメータの値を設定する．キャプチャしたフレーム 1 枚を処理した後にフレームあたりの処理時間とキャプチャ間隔を比較し，処理時間が大きい場合は解像度ストライドを増加させる．この際，処理画素数の解像度ストライド S_I と処理フレームレートの解像度ストライド S_T のどちらを増加させるかは，優先度パラメータ (P_I, P_T) をもとに決定する．また一方でフレームあたりの処理時間よりもキャプチャ間隔が大きい場合は，(P_I, P_T) をもとに解像度ストライドを減少させる．解像度ストライドの決定後は次のフレームの処理に移るが，その際図 2 で示すように処理フレームレートの解像度ストライドに応じた処理フレームを選択する．そして処理画素数の解像度ストライドに応じた処理画素に対して 1 フレーム分の処理を適用する．

3.1.2　解像度の隠蔽

RaVioli では，画素数を隠蔽するためにフレームの画素配列をカプセル化する．カプセル化されたフレームインスタンスは，処理を適用するためのインタフェースとして，処理対象の構成画素の指定された範囲に処理を適用するさまざまな**高階メソッド**を持つ．高階メソッドは，画像処理における処理単位に対して行うべき処理を定義した関数を引数にとり，その関数を構成画素全体または指定された範囲内に施すメタ関数である．

また，動画像におけるフレームの配列も画素配列と同様にカプセル化する．カプセル化された動画像インスタンスは高階メソッドを 2 種類持つ．引数にとる関数の処理単位が 1 枚のフレームの場合は，動画像の各フレームに処理を適用し，2 枚のフレームの場合は，配列上で隣り合うフレームに処理を適用する．

3.2　主なクラスと高階メソッド

前節で述べた機能を実現するにあたり，RV_Pixel，RV_Image および RV_Streaming の 3 つのクラスを持つライブラリ RaVioli を C++で実装した．以下それぞれのクラスと，RV_Image クラスおよび RV_Streaming クラスが持つ高階メソッドついて示す．

3.2.1　RV_Pixel クラス

画像を構成する画素の情報を保持するクラスであり，RGB それぞれの値を，8 ビットのメンバ変数として持つ．また，今後ビット深度の異なる色情報も扱えるように，濃度の範囲を 0 から 1000 の整数値で表した値（以下，**抽象濃度**）を用いて操作するメソッドを持つ．

3.2.2　RV_Image クラス

画像の実体を表すクラスであり，メンバ変数として構成画素数分の RV_Pixel インスタンスを配列の形で保持する．また，空間解像度における解像度ストライド変数 stride を持ち，画素アクセスの際の現スキップ間隔を保持する．

ここで，処理を施す範囲を画像全体ではなく画像の一部とする場合には，処理範囲の始点と終点を指定する必要がある．同様に，拡大縮小処理やハフ変換なども座標を使った処理が必要となる．しかし RaVioli では空間解像度を隠蔽しており，画像内の絶対的な座標値は使用できないため，代替手段を用意している．空間解像度を隠蔽した環境において画像内の位置を指定するために，画像の幅と高さを 1 として正規化した座標（以下，**抽象座標**）を用いる．たとえば画像の中心は抽象座標を用いて (0.5, 0.5) と表すことで，解像度に影響されずつねに中心を指示することができる．

表 1 に実装した高階メソッドの種類，返り値および使用例を，そして以下にその詳細を示す．なお，高階メソッドの引数として定義されている抽象座標 C_S および C_E は，画像内における処理範囲の対角頂点を指定するものであり，省略した場合は画像全体が処理対象の範囲となる．

procPix(RV_Pixel **Func(P)*, C_S, C_E)

すべての対象画素に対して，指定された関数 *Func* を適用し，処理結果の画像を返す．プログラマは，RV_Pixel クラスである画素 P を引数とした関数を定義する．閾値判定によ

表 1　高階メソッドの種類
Table 1　List of high-order methods.

メソッド名	処理単位	返り値	使用例
procPix	1 画素	同位置の画素	二値化
procNbr	1 画素，近傍	同位置の画素	畳み込み積分
procCoord	1 画素，座標	なし	ハフ変換
transCoord	1 座標	変換座標	拡大縮小
procBox	ウィンドウとその左上座標	なし	テンプレートマッチング
compImg	1 画素，別画像の画素	同位置の画素	差分検出

る二値化などに利用する．

procNbr(RV_Pixel *Func(P, *P_{Nbr}, Num), C_S, C_E)

すべての対象画素に対して，1 画素とその近傍を入力として *Func* を適用し，処理結果の画像を返す．プログラマは，画素 P，その近傍画素の集合である RV_Pixel 配列へのポインタ P_{Nbr}，および近傍画素数 Num を引数とした関数を定義する．畳み込み演算などに利用する．

procCoord(void *Func(P, C), C_S, C_E)

すべての対象画素に対して，1 画素およびその抽象座標を入力として *Func* を適用する．プログラマは，画素 P とその抽象座標 C を引数とした関数を定義する．また *Func* は，必要であれば画素および抽象座標から得た中間データを大域変数に格納する．ハフ変換などに利用する．

transCoord(void *Func(C_{bfr}, *C_{aft}))

画像全体を処理範囲とした場合の全対象画素に対して，定められた対応規則 *Func* に従って画素の座標を変換する．プログラマは，変換前，変換後の 2 つの抽象座標 C_{bfr} および C_{aft} を引数とした関数を定義する．画像の回転，拡大縮小などに利用する．

procBox(void *Func(Img_B, C_{Bs}, C_{Be}), W, H)

全対象画素を始点とする，抽象座標を利用して指定された幅 W，高さ H を持つウィンドウに対し，そこに含まれる画素に対し処理を行う．W および H は全体画像に対する割合（0〜1）で指定する．プログラマは，マッチパターンなどに用いる RV_Image インスタンス Img_B，およびウィンドウの始点・終点抽象座標 C_{Bs}，C_{Be} を引数とした関数を定義する．また procCoord() と同様に，*Func* は必要であれば画素から得た中間データを大域変数に格納する．テンプレートマッチングなどに利用する．

compImg(RV_Pixel *Func(P_1, P_2), Img)

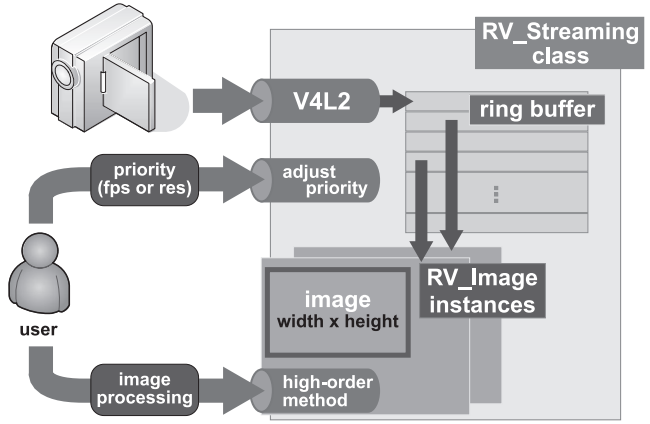


図 4　RV_Streaming クラスの構造
Fig. 4　Structure of RV_Streaming class.

画像全体を処理範囲とした場合の全対象画素と，引数で受け取った RV_Image インスタンス *Img* の対応する位置の画素との比較を行う．プログラマは，比較元，比較対象の 2 つの画素 P_1，P_2 を引数とした関数を定義する．2 枚の画像間の差分検出などに利用する．

3.2.3　RV_Streaming クラス

動画像を処理するクラスであり，カメラから動画像をキャプチャするメソッドと，数枚のフレームを配列で格納するリングバッファおよびそのメソッドなどを持つ．概略を**図 4** に示す．カメラから動画像をキャプチャする部分は，Video4Linux2 (V4L2) [⋆1] ドライバを用いて実装し，取り込んだフレームをリングバッファに一時的に保存する．フレーム処理時にはリングバッファにあるフレームの情報および処理画素数の解像度ストライドを RV_Image クラスのインスタンスに格納し，動画像用の高階メソッドを使ってプログラマが記述した処理を動画像に適用する．1 フレームの処理が終了した後，優先度ストライドの値を元に各解像度ストライドを変更する．そしてリングバッファから処理フレームレートの解像度ストライドに応じた時刻のフレームを選択し，新たに生成された RV_Image インスタンスに処理画素数の解像度ストライドを設定する．

ここで，例として当該フレームと 1 つ前のフレームをそれぞれグレースケールに変換し，

⋆1 http://linux.bytesex.org/v4l2/

それぞれのフレーム間の差分をとる処理を考える．1 つ前のフレームをグレースケールに変換した時点での処理画素数の解像度ストライドと，当該フレームを変換した時点での処理画素数の解像度ストライドは，それぞれが異なる場合が存在する．当該フレームの処理画素数が 1 つ前のフレームの処理画素数より多い場合は，位置によっては対応する差分処理の処理画素が 1 つ前のフレームに存在しない．このような処理に対しては，本来差分処理の対象画素が存在する位置の近傍画素を比較対象として代用することで対応する．具体的には，当該フレームと 1 つ前のフレームのそれぞれが持っている処理画素数の解像度ストライドを比較し，当該フレームにおける処理画素数の解像度ストライドが大きい場合は，前フレームで処理がされていない画素へのポインタを，その近傍で処理がされている画素に読み替える．このように実装することで，画素数の異なるフレームどうしの比較・差分処理をすることができる．

代表的な高階メソッドを以下に示す．

proc1Frm(*Func(*Img_{Curr}*))

すべての処理対象フレームに対し，指定された関数 *Func* を適用する．*Func* は 1 つの RV_Image インスタンス Img_{Curr} を引数にとる関数である．背景差分などに利用する．

procAdjFrm(*Func(*Img_{Curr}*, *Img_{Prev}*))

すべての処理対象フレームとその隣接フレームに対して指定された関数 *Func* を適用する．*Func* は 2 つの RV_Image インスタンス Img_{Curr}，Img_{Prev} を引数にとる関数である．RaVioli を使用した環境では，状況に応じて処理フレームをスキップすることで自動的に処理フレームレートが変動する．この際隣接フレームがどのキャプチャフレームにあたるかは RaVioli によって自動的に判断され，当該フレームと共に *Func* に渡される．動画のフレーム間差分などに利用する．

4. 評　　価

4.1 記述能力と動作

動画像に対する処理の評価として顔検出プログラムを用い，RaVioli の記述能力や解像度変更への対応の評価を行った．

4.1.1 評価プログラム

RaVioli の動画像処理能力を評価するにあたり，サンプルプログラムとしてハフ変換を使った複数人の顔検出プログラム[3] を実装した．

まず背景画像と各時刻での入力画像における色の差分値を求め，差分値が閾値以上なら黒，閾値以下なら白とした 2 値画像を得る．次に，画像中で人が存在している可能性の高い領域を検出するために，縦方向のスキャンラインに対し，それぞれに含まれる黒の画素数をカウントし，画像の幅を区間とした黒の画素数の分布を算出する．黒の画素数の和が閾値以上あるラインが一定数以上続けば，その範囲は人が存在する可能性があると判断し，処理対象範囲とする．また同じ 2 値画像から，エッジの部分だけを黒とし他を白としたエッジ抽出画像を得る．エッジ抽出画像中の処理対象として指定されたそれぞれの範囲に，ハフ変換を用いた円検出を適用する．検出された円のうち，円を構成する画素数の多い上位 5 つを顔の候補とし，5 つの円のうち領域内で一番上の位置にある円を顔とする．以上の顔検出プログラムを RaVioli を用いて記述し，動作させることにより評価を行った．

4.1.2 記述能力の評価

RaVioli を使用した場合と使用しない場合において，プログラマが記述するプログラムの変化を**図 5** に示す．なお顔検出処理の中の背景差分の部分のみを抽出して掲載した．

具体的には，背景フレーム `bgFlm` と当該フレーム `newFlm` との r，g，b それぞれの差分の絶対値 `difr`，`difg`，`difb` を算出し，それらの値のいずれかが閾値以上である場合はその画素を黒に，そうでなければ白にするという処理を一定フレーム数ごとに行うプログラムである．なお，プログラム中に出てくる閾値は 77（抽象濃度では 300 に相当）とした．

RaVioli を使用しない場合のプログラムにおいて，一番外側の `for` ループは繰返しフレームをキャプチャするための記述である．キャプチャを行う関数 `readFlame()` を使って読み込んだフレーム `newFlm` と，事前に取得したフレーム `bgFlm` を用いて，2 重の `for` ループで背景差分を行う．ここで，フレームレートや画素数を動的に変動させる際には，`ST` や `SI` を動的に変動させるプログラムを追加する必要がある．

これに対し RaVioli を使用する場合，RV_Streaming クラスに定義されている高階メソッド `proc1Frm()`，および RV_Image クラスに定義されている高階メソッド `compImg()` を用いる．`proc1Frm()` には 1 フレームを処理単位とした関数 `interframe()` を渡すだけでよく，また `compImg()` には背景フレームと当該フレームにおける同位置の 2 つの画素 `newP` と `bgP` を処理単位とした関数 `compare()` を渡すだけでよい．このように，プログラマがフレームレートや構成画素数の値を使用することなく，RaVioli を使用しない場合とまったく同じ処理を記述することができる．また，本来であれば画素数やキャプチャフレームの位置などを強く意識して記述する必要のある部分であるループなどの繁雑な箇所が省かれることで，バグの混入可能性を低下させるとともに，RaVioli 内で自由に解像度を変更することができ，演算量の調整が可能になる．

RaVioli を用いない記述

```
int main(){
  IMG bgFlm; //背景画像（事前に取得）
  for(;;CapFlm += ST){
    newFlm = readFrame(CapFlm);
    for(y = 0; y < img.H-SI; y += SI){
      for(x = 0; x < img.W-SI; x += SI){
        int difr, difg, difb;
        difr = abs(newFlm[x][y].R - bgFlm[x][y].R);
        difg = abs(newFlm[x][y].G - bgFlm[x][y].G);
        difb = abs(newFlm[x][y].B - bgFlm[x][y].B);
        if( difr > 77 || difg > 77 || difb > 77)
          newFlm[x][y].R=
          newFlm[x][y].G=
          newFlm[x][y].B=0;
        else newFlm[x][y].R=
             newFlm[x][y].G=
             newFlm[x][y].B=255;
}}}}
```

RaVioli を用いた記述

```
RV_Pixel compare(RV_Pixel newP, RV_Pixel bgP){
  int r, g, b, br, bb, bg, difr, difg, difb;
  newP->getRGB(r, g, b);
  bgP->getRGB(br, bg, bb);
  difr=abs(r-br); difg=abs(g-bg); difb=abs(b-bb);
  if( difr > 300 || difg > 300 || difb > 300)
    newP->RGBabs(0, 0, 0);
  else newP->RGBabs(1000, 1000, 1000);
  return(newP);
}
RV_Image bgFlm; //背景画像（事前に取得）
void interframe(RV_Image newFlm){
  RV_Image outFlm;
{  outFlm=newFlm->compImg(compare, &bgFlm);  }
}
int main(){
  RV_Streaming video;
  video.Proc1Frm(interframe);
}
```

図 5　RaVioli 不使用/使用の場合のプログラム（部分）
Fig. 5　Difference of codes between ‘without RaVioli’ and ‘with RaVioli.’

前項で示した顔検出プログラムを RaVioli を用いて記述し，処理画素数の解像度ストライド S_I を変えて動画像処理を行った結果を**図 6** に示す．ここでは，$S_I = 1$，$S_I = 2$（処理画素数 1/4），$S_I = 3$（処理画素数 1/9）とした．この結果から，プログラムをいっさい変更することなく解像度の変更に対応できており，期待した結果が得られることを確認した．

また S_I の値を上げていくと，S_I=7 の時点で画素数の減少により情報量が少なくなりすぎたために，抽出された円と顔の部分が一致せず認識に失敗した．この失敗例を**図 7** に示す．顔の検出のような画像から特徴を検出する処理は，一定の処理画素数を保持する必要があるため，処理画素数を優先させるように優先度パラメータを指定する必要がある．

4.2　動画像処理のリアルタイム性の評価

処理画素数と処理フレームレートの調整による疑似リアルタイム処理の評価を，フレーム間差分を用いて行った．用いたシミュレーション環境を**表 2** に示す．

サンプルとして用いたフレーム間差分プログラムは，時間軸上で隣り合う 2 枚のフレームのすべての画素において，RGB それぞれの差分の絶対値をとり，抽象濃度 100（RaVioli 不使用の場合は 25）より大きい場合はその画素を白，それ以外の画素を黒とすることで移動体を検出する処理である．

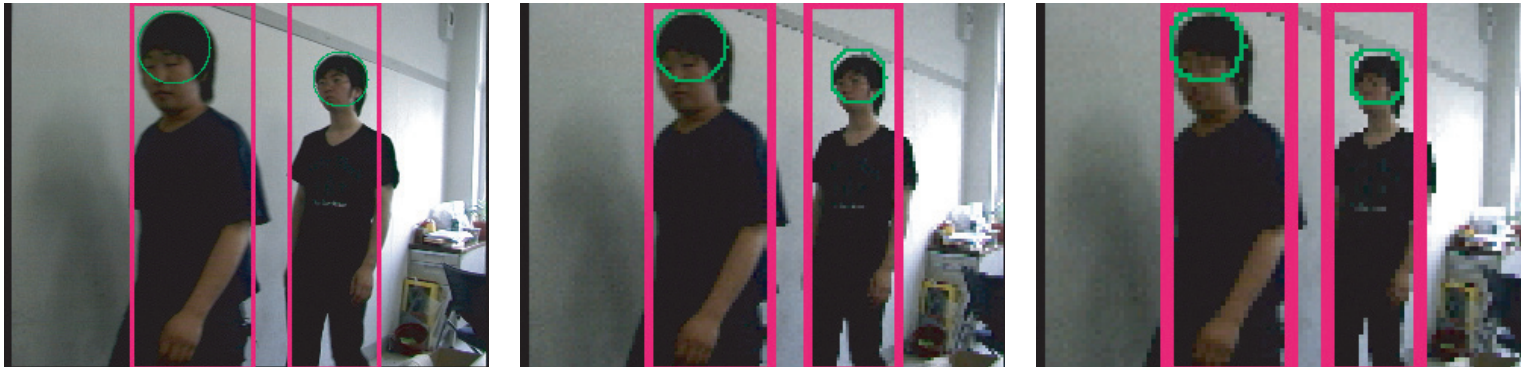
カメラから 30 fps で転送される解像度 320×240 のフレームをキャプチャし，フレーム間差分プログラムを出力フレームレート優先と出力ピクセル数優先でそれぞれ適用させることで，評価を行った．それぞれを優先した場合の処理画像，出力結果，および出力ピクセル数と出力フレームレートの変化を**図 8** に示す．

（a）は出力フレームレートと出力画素数の優先度パラメータ (P_I, P_T) を (1, 0) とした場合，(b) は (0, 1) とした場合，(c) は (7, 3) とした場合の，処理開始から 6 秒後までの時間変化を表したグラフである．なお処理開始の 2 秒後から 2 秒間，別プロセスとして無限ループを行うプロセスを動作させ，使用可能な CPU リソースを減少させた．(a) では出力フレームレートはつねに最大値が維持されている一方で，出力画素数は負荷を与えた時点で自動的に低下していき，約 4.5 秒後には元の画素数に戻る様子が読み取れる．(b) においても同様に，出力画素数は最大値を維持しつつ，出力フレームレートは負荷を与えた時点で自動的に低減し，その後に元のフレームレートの上昇が読み取れる．また（c）も，負荷が与えられた時間において優先度に沿った解像度の削減が行われているといえる．

以上の結果から，RaVioli がプログラマの指定する優先度に基づき正しく自動負荷調整を行えること，および処理画素数や処理フレームレートが変動した場合でもプログラムが正しく動作することを確認した．

4.3　RaVioli の適用範囲

プログラマは 3.2 節で述べたクラス及び高階メソッドを使用することで，処理画素数や



(a) $S_I = 1$　(b) $S_I = 2$　(c) $S_I = 3$

図 6　各解像度における出力画像
Fig. 6　Output images with various resolutions.

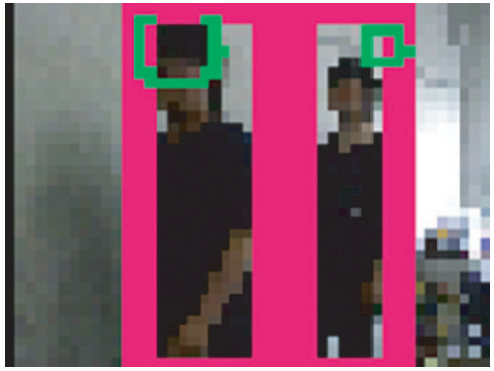


図 7　処理の失敗例（S_I=7）
Fig. 7　Example of failed case.

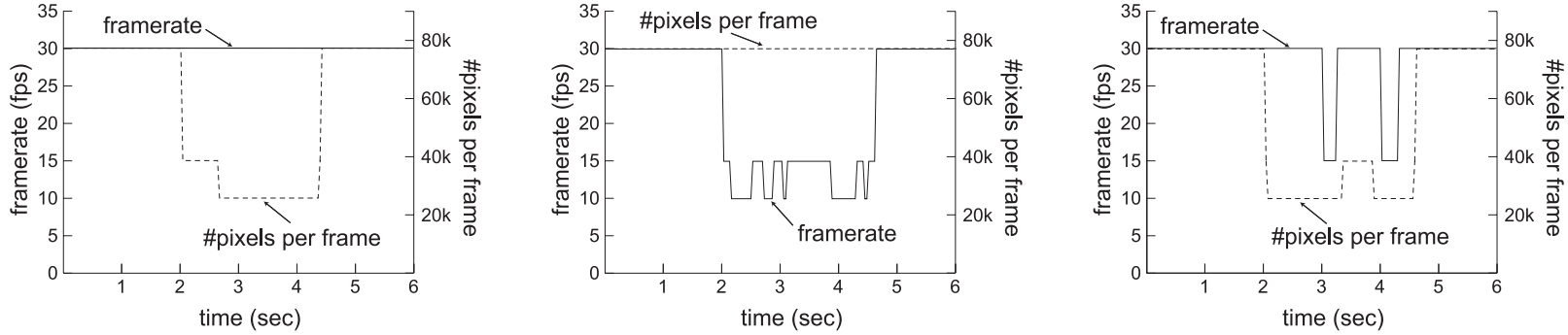
処理フレームレートを意識することなく，さまざまな動画像処理プログラムを記述することができることを，サンプルプログラムを作成することにより確認した．画像処理においては，閾値判定による二値化やグレースケール化といった画素ごとに独立した処理，メディアンフィルタによるノイズ除去やゾーベルフィルタによる輪郭抽出などの近傍処理，テンプレートマッチングによる顔検出などの処理が記述可能であった．また画像以外の配列を利用した画像処理として，ハフ変換による直線抽出などの処理も記述可能であった．さらに画素の色情報を使った処理に限らず座標の変換として，回転や拡大縮小や台形への変換など画像の形を自由に変更できることを確認した．また本論文中には示していないが，画素の処理順を決定する高階メソッドを利用することで，2 値化画像における輪郭追跡や領域のような逐次追跡型の処理が記述できることも確認している．

表 2　動画像処理シミュレーション環境
Table 2　Simulation environment for video processing.

CPU	AMD Opteron Dual-Core (2 GHz)
Memory	2 GB
Camera	SONY DCR-TRV900
Capture board	I·O DATA GV-VCP2M
Format	NTSC
Interface	S-video (S1)

動画像処理の例として，1 フレームに対する処理を毎フレームに適用する処理や，背景差分やフレーム間差分といった 2 枚の画像を比較するような処理が記述可能であった．

一方で，二次元画像からの三次元モデルの再構成や，投影による二次元画像への変換についてはいまだ検討段階である．今後ワイヤフレームモデルやサーフェスモデルのような三次元モデル[4] を実現するインタフェースの実装，また二次元データとの変換が可能となるメソッドの生成により実現可能であると考えている．



(a) $(P_I, P_T) = (1, 0)$　(b) $(P_I, P_T) = (0, 1)$　(c) $(P_I, P_T) = (7, 3)$

図 8　優先度を設定した場合の解像度の時間変化
Fig. 8　Resolution transitions with several priorities.

また RaVioli では，動画像処理において単純に画素およびフレームをスキップして処理を適用する方法をとるため，高周波成分情報の損失など，出力結果の精度低下という悪影響を及ぼすおそれがある．たとえばゾーベルフィルタを用いたエッジ検出において，スキップされた画素の輝度値が，隣り合った処理対象画素の輝度値と著しく異なっていても，エッジとして検出することができない．またオプティカルフローを用いた移動物体のトラッキングなどにおいて，移動物体の進行方向が頻繁に変わったり動作が速くなったりすると対応できなくなる．もちろん，画素をスキップする際にバイリニア法やハイパキュービック法など一般的な手法を用いて，スキップした画素やフレームの情報を補完し処理対象に反映させることにより，出力結果の精度を一定のレベルで保証することはできる．しかし RaVioli が行う解像度変更は，あくまで演算量低減のためのものであり，解像度変更自体のために演算量を増加させるべきではない．そもそも RaVioli の目的は，CPU リソース量に制限がある汎用 PC において動画像処理のリアルタイム性を維持するうえで，プログラマに負担を与えず演算量を自動的に削減することであり，また多少精度が落ちても処理を成立させること，そしてなるべく動画像処理プログラムの意図に応じた処理結果を出力することである．このため RaVioli は「優先度」というインタフェースを提供している．プログラマはこの優先度により，動画像処理プログラムにとって本質的に重要である解像度（1 フレームの構成画素数またはフレームレート）を維持することができる．

5. 関 連 研 究

並列処理の分野では，一般的に用いられる並列処理パターンをライブラリの形で抽象化して提供する並列スケルトン[5] の考え方などが古くからあるが，画像処理の分野でも，STL を基本とするテンプレートを用い抽象化したライブラリに VIGRA[6] がある．画像の反転や回転，エッジ処理などの基本的な処理から，ガウスやガボールに代表されるフィルタ処理，画像の分析処理などを抽象化して提供している．また MAlib[7] や OpenCV[8] は，画像処理の一般的なアルゴリズムを C の関数や C++のメソッドとして提供している．また OpenCV は Video4Linux2 を使用することにより，IEEE1394 カメラ経由のデータに対するリアルタイム処理も提供する．

一方 RaVioli のアプローチは，従来の抽象化ライブラリのように単純に処理内容を抽象化してユーザライブラリの形で提供するものとは完全に異なり，処理量に直接関係する 1 フレームの構成画素数やフレームレートをプログラマから隠蔽することを目的としている．プログラマは画像に対する処理内容を，解像度を考慮することなく記述できることから，従来の抽象化ライブラリと比べて，動画像処理の詳細なアルゴリズムを簡単に実現することが可能になる．

トレードオフの関係にある処理精度および処理時間を動的に調整するアプローチとしては，複数アルゴリズムの切換えがある．たとえば，不完全な演算の結果を利用することで，

与えた計算時間の長さに応じて精度が向上するモデル（Imprecise Computation Model）が提案されている[9]．またこのモデルに基づき，処理精度および処理時間に関して経験的に得た知識を利用することで，プログラマがあらかじめ記述した複数のアルゴリズムから，状況に応じて適したアルゴリズムを動的に選択する信頼度駆動アーキテクチャも提案されている[10]．しかしこの方法では，処理を計算負荷の異なる複数のアルゴリズムで実装する必要があり，依然プログラマに対する負担は大きい．

一方 Streaming VIOS[11] は，動的に処理画素数を変更し，プログラマから指定された画素座標を現処理画素数の対応座標に読みかえることのできるライブラリである．これは RaVioli に近い考え方であるが，Streaming VIOS の提供する枠組みではほとんどの動画像処理において処理画素数を透過的に扱うことはできない．このため，Streaming VIOS ではプログラマが，現解像度に依存するパラメータを取得したうえでそのパラメータを用いた処理を行う必要がある．

これらに対し本論文で提案する RaVioli は，動画像処理の抽象化と処理精度・処理時間の自動調整を両立させることのできるものである．行いたい処理に対してプログラマは複数のアルゴリズムを記述する必要はなく，RaVioli 側が処理対象となる画像の構成画素数およびフレームレートを状況に応じて自動的に変更することで処理精度・時間を調整する点で，既存手法とは完全に異なる．

また金井らは数式エディタを用いて記述できる独自の記述言語を用いることにより，画像処理プログラミングを抽象化している[12]．処理単位となる画素配列の大きさを定義し，その配列の要素に対して処理を記述しループレスな記述ができるという点は RaVioli と似ているが，この記述言語は構成画素数を明示的に指定する必要があるため，RaVioli で行っている動的な構成画素数の変動には対応していない．

6. おわりに

本論文では，動的に使用可能な CPU リソースが変動するような環境において，解像度を自動で変動させることによりリアルタイム動画像処理を保証する解像度非依存型動画像処理ライブラリ RaVioli を提案した．さらに，変動する解像度を考慮したプログラミングは複雑であることから，RaVioli は解像度に依存しないプログラミングパラダイムを提供することを示した．解像度を隠蔽するために動画像をカプセル化し，動画像処理に対するさまざまな高階メソッドを実装することでこのプログラミングパラダイムを実現した．

ハフ変換による顔検出プログラムと，フレーム間差分の検出プログラムを用いて RaVioli の評価を行った．評価の結果，プログラムをいっさい変更することなく期待する動作が得られ，かつ擬似的にリアルタイム性を保証した処理ができることを確認した．

今後の課題として，処理結果に依存した動的な優先度切替え機能の実装があげられる．たとえば動画像中に現れる人物の検出を行う場合，歩行中の人物の有無を認識するためには高いフレームレートでサンプリングを行うことが重要であるのに対し，人物の顔などの詳細な認識を行うには 1 フレームの構成画素数が多いことが重要である．したがって，歩行中の人物が現れるまではフレームレートを優先した処理を行い，人物が検出された時点で構成画素数を優先した処理を行う優先度の切替えが必要であると考える．よってプログラマが，過去の処理結果によって明示的に優先度切替えができ，さらに部分的（たとえば顔領域）に優先度を指定できるインタフェースの導入を検討する．

また最近は，家電機器などに実装されている組み込み OS を対象とした高速化や省電力化への要求が高まっていることから[13]，RaVioli でも高速化および省電力化手法を検討する．画像処理関数の入力値が RGB 色情報という限られた範囲の値であることに着目し，たとえばグレースケール化された画像において関数の入力数は，ビット深度次第では 256 にとどまり，2 値化された画像では黒と白のみとなる．このことから，関数の入力値に対応した出力値を表にそれぞれ記憶しておき，再度同じ関数が同じ入力値で呼び出された場合に，出力値を表から呼び出すことにより計算を省略する「メモ化手法」[14] を RaVioli に追加実装する．これにより，計算の省略によるさらなる高速化や省電力が見込める．

さらに，ストリーミング画像処理はパイプライン的に演算処理を行うことが可能な処理が多い．したがって，比較的演算量の多い処理の場合を考慮した，複数の計算機を用いたパイプライン処理機能の実装もこのライブラリの大きな課題の 1 つである．

参考文献

1) 細田寛人：車載用画像認識プロセッサ IMAPCAR，NEC 技報，Vol.59, No.5, pp.22–25 (2006).
2) Sato, H. and Yakoh, T.: A real-time communication mechanism for RTLinux, *Industrial Electronics Society* (*IECON 2000*), Vol.4, pp.2437–2442.
3) 馬場功淳，江島俊明：HeadFinder：単眼視動画像を用いた複数人追跡，画像センシングシンポジウム，pp.363–368 (2001).
4) Ganter, M.A.: From Wire-Frame to Solid-Geometric: Automated Conversion of Data Representations, *Computers in Mechanical Engineering*, Vol.2, No.2, pp.40–45 (1983).
5) Campbell, D.K.G.: Towards the Classification of Algorithmic Skeletons, Technical

report, Dept. of Computer Science, Univ. of York (1996).
6) Köthe, U.: VIGRA — Vision with Generic Algorithms, 1.6.0 edition (2008).
7) 飯尾　淳，谷田部智之，比屋根一雄，米元　聡，谷口倫一郎：動画像処理ライブラリ MAlib を利用した 3 次元ユーザインタフェースの実装，オブジェクト指向 2002 シンポジウム（OO2002），情報処理学会，pp.117–120 (2002).
8) Bradski, G. and Kaehler, A.: *Learning OpenCV: Computer Vision With the Opencv Library*, Oreilly & Associates Inc. (2008).
9) Liu, J., Shih, W.-K., Lin, K.-J., Bettati, R. and Chung, J.-Y.: Imprecise Computations, *Proc. IEEE*, Vol.82, pp.83–94 (1994).
10) 吉本廣雅，有田大作，谷口倫一郎：実時間ビジョンシステムのための信頼度駆動メモリ，情報処理学会論文誌，Vol.44, No.10, pp.2428–2436 (2003).
11) 奥村文洋，松尾啓志：疑似リアルタイム機能を備えた動画像処理系 Streaming VIOS の開発，第 2 回動画像処理実利用化ワークショップ，精密工学会，pp.62–65 (2001).
12) 金井達徳，瀬川淳一，武田奈穂美：組み込みプロセッサのメモリアーキテクチャに依存しない画像処理プログラムの記述と実行方式，情報処理学会論文誌：コンピューティングシステム，Vol.48, No.SIG 13(ACS 19), pp.287–301 (2007).
13) 金井　遵，佐々木広，近藤正章，中村　宏，天野英晴，宇佐美公良，並木美太郎：性能予測モデルの学習と実行時性能最適化機構を有する省電力化スケジューラ，情報処理学会論文誌：コンピューティングシステム，Vol.49, No.SIG 2(ACS 21), pp.20–36 (2008).
14) Norvig, P.: *Paradigms of Artificial Intelligence Programming*, Morgan Kaufmann (1992).

付　　録

A.1　ハフ変換のプログラム例

4.1 節で述べた顔検出プログラムの一部を**図 9** に示す．このプログラムはエッジが抽出された 2 値画像に対して，円のハフ変換を適用する部分の記述を抜き出したものである．この処理は 1 つの黒画素に対して，その黒画素が構成要素となりうる円をすべて検出するという過程を，画像中のすべての黒画素に対して行う．そして検出された円は，x 座標，y 座標，円の半径をインデックスとする 3 次元空間の該当部分に投票される．

以上の円検出処理をプログラムとして記述するためには，画像の幅と高さの画素数に基づいた配列を定義する必要があるが，構成画素数が隠蔽された RaVioli ではこれを定義できない．このため RaVioli には画像における，座標の情報を保持する RV_Coord クラス，距離の情報を保持する RV_Length クラスがあり，またこれらのクラスを元に配列を生成する RV_Array クラスがある．RV_Coord クラスおよび RV_Length クラスは，それぞれが内部

```
RV_Length radius_max, radius_min;
RV_Coord point, start, end;
RV_Array<RV_Array<int> > counter;

void newCounter(RV_Array<int>* factor){
  factor->setSize(radius_max);
}
void dataSet(RV_Array<int>* factor,
             RV_Coord coord){
  RV_Length radius;
  int tmpValue;
  radius=(point-coord).getLength();
  if(radius < radius_max){
    if(radius > radius_min){
      tmpValue=factor->getData(radius);
      tmpValue++;
      factor->setData(radius,tmpValue);
    }
  }
}
void hough(RV_Pixel* p,RV_Coord coord){
  if(p->getR() == 0) {
    point=coord-start;
    counter.procCoord(dataSet);
  }
}
void serialProc(RV_Image* Fnew){
  RV_Length areaWidth;
  RV_Length areaHeight;
     :  //前処理
  areaWidth=(end-start).getXLength();
  areaHeight=(end-start).getYLength();
  if(areaWidth < areaHeight)
    radius_max=areaWidth/2;
  else radius_max=areaHeight/2;
  radius_min=radius_max/10;
  counter.setSize(areaWidth,areaHeight);
  counter.procVal(newCounter);
  Fnew->procCoord(hough,start,end);
     :  //ハフ逆変換，出力
}
int main(){
  RV_Streaming video;
  video.StreamProc(serialProc);
}
```

図 9　RaVioli を用いたハフ変換による円抽出（部分）
Fig. 9　Circle detection program using hough transform with RaVioli.

で持つ値をクラス外から隠蔽しており，代わりに座標どうしとの四則演算や数値との乗除演算ができるようなオペレータを提供する．つまりプログラマは，これらクラスのオブジェクトに格納された幅および高さを利用して画像中の任意の位置や距離を指定することができる．

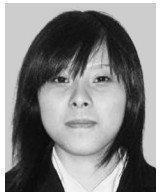
これらのクラスを使ったハフ変換のプログラムの処理内容について概説する．`main` 文では RV_Streaming インスタンス `video` を定義し，1 フレームに対する処理を記述した関数 `serialProc` を動画像中の該当フレームに適用させる．関数 `serialProc` では，まず画像に対して 2 値化，人が存在する可能性のある領域の決定，エッジ抽出といった前処理を行う．なお変数 `start` および `end` を，人が存在する可能性のある領域の左上と右下の座標とする．次に抽出する円の半径の最大値 `radius_max` および最小値 `radius_min` を決定する．次に円候補に投票する 3 次元配列の RV_Array オブジェクト `counter` を生成する．メソッド `setSize` は配列に対して 1 次元および 2 次元のサイズを確保するメソッドであり，次元数は引数の数に対応している．またメソッド `procVal` は高階メソッドであり，`setSize` メソッドにより生成された 2 次元配列の各要素に 1 次元配列を生成することで 3 次元配列を実現する．次に `start` から `end` に関数 `hough` を適用する．関数 `hough` では黒画素 `p` の座標を `point` として一時的に大域変数に格納しておき，`counter` の高階メソッド `procCoord` を用いて `p` を構成要素とするすべての円候補を検出し，`counter` に投票する．

(平成 20 年 5 月 13 日受付)
(平成 20 年 11 月 28 日採録)

(担当編集委員　橋本 学)

岡田慎太郎（学生会員）

1983 年生．2007 年名古屋工業大学工学部電気情報工学科卒業．現在，同大学大学院工学研究科情報工学専攻修士課程在籍．動画像処理プログラミング，並列コンピューティング等に興味を持つ．



桜井　寛子（学生会員）

1985 年生．2008 年名古屋工業大学工学部情報工学科卒業．現在，同大学大学院工学研究科創成シミュレーション工学専攻修士課程在籍．動画像処理プログラミング等に興味を持つ．

津邑　公暁（正会員）

1973 年生．1998 年京都大学大学院工学研究科情報工学専攻修士課程修了．2001 年同大学大学院情報学研究科博士後期課程学修認定退学．同年同大学院経済学研究科助手．2004 年豊橋技術科学大学工学部助手．2006 年名古屋工業大学大学院工学研究科助教授．2007 年同准教授．博士（情報学）．計算機アーキテクチャ，並列処理応用，脳型情報処理等に関する研究に従事．日本神経回路学会，電子情報通信学会，ACM，IEEE-CS 各会員．

松尾　啓志（正会員）

1960 年生．1985 年名古屋工業大学大学院修士課程修了．1989 年同大学大学院博士後期課程修了．同年同大学電気情報工学科助手．1993 年同講師．1995 年同助教授．2003 年同教授．2004 年同大学情報工学科教授．工学博士．計算機工学，分散協調システムに関する研究に従事．IEEE，人工知能学会，電子情報通信学会各会員．