解像度非依存型画像処理ライブラリの提案と実装

岡田　慎太郎[†]　津　邑　公　暁[†]　松　尾　啓　志[†]

使用可能リソースが時々刻々変化する汎用環境において，リアルタイム動画処理を保証することは一般に困難である。この解決法として動的に解像度を変化させ計算量を調節することが考えられる。本稿では，解像度を意識せずに画像処理を記述可能とするライブラリを提案する。対象画像の構成画素を隠蔽することで解像度変更に伴うバグ混入の可能性を低下させ，プログラムの可搬性も向上させる。様々なアルゴリズムを提供し，かつプログラマが自由に機能を追加できる枠組みを提供するため，高階関数を用いてこれを実現した。顔認識プログラムを用いた検証の結果，本ライブラリが十分な記述能力を持つこと，および解像度を変更した場合にプログラムを一切変更することなく期待する処理が行えることを確認した。

Proposal and Implementation of a Resolution-Independent Imaging Library

Shintaro Okada,[†] Tomoaki Tsumura[†] and Hiroshi Matsuo[†]

The amount of available computation resource fluctuates every moment on general-purpose computers. Hence, it is difficult to guarantee realtime video processing. One solution to this problem is changing the resolution of target image according to the resource. This paper describes about a resolution-independent imaging library. Our library conceals element pixels of target images from programmers. It can make programs bug-free and portable. We implement this library using high-order functions for providing both many general algorithms and high extensibility. 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 some different resolutions without any modification.

1. はじめに

計算機の高性能化により，認識アルゴリズム等に代表されるある程度高度な画像処理を，汎用 PC でも行うことが可能となりつつある．一方で高速なインタフェースも普及し，カメラ等からリアルタイムで動画像を取り込むことも容易となった．このため，汎用 PC および汎用 OS 上でのリアルタイム動画像処理が今後広く一般に行われると予想される．

一方で Linux などの汎用 OS では，使用可能なリソースが時々刻々変化するため，1/30 秒もしくは 1/60 秒毎に処理を行う必要のあるリアルタイムストリーミング処理を実現することは困難である．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，一般に毎フレーム処理量が変動する認識処理においてリアルタイム処理を実現するためには，ワーストケースに基づく処理量削減が必要となる．

そこで我々は，解像度およびフレームレートをパラメータ化し，いずれを優先させるかを動的に選択可能とする動画像処理環境を目指し研究を行っている．本稿では，この動画像処理環境の一部となる，解像度非依存型画像処理ライブラリについて述べる．

動的に解像度を変更する場合，それを考慮したプログラミングが必要となる．しかしこれは一般にプログラムを複雑化させ，バグの混入を助長すると同時に，プログラムの可搬性も低下させ，変更に大きな労力が必要となる．よって処理対象画像の解像度，ひいてはそれを構成する画素そのものを意識する必要のないプログラミング環境が望まれる．

本稿で提案する解像度非依存型画像処理ライブラリは，プログラマに対し処理対象画像の解像度および構成画素という概念を隠蔽し，画像処理プログラミングを簡易化，抽象化するライブラリである．多くの一般的な処理アルゴリズムを提供すること，およびプログラマに高い自由度を与えることを両立するため，高階関数を利用してこれを実現する．以下，ライブラリの設計と実装，および評価結果について述べる．

2. 関連研究

画像処理を 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[1] 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

† 名古屋工業大学
Nagoya Institute of Technology

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 OpenCV[2] 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．Linux 向け動画取込みライブラリ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．しかしこれらのライブラリを使用する場合，リソースの動的な変化に応じた解像度変更は容易ではない．

一方で，計算時間に応じて演算出力の精度を向上させる Imprecise 計算モデル[3] に基づく信頼度駆動アーキテクチャが提案されている[4]．しかしこの方法では，処理を計算負荷の異なる 2 種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．

これに対し Streaming VIOS[5] は，動的に解像度の変更を行い，プログラマから指定された画素座標を現解像度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．これは本ライブラリに近い考え方であるが，Streaming VIOS の提供する枠組では，ほとんどの画像処理において解像度を透過的に扱うことはできていない．このため，Streaming VIOS では「画素」は依然プログラマから可視とされており，プログラマが逐一，現解像度に依存するパラメータを取得したうえで処理を行う必要がある．

3. 解像度非依存型画像処理ライブラリ

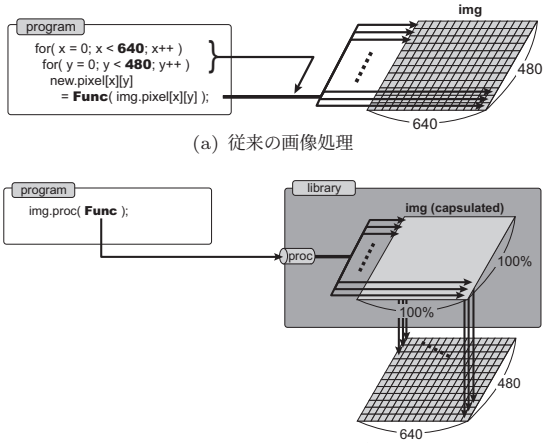
3.1 方　　針

人間の脳が視覚情報に対して認識処理を行う場合，処理の緊急度に応じて瞬時に大局的な判断を行ったり，時間をかけて詳細な分析を行ったりする．すなわち費やすことのできる処理時間に応じて精度を変更している．また，そこには「画素」という概念は存在しない．

これと同様に本ライブラリは，汎用 PC および汎用 OS 環境のように使用可能リソースが時々刻々変化する環境において画像処理を行う際，解像度を動的に変化させることで負荷の調整を行い，実時間処理を実現することを目指す．

この際，記述しやすさおよび可搬性の観点から，プログラマに対して解像度ひいては画像の構成画素を意識させることなく，従来の画像処理よりも抽象度の高い記述が可能となるような環境の提供を目標とする．

本ライブラリでは入力画像はカプセル化されるが，この際画像の構成画素数は隠蔽され，プログラマは画像内の位置や画像全体に対する長さを指定する際はパーセンテージを用いる．例えば 640 × 480 ピクセルの画像の中心を指定したい場合，(320,240) ではなく (0.5, 0.5) と指定する．画像に対する処理は全て画像クラスの持つメソッドを通じて行う．画像クラスの持つメソッドは，「グレースケール化」などの具体的な機能を持つものではなく，ユーザが記述した関数を引数にとり，それを画像全体あるいは指定された部分に適用するという機能を持つ**高階関数** (high-order function) として実装する．



(a) 従来の画像処理

(b) 本ライブラリを用いた画像処理

図 1　画像処理のイメージ

例えばグレースケール化のように，画像の構成画素全てに対して同じ処理を適用する場合を考える（**図 1**）．従来の画像処理プログラミングでは，まず入力画像の構成画素数に応じたループを構成し，そのループ内で各座標に対応する画素に対してユーザが定義した関数を適用することでこれを実現する．対象画像の解像度が変わると，ループイタレーション部分の記述に変更が必要となる（図 1(a)）．

これに対して本ライブラリを使用した場合では，入力画像をカプセル化した画像インスタンスに対し，そのインスタンスが持つ「現解像度における全画素に処理を適用する」高階メソッド（図中 `proc`）に，各画素に対して行うべき処理を定義した関数を渡すことで，この処理が可能となる．この場合，プログラマは入力画像の構成画素数を用いた記述をする必要がなく，解像度が変更された場合でもそれはライブラリにより吸収され，自動的に必要な画素にのみ処理が適用される（図 1(b)）．

3.2 実　　装

前節で述べた機能を実現するにあたり，PIXEL および IMG の 2 つのクラスといくつかの高階メソッドを持つライブラリを C++で実装した．以下それぞれのクラスについて述べる．

PIXEL クラス

画像を構成する 1 画素の情報を保持するクラスであり，R，G，B のそれぞれの値を，8 ビットのメンバ変数として持つ．また，RGB 値を操作するためのメソッドを持ち，プログラマはこのメソッドを通じて画素の色を変更することができる．なおこの際，プログラマはやはり絶対値ではなくパーセンテージで値を指定する．これにより，ビット深度の変更を意識することのないプログラミングを可能にする．

IMG 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 PIXEL インスタンスを構成画素数ぶん配列の形で保持する．また，現解像度に対応したパラメータ stride を持ち，画素アクセスの際の粒度を表す．IMG クラスのもつ高階メソッドは，一般に構成画素全体に対しプログラマから指定された関数を適用するが，例えば stride が 2 のときは画像の構成画素へのアクセスは 1 画素置きとなり，1/4 の解像度で処理が行われる．以下に実装した高階メソッドを示す．

proc()：現 stride でアクセスされる全ての画素に対して指定された関数を適用し，処理結果の画素を出力する．グレースケール化などに利用する．

procNeighor()：現 stride でアクセスされる全ての画素に対して当該画素およびその隣接画素を入力とし，処理結果の 1 画素を出力する．畳み込み演算などに利用する．

procCoord()：現 stride でアクセスされる全ての画素に対して，当該画素およびその相対値座標から中間データを生成する．ハフ変換などに利用する．

trans()：現 stride でアクセスされる全ての画素に対して，定められた対応規則に従って当該画素の座標を変換する．画像の回転、拡大縮小などに利用する．

procBox()：現 stride でアクセスされる全ての画素を始点とし，指定された大きさ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．テンプレートマッチングなどに利用する．

4. 評　　価

顔検出プログラムを用い，ライブラリの記述能力や解像度変更への対応の評価を行った．

4.1 評価プログラム

本ライブラリを評価するにあたり，サンプルプログラムとして色テンプレートを用いた顔検出プログラムを実装した．用いたアルゴリズムは肌色情報を使用する手法[6] に基づく単純なものである．

まず入力画像の肌色の部分とそれ以外の部分で 2 値化した中間画像を得る．ここで HSV 表色において色相値 H が 0〜30 の場合を肌色と判定した．次に，指定した大きさでかつ肌色部の占める割合が最大となるようなウィンドウを画像全体から検索することにより，画像内の顔の位置を検出する．

4.2 記述面における違い

ライブラリ使用の有無によってユーザが記述するプログラムがどのように変化するかを**図 2** に示す．

ライブラリを使用しない場合，まず顔として検出するウィンドウの大きさを，現解像度に応じて変更する必要がある．次に画像内をウィンドウをずらしつつ，ウィンドウに含まれる各画素に対し肌色か否かを判定し，その数を数える．`skinp()` は，当該画素が肌色である場合に `sum` をインクリメントするユーザ定義関数

ライブラリ非使用

```
// 解像度に応じたウィンドウサイズを設定
box.W = img.W * 0.2;
box.H = img.H * 0.3;
int max = 0;
int sum = 0;
Coord target;
// 現解像度に応じた増幅でイタレーション
for(y = 0; y < img.H-box.H-stride; y += stride){
  for(x = 0; x < img.W-box.W-stride; x += stride){
    sum = 0;
    for(by = 0; by < box.H-stride; by += stride)
      for(bx = 0; bx < box.W-stride; bx += stride)
        skinp( img.get_pixel(x+bx, y+by) );
  }
  if( max < sum ){
    max = sum;
    target.x = x;
    target.y = y;
  }
}
```

ライブラリ使用

```
int max = 0;
int sum = 0;
Coord target;
void countSkin( IMG* img, Coord cs, Coord ce ){
  sum = 0;
  img->proc( skinp, cs, ce );
  if( max < sum ){
    max = sum;
    target = cs;
  }
}
int main(){
  // 画像全体に対し幅 20%，高さ 30% の box を
  // ずらしながら countSkin を適用
  img.procBox( countSkin, 0.2, 0.3 );
}
```

図 2　ライブラリ非使用/使用の場合のプログラム（部分）

である．ここで，画像をウィンドウでスキャンする際にも，ウィンドウ内の画素に対し肌色判定を行う際にも，現解像度に応じた差分値 `stride` を用いてループを記述する必要がある．

これに対しライブラリを使用する場合，IMG クラスに定義されている高階メソッド `procBox()` を用いる．プログラマは IMG インスタンス `img` が持つ幅や高さを意識する必要がなく，高階メソッド `procBox()` に，プログラマが別途定義したウィンドウ内に対する処理を行う関数 `countSkin()`，および画像全体の大きさに対する比で指定したウィンドウの大きさを渡すだけでよい．`procBox()` は，比で指定されたウィンドウの大きさを現解像度に合わせて画素数に読みかえ，その大きさのウィンドウを画像内で現解像度に沿った stride でスキャンしつつ，`countSkin()` で定義されたウィンドウ内に対する処理を行う．

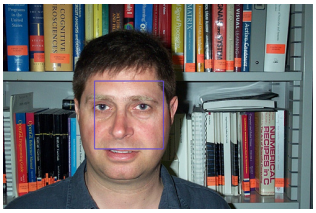
このように，プログラマが現解像度や画像の真のサイズを使用することなく，ライブラリを使用しない場合と全く同じ処理を記述することができる．また，本来現解像度を強く意識して記述する必要のある部分であるループなどの繁雑な箇所が省かれることで，バグの混入可能性を低下させるだけでなく，プログラムの記述自体も少なく済む．

4.3 出 力 結 果

前節で示した顔認識プログラムを，我々が開発した

（a）出力画像 1

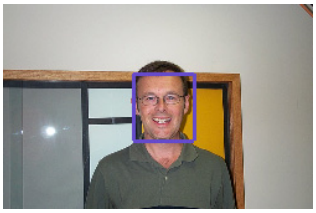
（b）出力画像 2



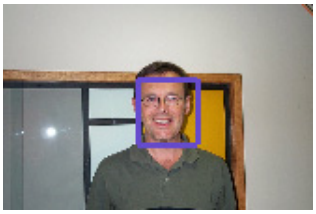
（c）出力画像 3

図 3　顔検出の出力画像

（a）stride=1 の場合



（b）stride=2 の場合



（c）stride=3 の場合

図 4　各解像度における出力画像

解像度非依存型画像処理ライブラリを用いて記述し，カリフォルニア工科大の Computer Vision☆ で配布されているサンプル画像に対して処理を行った結果を**図 3** に示す．これら出力画像から分かるように，さまざまな入力画像に対し正しく顔の位置が検出できており，ライブラリの記述能力および処理の正当性が確認できた．

次に，解像度を 1/4 (stride = 2) および 1/9 (stride = 3) に変更した場合の結果を**図 4** に示す．この結果から，プログラムを一切変更することなく解像度を変更に対応できており，期待される結果が得られることを確認した．なお，(a) に比べて (c) の出力結果に若干の違いがあるが，これは画像が低解像度になったために精度が落ちたためであり，本ライブラリの実装に起因するものではない．

5. おわりに

本稿では，動的に使用可能リソースが変動するような環境においてリアルタイム動画像処理を行うために解像度の動的な変更が必要であることを示し，これを容易にするための解像度非依存型画像処理ライブラリについて述べた．

プログラマから入力画像の解像度および構成画素を隠蔽し，画像全体に対する相対的な座標および長さのみを使ってプログラマに操作させることで，プログラムの記述容易性および可搬性が向上することを示した．

また解像度非依存型画像処理ライブラリを実装し，顔認識プログラムを用いてその評価を行った．評価の結果，当ライブラリが十分な記述能力を持つこと，および解像度を変更した場合にもプログラムを一切変更することなく期待する動作が得られることが分かった．

今後の課題としては，より多種な画像処理プログラムを用いた記述能力の検証および高階メソッドの拡張，ライブラリを使用した場合のプログラム動作速度の検証とその改良，そして動画処理に対応するためのライブラリの拡張などが挙げられる．

謝辞　本研究の一部は，文部科学省科学研究費補助金（萌芽研究 18650020）による．

☆ http://www.vision.caltech.edu/

参 考 文 献

1) Köthe, U.: Generic Programming for Computer Vision: The VIGRA Computer Vision Library, http://kogs-www.informatik.uni-hamburg.de/~koethe/vigra/ (2006).
2) Intel Corp.: *Open Source Computer Vision Library* (2001).
3) Liu, J., Shih, W.-K., Lin, K.-J., Bettati, R. and Chung, J.-Y.: Imprecise Computations, *Proceedings of IEEE*, Vol.82, pp.83–94 (1994).
4) 吉本廣雅，有田大作，谷口倫一郎：実時間分散画像処理システムのための信頼度駆動アーキテクチャ，情処研報 2004-ARC-149 (HOKKE 2004), pp.85–90 (2004).
5) 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).
6) Yao, H. and Gao, W.: Face Detection and Location Based on Skin Chrominance and Lip Chrominance Transformation from Color Images, *Pattern Recognition*, Vol. 34, pp. 1555–1564 (2001).