5L-05　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*

奥村文洋　　松尾啓志†

名古屋工業大学電気情報工学科‡

1　はじめに

近年、人物追跡など、リアルタイム動画像処理に関する研究が盛んに行われている。また、計算機の価格低下により、高い計算能力を持つ計算機を容易に手に入れることが可能となり、種々の動画像処理アルゴリズムが提案、実装されている。

しかし、汎用OS上での実装では厳密なリアルタイム性を保証することは困難であり、動画像処理アルゴリズム側での工夫が必要となる。

そこで本研究では、解像度を動的に変化させることにより、汎用OS上で疑似的なリアルタイム動画像処理を実現可能とする動画像処理環境Streaming VIOSを提案するとともに、処理能力の検討を行う。

2　リアルタイム性の維持

一般に汎用OSにおいて、リアルタイム性の保証は困難である。主な要因として

- 画像の内容に依存する処理負荷の変動
- 他のプロセスなどが原因となる、動画像処理に分配されるCPUリソースの減少
- I/Oなどによる待ち時間

などが挙げられる。

本研究では汎用OS上で疑似的にリアルタイム性を維持する手法として、解像度の変更による動画像処理負荷の動的な変更を提案する。これは、処理画像の解像度、すなわち画像のサイズを変更することで、動画像の処理負荷を変更する手法である。本手法は、動画像からの人物追跡などの処理画像を直接結果としない処理に対して有効である。

*Psuedorealtime streaming processing environment –Streaming VIOS–

†Bunyou Okumura, Hiroshi Matsuo

‡Department of Electrical and Computer Engineering, Nagoya Institude of Technology

3　動画像処理系 Streaming VIOS

汎用OS上で、疑似的なリアルタイム性の維持が可能な動画像処理プログラムを容易に記述するための環境として、動画像処理系Streaming VIOSを提案する。機能として

- 解像度を意識することなくアルゴリズムの記述が可能な解像度非依存画像
- 変数に時刻の概念を導入したストリーミング処理機能
- 疑似リアルタイム性の維持を目的とした、処理負荷の自動的な変更機能

などを備えるC++ライブラリである。動作環境は、C++クラス及びC++テンプレートが利用可能な環境である。

4　Streaming VIOSの構成

Streaming VIOSはフレームレートの調節及び解像度の管理を行うStreaming VIOS本体及び、各種型から構成される。各種型の概略を以下に示す。

- 解像度非依存画像型
  解像度非依存画像は解像度（サイズ）を意識することなく画素へのアクセスを可能とする。これは以下の2種類に大別される。
  StrideImage 内部的に1つの画像を持ち、要求された画素の座標を変換することで画像へのアクセスを行う。用途として主に画像の入力に用いる。
  LayeredImage 内部的に各解像度に対応する画像を持ち、画像は必要に応じて解像度間でコピーされる。画像の出力などの汎用的な用途に用いる。
- Stream型
  テンプレートクラス`Stream<>`により、変数の各時刻における値を蓄え、適時過去の値へのアクセス

を可能とする。これは時系列処理の容易な記述を提供する。

- Layered 型
  テンプレートクラス Layered<>により。変数を多重化する。これは各解像度に対応する変数への簡潔なアクセスを提供する。

5 動画像処理の記述方式

ユーザーが解像度非依存画像を用いて動画像処理を記述する場合、処理の記述は以下の 2 種類に大別できる。

5.1 画素を基準とした記述

本記述は、解像度を完全に透過的に扱うことが不可能な部分で用いられる。プログラムは現在の画像から解像度に依存するパラメータを獲得し、その値を用いて画素への処理を記述する。この記述はほぼ全ての画像処理を記述する場合に必要となる。

5.2 画像を基準とした記述

本記述は、解像度を意識することなく処理を記述することを可能とする。すなわち、解像度に依存するパラメータを本来の解像度における値に変換することで解像度を意識させない。この記述は (5.1) だけでは不十分である、解像度に関する透過性を提供する。

5.3 Streaming VIOS を用いたプログラム

以下に画像処理を行い目標の位置を特定し、更に時系列情報を用いて追跡処理を行うプログラムの画像処理部分の例を示す。

Streaming VIOS によるプログラム例

```
Stream<POSITION> tPos;//目標の位置の履歴
IMAGE   in1, in2, out;
POSITON pos, posOnImg;
for( int h=0; h<out.height(); h++ )
  for( int w=0; w<out.width(); w++ )
    各画素に対する処理{
      //変数 pos に目標の位置を格納
      //ここで画素基準の値が代入される
      pos.x = w;  pos.y = h;
    }
//画素基準から、画像基準へ変換
posOnImg = out.GetOrgPos(pos);
tPos.Add(posOnImg); //ストリームへ追加
pos = GetSeries(tPos); //時系列情報を利用
```

上記のような記述により、時系列情報を用いる処理の記述部分は解像度の変更による処理負荷の軽減を考慮する必要がなくなり、プログラマは本来のアルゴリズムの記述に専念することが可能となる。

6 実験

Streaming VIOS の動作例として、カメラからの画像のフレーム間差分を行い、差分領域の重心の時系列情報を用いた簡単な人物追跡を実装する。さらに処理負荷を加え、解像度が変化することを確認する。実験環境は Pentium3 500MHz,Memory 512Mbyte の PC を用い、Linux 上に実装した。また、保証するフレームレートは 15frame/second、誤差 15 % を許容範囲とした。

動作状態を以下に示す（図 1、図 2）。負荷が低い場合は処理解像度が高く、負荷が高い場合には処理解像度が低下している。また、低解像度においても、人物の追跡が可能であることを確認した。

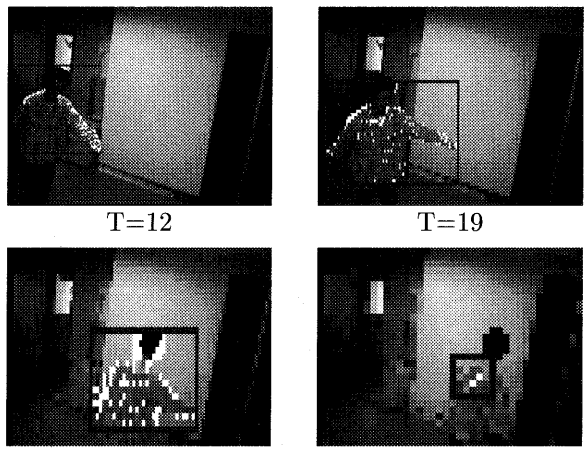

図 1: 処理結果画像

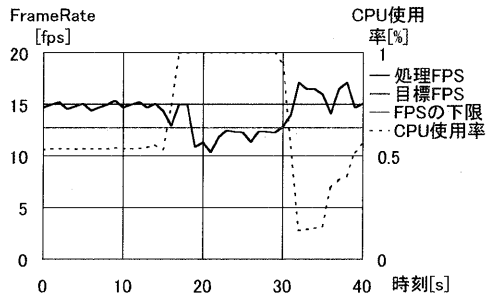

図 2: フレームレート及び CPU 使用率の変化

7 まとめ

汎用 OS 環境を用いて、疑似リアルタイム性を維持しつつ動画像を処理可能な環境として動画像処理系 Streaming VIOS を開発した。この環境は処理負荷に応じて画像の解像度を変更することにより疑似リアルタイム性の実現を可能とする。また、解像度の変更を考慮する必要のない簡潔な記述を提供可能である。さらに、実際に実装を行い、処理負荷の変動に対して疑似リアルタイム性の維持が可能であることを確認した。