コンピュータビジョンと
イメージメディア 115–17
（１９９９．３．１９）

再帰相関法とマルチメディア命令による高速オプティカルフロー計算法

東京大学情報工学専攻　○ 岡田 慧　加賀美 聡　稲葉 雅幸　井上 博允

概要： 局所相関による密なオプティカルフローの計算を再帰相関法を用いてアルゴリズム的に高速化した後，キャッシュを目的とした最適化を行い，最後にマルチメディア命令を用いて実時間で計算する方法について述べる．次に得られた結果の一貫性を評価する手法を提案し，信頼性の高い結果のみを検出することが可能であり，PC を中心としたシステム構成によりロボットへ応用が可能であることを示す．

Real-time Optical Flow Generation using Recursive Correlation Technique and Multi Media Instruction Set

Univ. of Tokyo　○ K. OKADA　S. KAGAMI　M. INABA　H. INOUE

Abstract : Optical flow has many information for a robot which moves in a real-world. This paper describes a development of real-time optical flow generation system using two-dimensional recursive correlation technique and Pentium MMX instruction set. The process includes: 1) Optimize algorithm by recursive correlation, 2) Extract algorithm in order to utilize second CPU cache, 3) MMX implementation. Finally several implementation and system evaluation are denoted.

1　はじめに

実世界で自律的に行動するロボットにとって，環境や自己の動きを知覚できるオプティカルフローは認識と行動のために本質的に重要である．特に小さな動きや動いている対象物の形状まで検出を可能にするためには，密度の高いオプティカルフロー生成が高速で行えるシステムが重要になってくる．オプティカルフローを実時間で得ることが出来れば，より上位の認識処理を行いながら行動するロボットの研究が可能になる．

これまで画像処理の研究分野ではオプティカルフローの研究が進み，様々なマッチング手法や精度を向上のための手法の研究が行われている．またロボット視覚の分野では高速なオプティカルフローを求めるために，相関演算ハードウエアを用いたアプローチが行われている [1,2]．しかしこの手法では，動きの早い物体の前後にオクリュージョン（隠れ領域）が発生し，この領域を発見できないことが実用上からは問題であった．

一方近年汎用 CPU がマルチメディア命令と呼ばれる一命令で複数のデータ処理を行う機能を備え，DSP や画像処理プロセッサよりも高速に画像処理を行うことが可能になってきた．

本論文ではこのような背景をふまえ，画像処理の分野で行われている再帰的相関演算手法を二次元に拡張することによりオプティカルフロー生成アルゴリズムを高速化し，最も普及している Pentium プロセッサの MMX 命令を用いて最適化し，市販の PC 部品のみを組み合わせて実時間で密度の高いオプティカルフローを得ることが可能であることを示す．

2　ロボット搭載用オプティカルフロー生成システム

2.1　オプティカルフロー

オプティカルフローは 3 次元的な動きの画面への投影である [3, 4]．オプティカルフローは多くの場合は一つのカメラから得られた連続する二つの画像中の各点の対応点を探索する問題である．これまで画像の微分からフローの方向を得る方法と，相関演算により近似的に局所領域をマッチングする手法の二種類が研究されている．局所領域のマッチングによる方法では相関演算ハードウエアによる高速化といったアプローチも存在する．いずれの手法も連続する画像での対応する領域はあまり離れていない，という仮定の元に探索範囲を拘束できる．またオプティカルフローでは前述のように原理的にオクリュージョン（隠れ領域）問題が発生し，この領域を発見可能なことが実用上から重要である．一般的に局所領域のマッチング手法は以下のように表される．

1. 画像のノイズ除去，正規化等の前処理を行う
2. 各小領域に対して可能な探索範囲で対応する各小領域ごとに相関演算を行う
3. それらの中から適切な対応点を選択する
4. （必要なら）サブピクセルレベルでフローを推定する

本論文では視差画像を高速に計算するために提案された再帰相関演算手法 [5, 6] を 2 次元に拡張することにより 2. の相関演算を，Left-to-Right Right-to-Left Consistency Checking(LR-check)) 法 [7, 8] を応用し 3. の

対応点の選択を行う．実時間で密度の高いオプティカルフローの生成が可能となるように次節に述べるような 3 段階の高速化を行った．

2.2 オプティカルフロー画像生成システムの高速化

ロボット搭載用システムでは，小型化と高速化が重要になる．本稿では PC/AT 互換機上で動作するオプティカルフロー生成システムを開発した． PC をロボットに搭載することで低次から高次のロボットの行動をコンパクトなシステムで実現でき [9]，また近年汎用 CPU の性能の向上が著しく，後述するように処理によっては専用ハードウエアよりも高速に画像処理を行うことが可能になっている．ソフトウエアによる距離画像を生成する処理のみではオペレーションシステム (OS) を選ばないが，システムを構築する際には画像取り込みハードウエアのデバイスドライバを制御する必要があることから，ここでは 便宜上筆者らがロボットで使用している Linux を採用する．

以下第 3,4 節では，オプティカルフロー生成を行う高速アルゴリズムと，その実現について以下の 3 つの段階に分けて述べる．

1. 再帰相関演算によるアルゴリズムの効率化
2. キャッシュを意識したアルゴリズムの展開
3. MMX 命令による高速化

1,2 は C 言語で， 3 は MMX のアセンブラによって記述する．

3 二次元再帰相関演算

3.1 相関演算

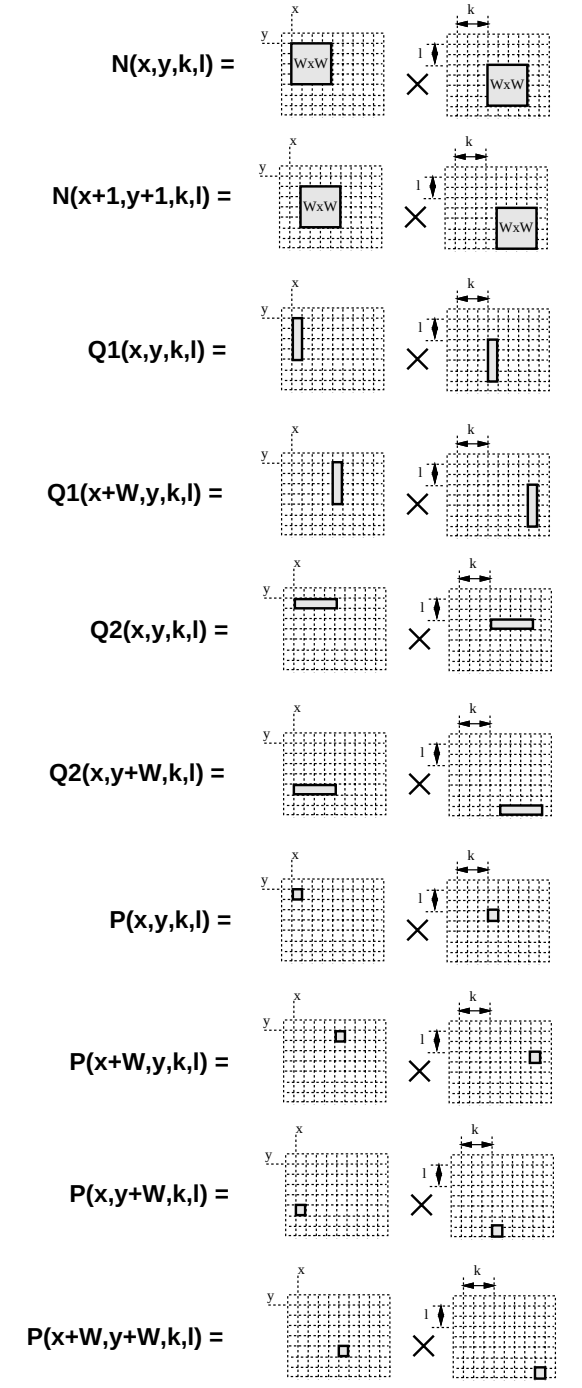
オプティカルフローを計測するためには，連続する 2 枚の画像において，一方の画像の各小領域に対して，もう一方の画像の対応する小領域を求める必要がある．各小領域の対応を計算するためには局所領域の相関値を求める相関演算を用いる．画像の各小領域の輝度値の平均が等しいと仮定すると例えば式 (1) のような正規化項を含んだ形が，また，画像の各小領域の輝度値の分散が等しいと仮定し，式を簡略化すると例えば式 (2) の SAD を利用することができる[1]．

$$C_2(x,y,d)=\frac{\sum_{i,j}\{I_1(x+i,y+j)\times I_2(x+i+k,y+j+l)\}}{\sqrt{\sum_{i,j} I_1(x+i,y+j)^2}\times\sqrt{\sum_{i,j} I_2(x+i+k,y+j+l)^2}} \quad (1)$$

[1] 仮定を用いない一般的な相関演算は以下のように表される．

$$\frac{\sum_{i,j}(I_1(x+i,y+j)-\overline{I_1(x,y)})\times(I_2(x+i+k,y+j+l)-\overline{I_2(x,y)})}{\sqrt{\sum_{i,j}(I_1(x+i,y+j)-\overline{I_1(x,y)})^2}\times\sqrt{\sum_{i,j}(I_2(x+i+k,y+j+l)-\overline{I_2(x,y)})^2}}$$

ここで $\overline{I(x,y)}$ は $\frac{1}{W^2}\times\sqrt{\sum_{i,j} I(x+i,y+j)^2}$ を表す．



N(x+1,y+1,k,l) = N(x,y,k,l)
+Q1(x+W,y,k,l) - Q1(x,y,k,l)
+Q2(x,y+W,k,l) - Q2(x,y,k,l)
+P(x,y,k,l) + P(x+W,y+W,k,l)
-P(x,y+W,k,l) - P(x+W,y,k,l)

Q1(x,y+1,k,l) = Q1(x,y,k,l) + P(x,y+W,k,l) -P(x,y,k,l)

Q2(x+1,y,k,l) = Q2(x,y,k,l) + P(x+W,y,k,l) -P(x,y,k,l)

Fig. 1: Recursive correlation calculation

$$C_1(x,y,k,l) = \sum_{i,j} |I_1(x+i,y+j) - I_2(x+i+k,y+j+l)| \tag{2}$$

ただし，画素 (x,y) での輝度値を $I_1(x,y), I_2(x,y)$，x,y の値域を $0 \leq x,y < N$ とする．また，相関演算のウインドウサイズを $W \times W$ とし，その値域を $0 \leq i,j < W$ とする．さらに，探索範囲 k,l の値域を $0 \leq k,l < D$ とする．x,y の値を固定したときに最も相関度の高い視差 k,l は，式 (2) では最も値の大きいもの，式 (1) では最も値の小さいものとして与えられる．

以下，本節では式 (2) を例に高速化の手法を述べる．

3.2 二次元再帰相関演算アルゴリズム

再帰相関演算と呼ぶアルゴリズム [5, 6] が視差画像を高速に計算する手法として提案されている．この方法を 2 次元に拡張することによりオプティカルフローの相関演算の計算量を $O(N^2W^2D^2)$ から，$O(N^2D^2)$ とすることが可能である．以下にアルゴリズムを説明する．

$$P(x,y,k,l) = I_1(x,y) - I_2(x+k,y+l) \tag{3}$$

とすると，式 (2) の相関演算は，

$$N(x,y,k,l) = \sum_{i=0}^{W-1} \sum_{j=0}^{W-1} P(x+i,y+j,k,l) \tag{4}$$

と書ける．ここで，

$$\begin{aligned} Q1(x,y,k,l) &= \textstyle\sum_{j=0}^{W-1} P(x,y+j,k,l) \\ Q2(x,y,k,l) &= \textstyle\sum_{i=0}^{W-1} P(x+i,y,k,l) \end{aligned} \tag{5}$$

とすると，$N(x+1,y+1,k,l)$ は

$$\begin{aligned} & N(x+1,y+1,k,l) \\ = \;& N(x,y,k,l) \\ & +Q1(x+W,y,k,l) - Q1(x,y,k,l) \\ & +Q2(x,y+W,k,l) - Q2(x,y,k,l) \\ & +P(x,y,k,l) + P(x+W,y+W,k,l) \\ & -P(x+W,y,k,l) - P(x,y+W,k,l) \end{aligned} \tag{6}$$

と再帰的に計算できる (図 1)．さらに $Q1(x,y,k,l)$，$Q2(x,y,k,l)$ 自身も再帰的に計算できる．

3.3 二次元再帰相関演算によるフロー計算

式 (2) に基づくオプティカルフローの計算は，以下のように書くことができる．

$$\begin{aligned} O(x,y) &= \max_{k,l}\{N(x,y,k,l)\} \\ N(x,y,k,l) &= \textstyle\sum_{i,j} |I_1(x+i,y+j) \\ & \qquad -I_2(x+i+k,y+j+l)| \end{aligned} \tag{7}$$

ここでは $W \times W$ 回の乗算が冗長である．すなわち，$N(x,y,k,l)$，は再帰相関演算アルゴリズムを用いることで，以下のように再帰的に計算できる．

$$\begin{aligned} P(x,y,k,l) &= I_1(x,y) - I_2(x+k,y+l) \\ Q1(x,0,k,l) &= \textstyle\sum_j P(x,j,k,l) \\ Q1(x,y+1,k,l) &= Q1(x,y,d) \\ & \quad +P(x,y+W,k,l) - P(x,y,k,l) \\ \\ Q2(0,y,k,l) &= \textstyle\sum_i P(i,y,k,l) \\ Q2(x+1,y,k,l) &= Q2(x,y,d) \\ & \quad +P(x+W,y,k,l) - P(x,y,k,l) \\ N(0,y,k,l) &= \textstyle\sum_i Q1(i,y,k,l) \\ N(x,0,k,l) &= \textstyle\sum_j Q2(x,j,k,l) \\ N(x+1,y+1,k,l) &= N(x,y,k,l) \\ & \quad +Q1(x+W,y,k,l) - Q1(x,y,k,l) \\ & \quad +Q2(x,y+W,k,l) - Q2(x,y,k,l) \\ & \quad +P(x,y,k,l) + P(x+W,y+W,k,l) \\ & \quad -P(x+W,y,k,l) - P(x,y+W,k,l) \end{aligned} \tag{8}$$

4 実時間オプティカルフロー生成システム

実時間で密なオプティカルフローを生成するために，前述の再帰相関演算をキャッシュを意識した形で高速化し，次に MMX インストラクションセットによる高速化について述べる．

4.1 キャッシュを意識したアルゴリズムの高速化

図 2は (a) 式 (2) を単純に実現したもの，および (b) 再帰相関演算を単純に実現した際のループの構成法を示している．しかし全画面の対応点を計算するためには，再帰相関演算の式を x, y, k, l の 4 重のループとして計算することになるが，単純に式の通りに実装するとキャッシュを利用できず，計算速度は CPU の演算速度ではなくメインメモリの読み出し／書き込み速度に依存することになってしまう[2]．

そこでキャッシュの利用効率を最大にするために演算途中に使用する $P, Q1, Q2,$ を計算する際に N も同時に計算することとする (図 2(c))．これにより単純にインプリメントすると $P, Q1, Q2, N$ は $N \times N \times D \times D$ の配列サイズをそれぞれ持つことになっていたが，(c) の方法では $P, Q1, Q2$ は $N \times W \times D \times D$，$N$ は $N \times D \times D \times D$，の配列サイズで良いことになる．

第 3.3 節で述べたアルゴリズムに必要なメモリ領域は，N=64, W=16, D=8，各ピクセルのデータ量を 8bit とすると，P のデータ幅は 8bit, $Q1, Q2, N$ のデータ幅は 16bit となることから，約 3.1MB となるが，この方法により約 655KB まで低減することができる[3]．

[2] メインメモリからのデータの読み出し，書き込みは 10 クロック以上かかるが，2 次キャッシュからは 1 または 2 クロックで行える

[3] これ以外に画像領域として $64 \times 64 \times 2 \times 8bit$= 8KB のメモリが必要である

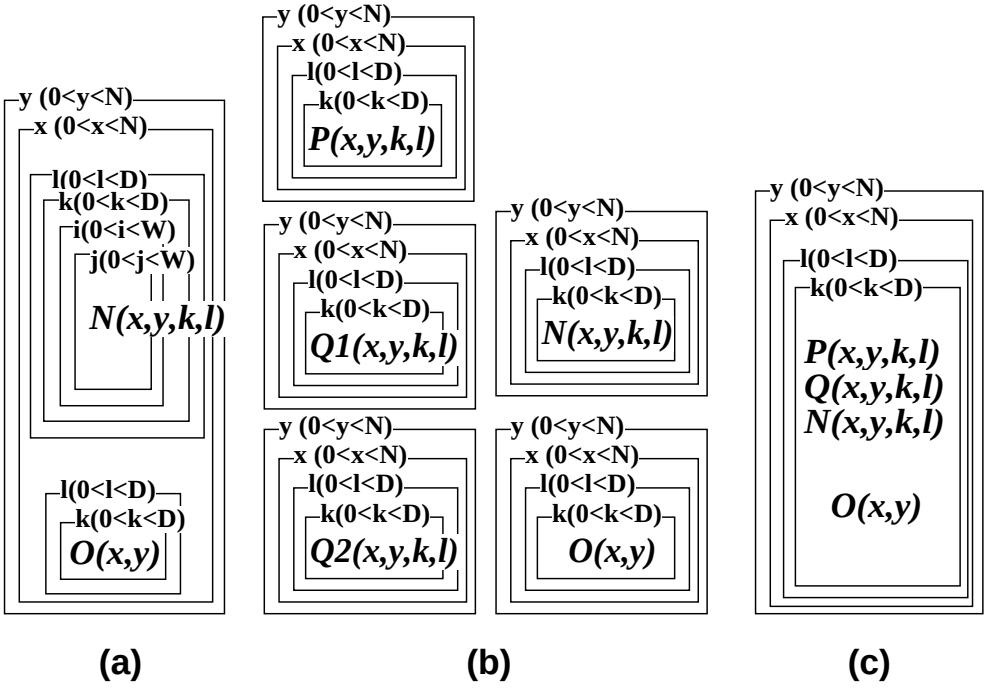


Fig. 2: Optimizing implementation with (a) normal correlation, (b) simple recursive correlation, (c) cache optimized recursive correlation. (Algorithms (c) don't work when x=0 or y=0)

4.2 MMX 命令

ここまでは C 言語によるオプティカルフロー生成の高速化について述べてきた．本節では最も計算時間を要している再帰相関演算の高速化を図るために，Pentium の MMX 命令を利用する．

MMX 命令は Intel の PentiumMMX プロセッサ以降に搭載されたマルチメディア命令で，一クロックで 64bit 長の MMX レジスタ上（あるいはメモリ上）におかれたデータ配列の和，積，積和，論理演算等を行う，いわゆる SIMD (Single Instruction Multi Data) な命令セットで，積以外は一クロックで実行される．ただし割り算命令をもっていないことと，積は 16bit × 16bit までしかできないといった制約がある．

4.3 MMX 命令による実現

式 (2) の $P(x,y,k,l)$ は MMX 命令を用いて図 3 のように実現される[4]．ここでは 8bit 整数の画像配列をレジスタに移した後，16bit 整数に拡張し，4つずつまとめてお互いに差をとり，2つの結果を or することにより条件分岐を行わずに差の絶対値を計算し，P に代入している．このコードは C 言語で書いたものに比べて約 2 倍早く実行される．また，Q1, Q2, N の計算も MMX 命令により実装可能である．

[4] gcc(gas) の記述のためにオペランドのソースとデスティネーションは通常の x86 の方式と逆になっていることに注意されたい．%%mm0,1,2 は 8 個存在する MMX レジスタを指す

```
unsigned char   I_1[N][N], I_2[N][N];
unsigned short  P[N][W][D];
asm volatile("pxor %%mm7,%%mm7"::);
for (d=0; d<D*D; d+=4) {
  l = d/D;
  k = d%D;
  asm volatile("movd %0,%%mm0"::"m"(I_1[x][y]));
  asm volatile("movd %0,%%mm1"::"m"(I_2[x+k][y+l]));
  asm volatile("punpcklbw %%mm7,%%mm0"::);
  asm volatile("punpcklbw %%mm7,%%mm1"::);
  asm volatile("movq %%mm1,%%mm2"::);
  asm volatile("psubusw %%mm0,%%mm1"::);
  asm volatile("psubusw %%mm2,%%mm0"::);
  asm volatile("por %%mm0,%%mm1"::);
  asm volatile("movq %%mm1,%0":"=m"(P[x][y%W][d]));
  ...
}
```

Fig. 3: MMX implementation of Equation (8)

5 一貫性評価法に基づく信頼度評価

5.1 一貫性評価法 (Consistency Checking)

ここまで述べてきたのは，高速な相関演算を行う手法であるが，オプティカルフローを生成するためには相関演算の結果から適切な動きベクトルを検出する機構をオプティカルフロー生成システムに組み込む必要がある．

相関演算による対応点の探索のための相関度の計算は，最も似ている領域の検出がおこなわれるだけで，オクリュージョンのように実際には対応点が存在しなかったり，輝度が均一な平面で対応が原理的に計算できない場合がある．実際に単純な相関演算の結果から得られる

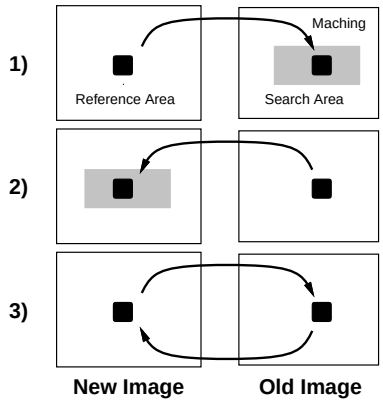


Fig. 4: Consistency checking between two images

オプティカルフローは，非常に信頼性がない．

この問題を解決するために様々な研究が行われているが，本論文では対応点探索における相関演算の信頼度評価に，視差画像を求める手法として開発された”Left-to-Right Right-to-Left Consistency Checking(LR-check)” [7,8] を応用する．

これは一つの画像の点に対応する反対側の画像上の点からの対応点が，最初の点に一致する場合のみ，その対応が信頼できるとするものである (図 4)．この方法により，対応点がとれない場所がはっきり区別出来るようになる．アルゴリズムは以下の通りである．以降，このアルゴリズムを一貫性評価法と呼ぶ．

1. 古い画像の局所領域を参照領域 (領域 1) とし，新しい画像を探索し最も相関の高い局所領域を領域 1 に対応する領域 (領域 2) とする．
2. 領域 2 を参照領域とし，古い画像を探索し最も相関の高い局所領域を領域 2 に対応する領域 (領域 3) とする．
3. 領域 1 と領域 3 が同じ局所領域であれば，領域 1 と領域 2 は二枚の連続する画像中の対応する領域とする．

5.2 一貫性評価法の再帰相関演算への組み込み

一貫性評価法はキャッシュを意識した高速アルゴリズムのループ中に組み込むことにより，メモリ使用量，計算時間を増やすことなく信頼のあるオプティカルフローを高速に計算できるようになる．

まず二枚の画像の信頼度の計算を行う際には，対応点探索のための相関度の計算を古いものから新しいもの，新しいものから古いもの双方で行う必要はなく，相関演算は一度行うのみで，対応点探索だけを両側で行えばよい．

従って一貫性評価法は 図 2(c) の x のループの計算中に行うことができ，計算時間は数 % から十数 % 程度増加する程度で行える．

Table 1: Algorithmic optimization of optical flow

Cost Function	PentiumMMX 233MHz	PentiumII 450MHz
a)	3.394	2.243
b)	0.138	0.050
c)	0.111	0.047
d)	0.052	0.019

(sec) (N=64, W=16, D=8)

a) Correlation, b) Recursive correlation, c) Cache optimal correlation, d) MMX implementation

6 実験

全ての実験を比較のために PentiumMMX-233MHz(外部バス 66MHz) と PentiumII-450MHz(外部バス 100MHz) を用いて linux 2.0.35 上で行った．C 言語は全て gcc-2.7.2.3 を用いて，また MMX は gcc 上でインラインアセンブラを用いて実現された．実験は全て同一のバイナリを用いて行った．

6.1 アルゴリズムの高速化の性能評価

式 (2) の相関演算と LR-Check を用いてオプティカルフローを生成する問題に対して，図 2 で述べたように，a) 普通に，b) 再帰的高速相関演算手法によるアルゴリズム，c) キャッシュを意識したメモリ利用効率化アルゴリズム，d) MMX 命令を用いた高速化，それぞれのバージョンのソフトウエアについて計算時間を計測したものを表 1に示す．

6.2 一貫性評価法の性能評価

第 5 節で示した一貫性評価法の性能評価を行った．まず，一貫性評価法を組み込むことによる，計算時間のロスを計測した．図 2 の c) キャッシュを意識したメモリ利用効率化アルゴリズム，d) MMX 命令を用いた高速化，それぞれのバージョンについて，一貫性評価を組み込まない場合，組み込んだ場合の計算時間を計測し，表 2に示す．

6.3 ロボット搭載用システムの開発

前節まで述べてきたアルゴリズムを実際にロボットに搭載する際の，画像取り込み装置は Bt848 という画像取り込み IC を用いたキャプチャーカードを用いた．このハードウエアはリアルタイムで画像をメモリに DMA 転送する機能を持ち，Linux 用のデバイスドライバが公開されていることから，容易に実時間画像処理システムを構築することができる，という特徴を持っている．

Table 2: Time consumption by Consistency Checking(CC)

Cost Function	PentiumMMX 233MHz		
	with CC	without CC	loss
c)	0.111	0.102	8.8%
d)	0.052	0.049	6.4%

Cost Function	PentiumII 450MHz		
	with CC	without CC	loss
c)	0.047	0.043	0.9%
d)	0.019	0.015	26.6%

(sec) (N=64, W=16, D=8)

a) Correlation, b) Recursive correlation, c) Cache optimal correlation, d) MMX implementation

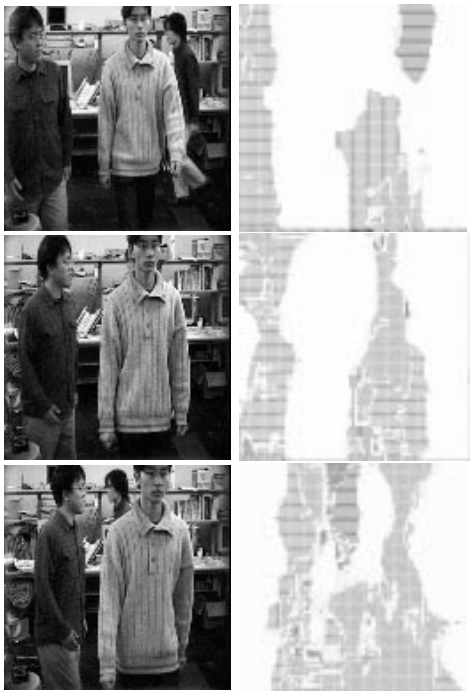


Fig. 5: Experiment : Real-time depth optical flow generation

6.4 オプティカルフロー生成実験

上記のシステムを用いて，実環境中のシーンにおけるオプティカルフローの生成実験を行った様子を図5に示す．左列が入力画像，右列が出力画像である． PentiumII-450MHz を用いて， N を 128 としている．

7 おわりに

本論文では，画像処理の分野で行われている再帰的相関演算手法を拡張し，連続する二枚の画像で高速な対応点探索を行うアルゴリズムを， Pentium プロセッサのマルチメディア命令（MMX 命令）を用いて最適化し，市販の PC 部品のみを組み合わせて実時間でオプティカルフローを生成することが可能であることを示した．

また，本稿における相関演算は 式 (2) の SAD を採用したが，我々は基礎実験を通じて実環境で活動するロボットでは式 (1) の正規化相関演算がより適していることを確認している． MMX 命令では命令セットとデータ長の制約から正規化相関を実装できないが，本年度に出現する PentiumIII に搭載された 次期マルチメディア命令 KNI(MMX2) ではこの問題もクリアされるため，式 (1) を実時間で計算できると予想される．

さらに，本稿で示した手法により得られるオプティカルフローと，すでに開発した実時間距離画像システム [10] で得られる距離情報と統合することで，我々が 3D Flow [11] と呼ぶ物体の三次元的な動きを直接，実時間で獲得することができる．今後は 3D Flow を用いた実環境認識とロボットの行動生成法に取り組む予定である．

参考文献

1) 稲葉雅幸. 局所相関プロセッサを用いたロボットビジョン. 日本ロボット学会学会誌, Vol. 13, No. 3, pp. 327–330, 1995.

2) 森田俊彦, 沢崎直之, 内山隆, 佐藤雅彦. カラートラッキングビジョン. 第 14 回日本ロボット学会学術講演会予稿集, pp. 279–280, 1996.

3) A. Verri and T. Poggio. Motion Field and Optical Flow: Qualitative Properties. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 5, pp. 490–498, 1989.

4) B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, Vol. 17, pp. 185–204, 1981.

5) O. Faugeras, B. Hots, H. Mathieu, T. Viéville, Z. Zhang, P. Fua, E. Théron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, and C. Proy. Real time correlation-based stereo: algorithm, implementations and applications. Technical Report N°2013, INRIA, 1993.

6) T. Kanade. Development of a Video Rate Stereo Machine. In *Proc. of the 1994 ARPA Image Understanding Workshop*, pp. 14–16, Nov 1994.

7) R. Bolles and J. Woodfill. Spatiotemporal Consistency Checking of Passive Renge Data. In T. Kanade and R. Paul, editors, *Robotics Research: The Sixth International Symposium*, pp. 165–183. International Foundation for Robotics Research, 1993.

8) P. Fua. A Parallel Stereo Algorithm that Produces Dense Depth Maps And Preserves Images Features. In *Machine Vision and Applications*, pp. 35–49, 1991.

9) 松本吉央, 坂井克弘, 稲邑哲也, 稲葉雅幸, 井上博允. PC ベースのハイパーマシン. 第 15 回日本ロボット学会学術講演会予稿集, pp. 979–980, 1997.

10) 加賀美聡, 岡田慧, 稲葉雅幸, 井上博允. ロボット搭載用実時間視差画像生成システム. 第 4 回ロボティクスシンポジア予稿集, pp. –, 1999.

11) Satoshi Kagami, Kei Okada, Masayuki Inaba, and Hirochika Inoue. A Fast 3D Optical Flow Generation System. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. –, 1999.