論　文

ビジョンチップのためのマルチターゲットトラッキングとその応用

渡辺　義浩[†]　小室　孝[†]　鏡　慎吾[†]　石川　正俊[†]

Multi Target Tracking Algorithm for a Vision Chip and Its Applications

Yoshihiro WATANABE[†], Takashi KOMURO[†], Shingo KAGAMI[†], and Masatoshi ISHIKAWA[†]

あらまし　本論文では，ビジョンチップのための複数対象同時追跡を行うマルチターゲットトラッキングのアルゴリズムを提案する．本アルゴリズムは 2 分探索を用いたラベリング処理と Self Window 法によるトラッキング処理からなり，ビデオフレームレートで代表される従来の視覚システムにおけるものに比べて，高速性の点で優れている．更に，本アルゴリズムのアプリケーションとして，回転物体の回転軸，速度を計測する回転計測と画像上に出現する領域数を計測する個数カウントを提案する．提案するアプリケーションに関してビジョンチップによる評価を行い，100〜1000 Hz といった高いフレームレートでの実現が可能であることを示す．

キーワード　ビジョンチップ，リアルタイム画像処理，トラッキング，回転計測，個数カウント

1. まえがき

視覚が認識に与える効果は，その獲得できる情報量の多さから，他の感覚に比べ，大きいと考えられる．しかし，視覚情報を用いた認識システムは，視覚装置のフレームレートの低さとその視覚情報の処理の膨大さにより，対応できる実時間性に制約を受けることが多かった．

本論文で着目する対象追跡処理に関しても，従来のシステムではフレームレートの低さにより，対象の動きが高速である場合，追跡が困難であった．更に，対象が複数になると，低速である場合でも，処理時間が制約となり，その実現性はより低いものとなっていた．しかし，任意の動きをする複数対象の実時間追跡処理は様々な画像認識に適用できる可能性をもっており，有用性が高いと考えられる．

従来のシステムの問題は，主に視覚装置の構造に起因している．高い実時間性が要求される視覚処理において，視覚装置が画像情報のすべてを出力するのは冗長であり，必要とされる特徴量のみを出力する機構を有するべきであると考えられる．この点で，ビジョンチップと呼ばれるデバイスは高い可能性をもっている．ビジョンチップとは，センサと並列演算装置を一体化し，1 チップ上に収めたデバイスであり，高いフレームレートを実現することができる．

本論文は，複数対象の実時間同時追跡処理を行うマルチターゲットトラッキングのアルゴリズムを提案する．本アルゴリズムは，小室らの開発した汎用ビジョンチップ [1] を用い，同チップの有する高フレームレート性を利用した Self Window 法 [2] によるトラッキング処理に加え，総和演算機能を利用した 2 分探索ラベリング処理を新たに導入することで高速なマルチターゲットトラッキングを実現するものである．

更に，高速な実時間処理にも対応し得るマルチターゲットトラッキング応用の発展性を示すため，マルチターゲットトラッキングを利用した回転計測と個数カウントのアプリケーションを示す．

2. 本研究の背景

本章では，本研究で用いるビジョンチップの構成，及び従来の対象追跡に関する研究について述べる．

2.1 ビジョンチップ

図 1 に，ビジョンチップのモデルを示す．従来の視覚システムではセンサと処理装置が分離されていたのに対し，ビジョンチップは自らが処理能力をもったセンサとなっている．ビジョンチップは従来のシステムに比べ，小型，軽量，低消費電力であるだけでなく，

[†] 東京大学情報理工学系研究科，東京都
School of Information Science and Technology, The University of Tokyo, Tokyo, 113–8656 Japan

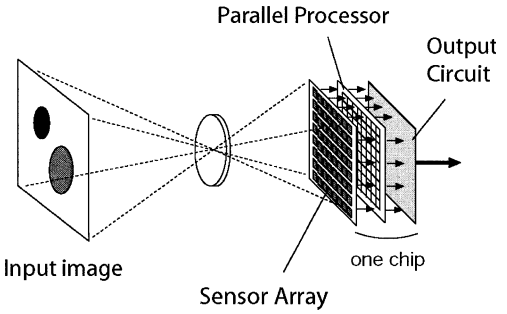

図 1　ビジョンチップのモデル
Fig. 1　Model of vision chip.

センサと処理装置の一体化によりセンサから画像情報を走査し，処理装置に伝送する必要がなく，高い実時間性をもった視覚処理を行うことができる．また，センサが後段に渡す情報は，センサ内の処理によって，その情報量を縮約できる．以上の点で，ビジョンチップは，視覚情報処理システムの入口となるセンサとして望ましい性質を有している．また，実装するアルゴリズムに関しても，特定の分野の画像処理は，高速性が伴うことによりフレーム間の画像の変化が少なくなるため，通常よりもシンプルな形でアルゴリズムを記述することができる．

本研究においてビジョンチップは，小室らによって開発されたディジタルビジョンチップを用いている [1]．以下，本論文において用いられているビジョンチップとはこれを指すものとする．

本ビジョンチップは，画像の特徴抽出・認識を行う前の処理として用いられるエッジ検出や平滑化，細線化などの初期視覚処理と呼ばれるもののほとんどを，μs のオーダで処理することができる．

更に，画像からの特徴量抽出に関しても，画面全体の総和，全 OR などを μs のオーダで演算することができる [3]．また，対象追跡のアルゴリズムとして動画像間の領域の対応付けを高速に行う Self Window 法が提案されている [2]．

2.2　従来の対象追跡に関する研究

視覚情報を用いた対象追跡は，対象が高速かつ複雑な動作をした場合，従来のシステムでは実時間で正確に追跡をすることが困難であった．これは，用いられる視覚装置のフレームレートが低いことに主な原因がある．フレームレートが低いと，追跡のためのアルゴリズムも確率に基づく予測をする必要があり，複雑な動きには適用できない．視覚を用いた対象追跡は，光学式モーションキャプチャなどにおいて用いられているが，従来のシステムでは，計測対象に反射性のマーカを付け，複数のマーカの動きから対象全体の動きを再構成していた．このシステムでは視覚装置のフレームレートは高いものを用いているが，視覚装置の制限により，いったん画像を蓄積してから処理するものしかなく，実時間追跡処理を行うものは存在しない．また，発光する赤外線マーカを用い，位置センサによって高レートで計測を行う例もあるが，この場合，発光を制御するためにマーカをコードでつなぐ必要があり，これによる動作の制限は大きいと考えられる．

一方，撮像系と処理系を 1 チップ化した，対象追跡のための特定用途センサ開発の例として，Brajovic らの Tracking Computational Sensor [4] が挙げられる．彼らの開発したセンサは，フレームレートに関しては，従来の視覚装置に比べて格段に高速化されているが，対象を一つしか追跡することができない．複数の対象を追跡するセンサも開発されているが [5]，個々の対象の判別を明るさで行っているだけでなく，重心計算が領域ごとに行われていないため，正確な追跡が実現できていない問題がある．

3.　マルチターゲットトラッキング

従来，ビジョンチップを用いた実時間追跡処理は，対象が単一である場合に関しては Self Window 法により，高いフレームレートで実現されている．一方，対象が複数である場合には，Self Window 法の初期処理としてラベリングが必要になる．しかし，ラベリング処理に関して，ビジョンチップへの実装面，高速性の両面で十分な要求を満たすものが提案されていないため，複数対象の追跡処理は実現されていなかった．

そこで，本論文では 2 分探索を利用したラベリング処理を新たに導入し，複数の対象を同時に追跡するためのマルチターゲットトラッキングアルゴリズムを提案する．前提として，アルゴリズムへの入力は 2 値画像とする．

本アルゴリズムは，分割領域に対するラベリング処理と各対象領域のトラッキング処理の 2 段階に分けられる．本アルゴリズムにおけるラベリング処理はトラッキング対象を生成する初期処理であり，トラッキング処理は以降のフレームにおいて領域分割/対応付けを行う処理である．つまり，ラベリング処理は画面内の対象数に変動があるなど新たにトラッキング対象

を生成するフレームにおいてのみ適用され，トラッキング処理は画面内にトラッキング対象がある限り毎フレーム行われる．またフレーム間の画像のずれが小さいという Self Window 法の前提となる高フレームレート性のためには，ラベリング処理，トラッキング処理のいずれも高速性が要求される．

それぞれの処理に関して，詳細を以下で述べる．

3.1 2 分探索ラベリング

高速なラベリング処理の実現は，フレームレート向上の点で有用であると考えられるが，従来は計算量が大きく，高速性が低かった．また，従来のアルゴリズムは画面を走査してラベリングを行う点で，ビジョンチップへの実装は適していなかった．並列処理化をすることで，高速な処理を実現した例もあるが [6]，探索を 1 行（列）ずつ行っているなど，効率化の余地があると考えられる．

本論文で提案する手法は，ビジョンチップのもつ画素並列性と画面全体の高速な総和演算機能を利用し，2 分探索によって高速にラベリングを行うものである．

図 2 にこの 2 分探索ラベリングアルゴリズムの流れを示す．本アルゴリズムは，2 分探索によって領域に含まれる 1 点を取得し，この点をもとに領域を復元することで，ラベリングが行われる．具体的なアルゴリズムを以下に示す．

1. 2 分探索による起点の取得 $g_k^i = S(I_k)$
   –時刻 k の画像 I_k を行に関して 2 分割していき，領域の存在する任意の 1 行を探索する
   –取得した行を更に列に関して，同様に 2 分探索することで領域 i の 1 点 g_k^i を取得する
   –取得した点を領域の起点と呼ぶ
2. ラベリング対象領域復元
$$\begin{cases} {}^0C_k^i = g_k^i \\ {}^{j+1}C_k^i = D_1({}^jC_k^i) \cap I_k \quad (j = 1, 2, \ldots, n_l) \end{cases}$$
   –元画像 I_k と起点 g_k^i から，1 画素膨張 D_1 と論理積を n_l 回繰り返し，起点を含む領域 i の画像 C_k^i を復元する
3. 元画像更新 $I'_k = I_k \cap \overline{C_k^i}$
   –抽出した領域を削除したものを元画像として更新する
4. 繰返し
   –1〜3 を M 回繰り返す
   （M は，ラベリングの必要がある領域の個数である）

2 分探索の部分において，探索範囲に領域が存在するかどうかに関しては，探索範囲の画像と元画像との論理積画像の面積を参照する．この面積が 0 でなければ，探索範囲に領域が存在するといえる．参照する面積に関しては，ビジョンチップの総和演算機能が利用できる．領域復元のための繰返し回数 n_l は，対象の大きさによって決定される．例えば，半径 5 の円に関しては領域が復元されるまでに 10 回の演算を要する．複数の領域を復元する場合は，最も大きな領域に対する値を適用する．また，全体の繰返し回数 M に関しては，アプリケーションに応じて決定されるものとする．ラベリング処理の前処理として，面積などの特徴に基づいて特定の領域だけを残すことにより，必要な領域のみを優先的にラベリングすることも可能である．

起点探索を 1 画素ずつ若しくは 1 行/列ずつ行うのではなく，2 分探索を用いることで探索回数は大幅に短縮することができる．

画素数を $N \times N$，対象数が M，対象の大きさを L とした場合の，ラベリング処理全体の計算量を，従来のラスタ走査を用いた手法と比較する．従来手法の計

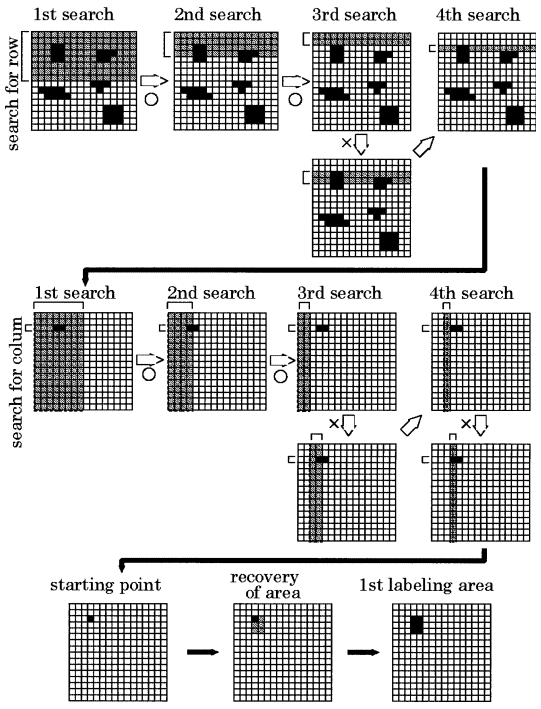

図 2 ビジョンチップによるラベリングアルゴリズム
Fig. 2 Labeling algorithm using vision chip.

算量のオーダは，走査回数を対象の大きさに依存する $f(L)$ で表すと，$O(N^2 f(L))$ となる．一方，2 分探索を用いたラベリング処理の場合は，起点取得のための探索回数は $\log_2 N$ に，領域復元は L にそれぞれ比例するとすると，$O(M(\log_2 N + L))$ となる．以上より，対象数や大きさが小さい範囲ならば，画素数が大きくなった場合に，2 分探索を用いたラベリングは従来手法に比べて，高速性の点で有効である．

3.2 トラッキング処理

図 3 にトラッキング処理の概要を示す．

上記のラベリング処理によって，複数のトラッキング対象が生成された．以降のフレームでは，この各領域に対して，以下で述べる Self Window 法を順番に適用することで，実時間処理によるマルチターゲットトラッキングが実現できる．

提案されている Self Window 法は，ビジョンチップの高フレームレート性を利用し，前フレームにおけるトラッキング対象画像が既知な限り，フレーム内でのラベリングとフレーム間の対応付けを統一化できるアルゴリズムである．具体的には，フレームレートが高速ならばフレーム間の画像の動きが 1 画素以内であるという仮定を利用することで，処理を論理積と dilation のみで記述したアルゴリズムである．また本アルゴリズムは，ビジョンチップの並列処理性により，高速に実行可能である．

本論文では，前述よりも緩やかな，フレームレートが高速ならばフレーム間で同一領域は重なっているという仮定を置き，より高速な対象に対してもトラッキングが行える形にする．この場合も，Self Window 法を数回繰り返すことにより，同様の効果を得ることができる．本仮定は，図 3 に示すようにラベリング処理を行うフレームにおいても成り立つ必要がある．本仮定は，ラベリング処理とトラッキング処理の高速性から常に成り立つものとする．具体的なアルゴリズムを以下に示す．

1. トラッキング対象領域復元

$$\begin{cases} {}^0C_k^i = C_{k-1}^i \\ {}^{j+1}C_k^i = D_1({}^jC_k^i) \cap I_k \quad (j = 1, 2, \ldots, n_t) \end{cases}$$

–時刻 k の入力画像と時刻 $k-1$ のトラッキング対象 i の画像 C_{k-1}^i から，1 画素膨張 D_1 と論理積を n_t 回繰り返し，時刻 k のトラッキング対象画像 C_k^i を復元する

–ただし，$n_t \leq n_l$

2. 繰返し

–1 を複数のトラッキング対象画像に対して，順番に行う

領域復元の繰返し回数 n_t は，少なくともラベリング処理で適用される n_l 回であれば十分である．通常は，処理時間削減のため，アプリケーションに応じて推定されるフレーム間の動きから決定する．トラッキング処理の信頼性に関しては，領域の面積や対象の軌跡を利用して評価することができる．例えば，対象同士の単純な衝突，分離に関しては，フレーム間の領域の面積変化や軌跡から検出可能である．また，上記の仮定が成立しなくなり，異なる対象を追跡した場合でも，同様に検出可能である．

3.3 アルゴリズム実行時間の評価

提案するマルチターゲットトラッキングアルゴリズムについて，実行時間を評価する．

ラベリング処理における起点を取得するまでの 2 分探索の実行時間 T_l^1 は，1 回の探索処理に要する時間を a，探索範囲の修正に要する時間を b，修正を要する回数を h とすると，$N \times N$ の画素数では，$2a \log_2 N + bh$ となる．ここで，探索範囲の修正とは，探索範囲に領域が存在しなかった場合に適用するものであり，$0 \leq h \leq 2 \log_2 N$ である．また，ラベリング処理における領域復元の実行時間 T_l^2 は，1 回の繰返し演算に要する時間を c とすると，前述の繰返し回数 n_l を用いて cn_l となる．よって，M 個の領域のラベリング処理実行時間は，$M(T_l^1 + T_l^2)$ となる．例えば，1 個の半径 4 の円領域をラベリングするためには，今回用いたビジョンチップでは，$a = 17\,\mu$s，$b = 4\,\mu$s，$c = 16\,\mu$s，$N = 64$ より，332〜380 μs を要する．

一方，トラッキング処理の実行時間は，領域数 M，

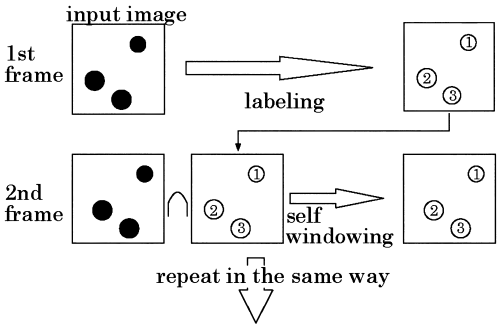

図 3 トラッキング処理
Fig. 3 Tracking process.

前述の繰返し回数 n_t，1 回の領域復元のための繰返し演算に要する時間 c を用いて，Mcn_t となる．例えば，繰返し回数が 5 回である場合は 1 ms のフレーム時間内に 12 個の対象追跡が可能である．

現行のビジョンチップは，ビジョンチップを制御するコントローラが最適化されていないという問題を含んでいるが，今後，コントローラ処理能力の向上により，各処理の実行時間は短くすることが可能である．また 2 分探索に関して，画面全体の全 OR を演算するための最適な回路を設けるなど，本処理に特化したビジョンチップの開発により，更なる高速化も期待できる．

4. 応 用 例

1 フレーム時間が 1〜10 ms といった，従来の視覚装置では不可能であった高フレームレートでのマルチターゲットトラッキングの導入により，アプリケーションの範囲は広がると考えられる．本章では回転計測とカウント処理の二つのアプリケーションを提案し，提案するマルチターゲットトラッキングの応用性が高いことを示す．

4.1 高速回転物体の運動計測

本節では，回転計測への応用を提案する．例えば，サッカーや野球のボールを計測するなど，高速に回転する対象の回転計測は有用性が高いと考えられる．

物体の運動状態を計測する手法は数多く存在するが，特に視覚情報を用いたものの一つにオプティカルフローによる動き検出がある．オプティカルフローとは，動画像情報から求められる，複数の点に対する画像上の移動ベクトルの分布のことである．複数の点に対する移動ベクトルを求めることができれば，対象の運動状態の推定が可能となる．しかし，従来のアルゴリズムで，オプティカルフローを高速に動く対象の運動計測に適用することは，計算量が多いため実時間性の点から困難である．

先に述べたマルチターゲットトラッキングを用いることで，高速な運動をする対象に対する対象に対しても実時間処理による運動計測が可能である．

4.1.1 計 測 手 法

今回，計測する対象は球に限定した．また，提案するマルチターゲットトラッキングアルゴリズムを適用するため，球には模様をつけた．以下，この前提のもとで，回転軸と回転速度の計測を行うものとする．

本手法は，次の 3 段階に分かれる．

（1） 軌跡検出 $\boldsymbol{P}^i = A(I_1, I_2, \ldots, I_n)(1 \leq i \leq m)$

n 枚の原画像群より，領域 i の画像上での重心の軌跡 $\boldsymbol{P}^i$ を求める．

（2） 3 次元速度ベクトル推定 $\boldsymbol{v}^i = B(\boldsymbol{P}^i)$

重心の軌跡より，領域 i の 3 次元速度ベクトルを推定する．

（3） 回転情報出力 $[\boldsymbol{l}, \omega] = C(\boldsymbol{v}^1, \boldsymbol{v}^2, \ldots, \boldsymbol{v}^m)$

m 個の領域の速度ベクトルから回転軸 $\boldsymbol{l}$ と回転速度 ω を計算する．

重心を用いることによって，サブピクセル単位で各領域の位置情報を得ることができる．

4.1.2 では，(1)〜(2) について述べる．**4.1.3** では，(3) について述べる．

計測するシステムは，ビジョンチップ，コントローラ，計算機の三つからなる．(1) をビジョンチップとコントローラ内で行い，(2)〜(3) を外部の計算機において行う．

4.1.2 速度ベクトルの取得方法

複数の対象を同時に追跡するマルチターゲットトラッキングによって，複数の領域の速度ベクトルを取得する．

上記（1）の具体的な手順を以下に示す．

（1-1） ラベリング $\boldsymbol{C}_1 = L(I_1)$

1 フレーム目の入力画像 I_1 に対して，ラベリング処理を施し，各領域の画像 $\boldsymbol{C}_1 = [C_1^1, C_1^2, \ldots, C_1^m]$ を得る．

（1-2） トラッキング $C_k^i = T(I_k, C_{k-1}^i)$ $(k \geq 2)$

時刻 k の入力画像 I_k と時刻 $k-1$ における領域 i の画像 C_{k-1}^i から，Self Window 法により時刻 k における領域 i の画像 C_k^i を得る．

（1-3） モーメント抽出 $\boldsymbol{M}_k^i = M(C_k^i)$

時刻 k における領域 i の画像 C_k^i より，0 次，1 次モーメント $\boldsymbol{M}_k^i$ を抽出する．

（1-4） 重心位置計算 $\boldsymbol{P}_k^i = E(\boldsymbol{M}_k^i)$

モーメント情報 $\boldsymbol{M}_k^i$ より，領域の重心位置 $\boldsymbol{P}_k^i$ を計算する．

以上の処理により，各領域の重心の軌跡 $\boldsymbol{P}^i = [\boldsymbol{P}_1^i, \boldsymbol{P}_2^i, \ldots, \boldsymbol{P}_n^i]$ が得られる．

更に，上記（2）の具体的な手順を以下に示す．

（2-1） 3 次元位置ベクトル計算 $\boldsymbol{r}_k^i = F(\boldsymbol{P}_k^i)$

画面上の重心位置より，3 次元位置ベクトルを計算する．

（2-2） 3 次元速度ベクトル計算 $\boldsymbol{v}^i = \bar{\delta}(\boldsymbol{r}_n^i - \boldsymbol{r}_1^i)/|\boldsymbol{r}_n^i - \boldsymbol{r}_1^i|$

3 次元速度ベクトルを，こう配は軌跡の始点と終点より，ノルムは各フレームの移動量の平均 $\bar{\delta}$ より求める．

以上の処理によって，n 枚の画像から m 個の各領域の 3 次元速度ベクトルが得られる．

4.1.3 回転情報計算方法

複数の速度ベクトルより，回転軸，回転速度を計算する．以下に，具体的な計算方法を示す．

簡単のため，球の中心を原点とし，球は画面の中心に撮像されているものとする．また，カメラの撮像面は xy 平面に平行であり，z 軸正方向から球を撮像するものとする．

回転軸 $\boldsymbol{l}$ は，各領域の速度ベクトル $\boldsymbol{v}$，位置ベクトル $\boldsymbol{r}$，画面上での球の半径 R より，式 (1) から，最小 2 乗法を用いて求められる．ただし，本計測の場合，誤差が直線をプロットする 2 次元平面において非一様的に分布するため，最小 2 乗法によって得られる解は必ずしも最適であるとはいえない．最適な解法は存在するが [7]，計算量が大きく，実時間処理には不適であるため，本実験では採用しない．本実験では，実時間性の面から最小 2 乗法を採用する．

$$l_x v_x + l_y v_y - \frac{r_x v_x + r_y v_y}{\sqrt{R^2 - r_x^2 - r_y^2}} = 0 \qquad (1)$$

回転速度 ω は，点 $\boldsymbol{r_0}$ における速度ベクトルを $\boldsymbol{v_0}$，また点 $\boldsymbol{r_0}$ を通り，回転軸 $\boldsymbol{l}$ に垂直な平面と原点との距離 h とすると，式 (2) より求められる．

$$\omega = \frac{|\boldsymbol{v_0}|}{\sqrt{R^2 - h^2}} \qquad (2)$$

4.1.4 ビジョンチップによる評価

前述手法をビジョンチップに実装し，実験を行った．

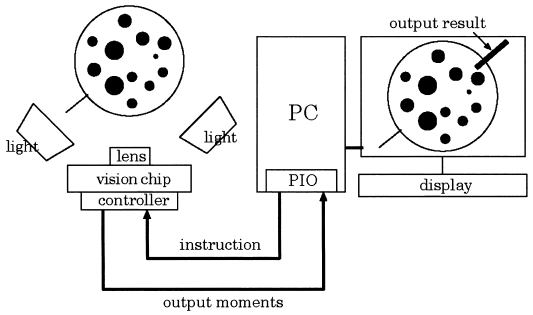

図 4 回転計測実験環境
Fig. 4 Experimental setup for rotation measurement.

実験環境の模式図を図 4 に示す．本実験では，捕そく領域数を 5 点，軌跡の点数を 5 回とした．それぞれの数は **4.1.2** における m，n に相当する．これにより，初期処理としてラベリング処理を行うフレーム時間が 3.3 ms，トラッキング処理を行うフレーム時間は 2.5 ms，回転情報を取得するまでに要する時間は 16 ms となった．

図 5 は軸の計測結果を表示したものである．このときの回転速度は 250 rpm である．白く表示されている線は，計算結果から描画したものであり，手前側に突き抜けている軸である．逆側に薄く表示されているのが，球を支えている軸が映っている像である．これより，結果がおよそ正しい値を示していることがわかる．

図 6 に，回転軸を $\theta \approx 0°$，$\gamma \approx 50°$ 度に固定して計測した結果を示す．黒色のヒストグラムが θ を，灰色のヒストグラムが γ を示している．θ は xy 平面における回転軸の傾き，γ は xy 平面と回転軸が成す角である．値は，多少ばらついているもののほぼ固定した軸の角度の辺りで集中している．実験環境の原因で固定軸を大まかな値にしか設定できないこと，軸が不安定であることを考慮にいれれば，回転軸はほぼ正確な値が計算できていることがわかる．

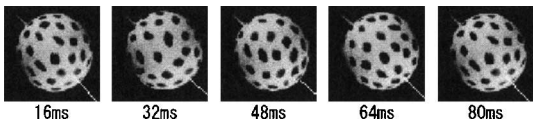

図 5 回転軸の計測結果画像
Fig. 5 Images of measured rotation axis.

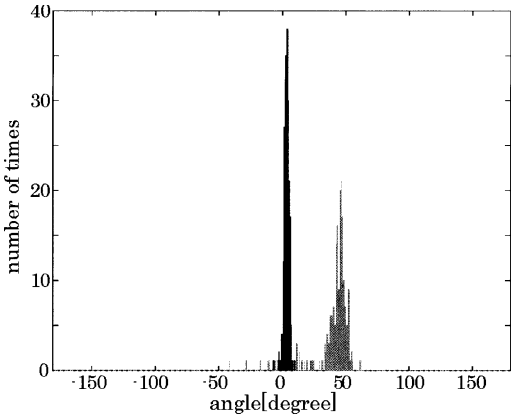

図 6 回転軸の分布
Fig. 6 Measured data distribution of rotation axis.

図 7 に，対象の回転速度を変えた場合の，回転軸の計測値の試行回数 200 回に対する平均値の変化を示す．これより，対象の回転速度に影響されず，安定した軸計測が行われていることがわかる．

図 8 に，回転速度の真の値と計測値を比べた結果を示す．×が測定値，○が真の値である．測定値は，真値と同様に直線的な増加を示しているが，傾きにわずかなずれが生じている．これは，入力画像から推定する球の半径の誤差が原因と考えられるが，回転速度に依存しない定数の誤差であるので，補正が可能である．

本装置は，対象の回転速度が 550 rpm 付近でその測定限界を迎える．これは，フレーム間で同一領域は重なっているという前述の Self Window 法を用いる際の仮定が成立しなくなるためである．しかし，ビジョンチップを制御するコントローラの処理能力が向上すれば，限界値を更に上げることができる．これは，現状ではフレーム時間に対する処理時間の占める割合が大きいため，処理能力の向上がフレームレートの拡大に大きく影響すると考えられるからである．

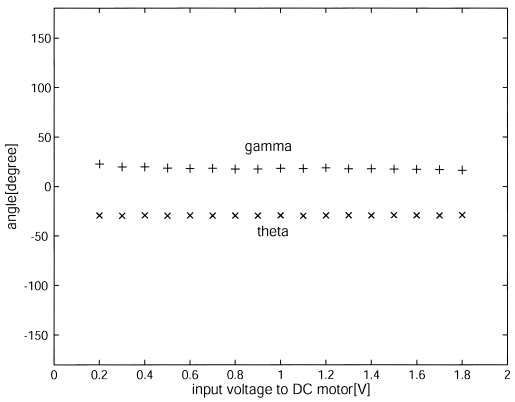

図 7　回転軸計測値の平均値
Fig. 7　Average of measured rotation axis.

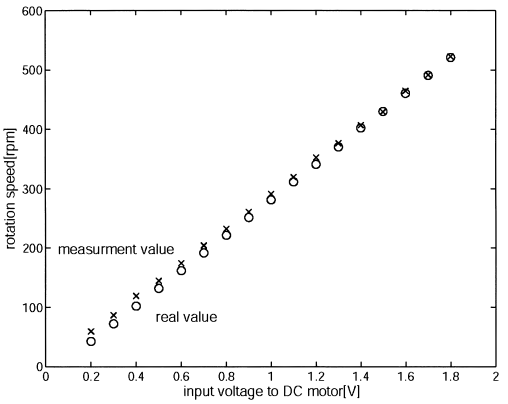

図 8　回転速度の計測結果
Fig. 8　Measured data of rotation speed.

得られた結果から，例えば球技における計測を考えた場合，回転数が 300〜600 rpm であるサッカーのフリーキックのボールに対しては，本システムによる計測が可能である．また，回転数が約 1800 rpm である野球のボールに対しては，現行のビジョンチップでは計測のための能力が足りないが，上記のようにコントローラ処理能力の向上により，計測は可能であると考えられる．

4.2　個数カウント

マルチターゲットトラッキングを用いることで，画像上に出現する分割領域の個数をカウントするアプリケーションの実現も可能である．更に，ビジョンチップはプログラムで処理を変えられるため，形や面積などを指定する条件を加えた個数カウントも可能である．

4.2.1　カウント手法

本手法は，マルチターゲットトラッキングを連続的に行い，新たな領域が出現した場合にのみ，個数を更新することで，実時間処理による領域の個数カウントを実現する．

入力画像中の領域群は，既にトラッキングをしている領域群と，新たに出現した領域群の 2 種類に分けられる．そのため新たに出現した領域群は，入力画像から捕そく済みの領域群を取り除くことで得られる．新たに出現した領域群の面積が 0 になるまで，ラベリング処理と個数の更新を行う．

本計測では，対象をただカウントするのではなく，同時にトラッキングも行っている．トラッキングを行うことにより，カウント済みの対象を再びカウントしてしまう誤処理を防止することができる．また，特定の形，面積のみの対象をカウントするなどの処理を加えることが可能である．

4.2.2　ビジョンチップによる評価

上記手法を実装し，実験を行った．実験環境を図 9 に示す．本実験では，五つの対象を回転ドラム上に設置して計測を行う．このため，計測結果は周期的に増加すると考えられる．対象の大きさは進行方向に約 1.3 cm (8 pixel) である．画像上に追跡を行っている対象が存在しない場合のフレーム時間は 1.2 ms であり，追跡対象が一つ増加するごとにフレーム時間は 0.2 ms 加算される．これは，トラッキング処理が各対象への

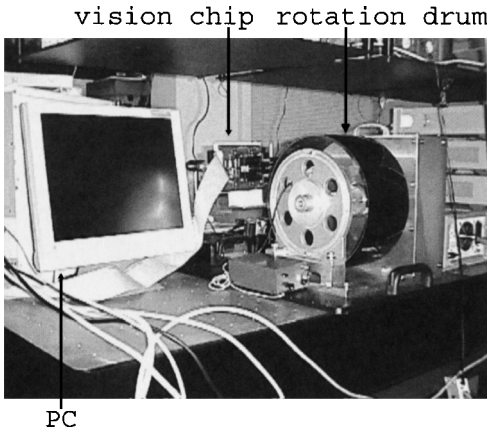

図 9　カウント処理実験環境
Fig. 9　Experimental setup for counting.

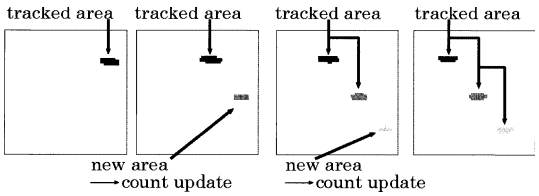

図 10　カウント処理計測画像
Fig. 10　Images of count process.

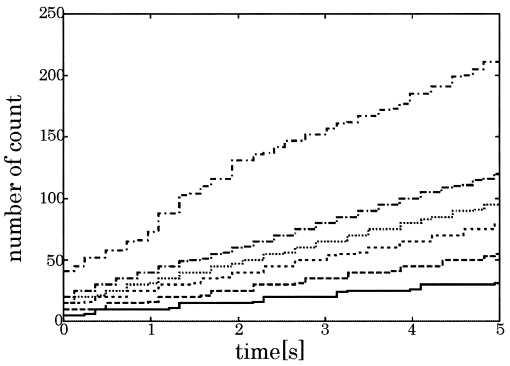

図 11　カウント処理実験結果
Fig. 11　Result of count process.

Self Window 法の逐次適用なので，処理時間が対象数に比例するためである．

図 10 にカウント処理時の画像列を示す．トラッキングしている領域をそれぞれ異なる灰色にて示している．画像から正しくトラッキングが行われていることが確認できる．

図 11 に実験結果を示す．周期的にカウントされていることから処理が正常に行われていることが確かめられる．こう配が急であるほど，高速な対象をカウントしていることを示している．ただし，グラフにおいて最も急こう配な対象に関しては，処理が失敗していることが確かめられる．これは，対象の動きが速く，Self Window 法の仮定が成立しなくなったためである．これより本環境における計測する対象の限界速度は約 4.5 m/s (2770 pixel/s) である．本処理も回転計測と同じく，コントローラの処理能力が向上すれば，限界値を上げることができると考えられる．

5. むすび

ビジョンチップを用いることにより，高速なマルチターゲットトラッキングが実現できることを示した．更に，トラッキングする対象が複数になることによって，その応用範囲が広がる可能性を示した．

今後は，マルチターゲットトラッキング用に特定用途化されたチップを開発し，より高速より複数の対象の実時間追跡処理を実現する予定である．

回転計測の今後の展開としては，自由に飛んでくるボールに対しても，リアルタイムで回転情報を計測することを目指す．また，球以外の形状の対象に対する計測を実現することを考えている．

また，個数カウントの今後の課題としては，マッチングなどの処理を加えることで，特定の対象のみをカウントする計測を行うことを考えている．

文　献

[1] 小室　孝，石川正俊，“PE 結合機能を持つ汎用デジタルビジョンチップの設計，” 信学技報，ICD2001-37 (SDM2001-114), 2001.
[2] 石井　抱，石川正俊，“高速ビジョンのための Self Windowing，” 信学論 (D-II)，vol.J82-D-II, no.12, pp.2280–2287, Dec. 1999.
[3] T. Komuro, S. Kagami, and M. Ishikawa, “A new architecture of programmable digital vision chip，” 2002 Symposium on VLSI circuits Proceedings, pp.266–269, 2002.
[4] V. Brajovic and T. Kanade, “Computational sensor for visual tracking with attention，” IEEE J. Solid-State Circuits, vol.33, no.8, pp.1199–1207, Aug. 1998.
[5] V. Brajovic, “An object tracking computational sensor，” Technical Report CMU-RI-TR-01-40, Robotics Institute, Carnegie Mellon University, Dec. 2001.
[6] 中野鉄平，彦本里美，森江　隆，永田　真，岩田　穆，“画像認識のための画素並列領域抽出アルゴリズムと FPGA への実装，” 信学技報，ICD2001-42, 2001.
[7] 金谷健一，空間データの数理，朝倉書店，1995.

（平成 14 年 12 月 4 日受付，15 年 3 月 24 日再受付）

渡辺　義浩

平 14 東大・工・計数卒．現在，同大大学院情報理工学系研究科システム情報学専攻修士課程在学中．実時間画像認識に興味をもつ．

小室　　孝

平 8 東大・工・計数卒．平 10 同大大学院修士課程了．平 13 同大学院博士課程了．現在，同大学院情報理工学系研究科システム情報学専攻助手．ビジョンチップ，並列プロセッサに関する研究に従事．博士（工学）．

鏡　　慎吾

平 10 東大・工・計数卒．平 12 同大大学院修士課程了．現在，同大学院工学系研究科計数工学専攻博士課程在学中．実時間センサ情報処理アーキテクチャ及びシステムの研究に従事．

石川　正俊

昭 52 東大・工・計数卒．昭 54 同大大学院修士課程了．同年通産省工業技術院製品科学研究所に入所．平元東大・工・計数助教授．現在同大大学院工学系研究科計数工学専攻教授．生体の情報処理機構の回路モデル，触覚センサの知能化，超並列・超高速ビジョン，光コンピューティング，センサフュージョン等に関する研究に従事．工博．