ラドン変換を用いたビジョンアルゴリズムによる平面運動物体の検出

坪井 辰彦

2004年

目 次

図 目 次

表 目 次

第1章　緒論

1.1　研究背景

視覚情報を用いる画像処理では，多くの計算量を処理する必要がある．コンピュータで画像処理を行う場合，コンピュータの処理能力と扱えるデータの容量が問題となる．そのため，コンピュータが開発されてまだ間もない頃は，画像処理専用のハードウェア，もしくはスーパーコンピュータのような高速な処理が可能で大容量のメモリを搭載したコンピュータでなければ画像処理を行うことができなかった．しかし，近年におけるコンピュータ技術の飛躍的な向上により，コンピュータの高速化と低価格化が進み，一般用のコンピュータが人々の間にも普及するようになった．一般のコンピュータでも高速な処理と大容量メモリの搭載が可能になったため，一般用のコンピュータを用いた画像処理が実行可能となり，以前と比べれば低コストで画像処理の実験や研究が行われるようになった．テキストデータや音声データと比べて，画像データの情報量は膨大であり，画像データを扱うアルゴリズムは計算量が多い．そのため，高速な処理が要求される場合では，コンピュータ上の画像処理では間に合わないという問題が生じている．そこで，画像処理の実行速度を上げるために，画像処理アルゴリズムを LSI 等のハードウェアに実装する手法が提案されている．

一口に画像処理といっても，記録や加工，伝送，計測，認識などの多くの種類がある．これら画像処理の分野の一つに，画像から対象の認識や位置と姿勢の検出を行うビジョンアルゴリズムの研究がある．人間は，視覚情報を脳が高度な処理を実行してくれるため，外界の状態を正確に認識することができる．コンピュータが，視覚情報である画像をもとに人間のような正確な認識を行うためには，画像処理を実行するためのアルゴリズムが必要である．そのため，画像からどのような特徴を取り出し，その特徴から認識を行うアルゴリズムの研究が行われている．このような画像処理を扱うアルゴリズムは，ビジョンアルゴリズムと呼ばれている．参照画像と入力画像を用いて画像から特徴を抽出し，マッチングを行う手法として，正規化相関法 [1, 18, 19]，一般化ハフ変換法 [2, 3]，フーリエ変換を用いた検出法 [4, 5, 6, 7, 8, 9]，ハフ・フーリエ変換法 [10]，フーリエ記述子法 [12, 13]，重心モーメント法 [14] が挙げられる．これらの検出法は，デジタル化された画像データをもとに変換や特徴抽出等の演算を実行する．また，ロバスト性の高い検出法は，複雑な演算を行うため，ロバスト性の高いものほど演算量が多くなる．

これらのビジョンアルゴリズムを実装する手段として，ソフトウェアによる実装とハードウェアによる実装がある．ソフトウェアによる実装では，アルゴリズムは C 言語等のプログラムで記述されるため，パーソナルコンピュータ等の汎用コンピュータで実行可能である．しかし，汎用コンピュータは，OS や別のプログラムの処理も実行するため，ビジョンアルゴリズムを実行中に割り込み処理が入り，ビジョンアルゴリズムの処理が遅れ

てしまう可能性がある．また，ソフトウェアでの実行は，基本的に逐次処理であり，ソフトウェアで並列処理を行うには，並列処理命令を持つ CPU が必要であり，その CPU を用いて局所的な並列処理が実行可能である．

一方，ハードウェアによる実装では，アルゴリズムは論理回路で記述され，そのアルゴリズム専用のハードウェアとして設計される．ハードウェアでの並列処理は，論理回路の記述で可能となるため，ハードウェアの制約を除けば，アルゴリズム内の並列処理が可能な部分は，すべて並列処理にすることが可能である．そのため，並列処理が可能なアルゴリズムでは，ソフトウェアによる実装よりも計算時間を短縮することができる．しかし，アルゴリズムを論理回路で記述することは，ソフトウェアでプログラムを記述することよりも難易度が高く，高度な並列処理ほど処理を実行するタイミングがシビアになるという問題がある．また，近年，LSI 技術の発展により，FPGA のような論理回路の再書き込みが可能な LSI が登場したことにより，ハードウェア実装のハードルは，年々低くなりつつある．しかし，それでもアルゴリズムを実装する難易度では，ソフトウェアによる実装よりもハードウェアによる実装のほうが，難易度が高い．したがって，ビジョンアルゴリズムの用途に応じて，アルゴリズムの実装を考慮する必要がある．

一方，ビジョンアルゴリズムを構築する段階では，アルゴリズムをソフトウェアで実行するほうが検証は容易である．また，検証が済みアルゴリズムが確定した後にハードウェアに実装したほうが効率がよい．

1.2 研究目的

本研究では，二次元の視覚情報を用いて平面運動物体の位置と姿勢を検出することを目的とする．はじめに複数の対象物の位置と姿勢を検出するアルゴリズムを提案し，次にビデオフレームレートで平面運動物体の位置と姿勢を検出するアルゴリズムを提案する．これらの提案するアルゴリズムには，ラドン変換を使用する．次に提案するアルゴリズムの検証は，パーソナルコンピュータを用いてソフトウェア上に行う．パーソナルコンピュータは，コンピュータ技術の発達により，画像処理の実行が可能となり，一般にも広く普及しているため，ビジョンシステムを安価に構築することが可能である．アルゴリズムの検証が済んだ後は，ビジョンアルゴリズムをハードウェアに実装し，ハードウェア上での実行性能を評価する．実装するハードウェアには，FPGA (Field Programmable Gate Array) を使用する．FPGA は，論理回路の再構成が可能な LSI であり，アルゴリズムを FPGA に実装した後でも，論理回路の書き換えが可能である．

1.3 本論文の構成

本論文は，全 5 章で構成されている．第 1 章では，研究背景と研究目的について述べる．第 2 章では，複数の対象物体を検出するビジョンアルゴリズムを提案する．第 2 章で提案するアルゴリズムでは，ラドン変換と一次元位相限定相関を用いており，提案するアルゴリズムをソフトウェアに実装し，物体検出の処理時間と精度について評価を行う．第 3 章では，第 2 章の考察をもとに，ソフトウェアによる実装でビデオフレームレートでの

検出が可能なアルゴリズムを提案する．まず，アルゴリズムの計算量を減らすため，検出する対象物体の数を一つとし，検出における画像環境と検出用途を限定する．次にラドン変換をもとに計算コストを減らした片側ラドン変換を定義し，片側ラドン変換を用いたビジョンアルゴリズムを提案する．さらに提案するアルゴリズムをソフトウェアに実装し，検出の処理時間と精度を評価する．第4章では，ビジョンアルゴリズムのハードウェア実装について考察する．アルゴリズムは論理回路として記述しなければならないため，ソフトウェア実装よりも難易度方が高く，計算量の多いアルゴリズムでは，多大な労力と時間を費やしてしまう．そこで第3章の考察をもとに，計算量の少ないビジョンアルゴリズムを提案する．そのアルゴリズムをハードウェアに実装し，検出実験を行う．この実験により，ハードウェア実装での処理時間や問題点を検討する．第5章では，これらの結論について述べる．

第2章　ラドン変換と一次元位相限定相関を用いた複数物体の平面運動検出

2.1　緒言

画像認識において対象物体の検出を行う場合，画像内の対象物体は一つとは限らない．例えば，乱雑に置かれた対象物体の仕分け作業に画像認識を用いる場合では，画像内に複数の対象物体が存在する．したがって，複数の対象物体を検出することは，画像認識にとって重要な要素の一つである．本章では，ラドン変換と一次元位相限定相関を用いて，複数の対象物体の位置と姿勢を検出するビジョンアルゴリズムを提案し，そのアルゴリズムをソフトウェアに実装したときの処理時間と検出精度について評価を行う．

画像認識においては，参照画像に一致する部分を入力画像内から検出することは，基本的な処理の一つである．このような検出の手法として，正規化相関法 [1]，一般化ハフ変換法 [2, 3]，フーリエ変換を用いた検出法 [4, 5, 6, 7, 8, 9]，ハフ・フーリエ変換法 [10] などが提案されている．

位置と姿勢の検出では，照明変動やオクルージョンに対するロバスト性が重要である．ロバスト性が高い位置の検出法として，位相限定相関法が提案されている [9]．位相限定相関法では，参照画像と入力画像の位相スペクトルのみを用いて位置を検出し，振幅スペクトルは用いない．照明変動やオクルージョンにより，振幅スペクトルは大きく変動するが，位相スペクトルの変動は比較的小さい．結果として，位相限定相関法における位置検出は，ロバスト性が高い．一方，姿勢検出法においては，振幅スペクトルを用いる手法 [4, 5, 6, 7, 8] が一般的であり，位相スペクトルのみを用いて位置と姿勢の両方を検出する手法は提案されていない．

本章では，ラドン変換と一次元位相限定相関を用いたビジョンアルゴリズムを提案する．従来より，画像の二次元パワースペクトルの極座標変換を用いて，姿勢を検出する方法が知られている．一方，フーリエ変換の中央断面定理により，画像の二次元フーリエ変換の極座標変換は，画像のラドン変換の一次元フーリエ変換に一致する [10]．したがって，ラドン変換の一次元フーリエ変換を用いることにより，姿勢の検出が可能になると考えられる．また，近年，画像処理アルゴリズムをハードウェアに実装することにより，計算時間を短縮する試みが成されている．画像の二次元フーリエ変換は，画像の取り込みと並行して実行することができない．一方，画像のラドン変換は，画像の取り込みと並行して求めることができる．さらに，ラドン変換をもとに検出を行う場合は，取り込んだ画像を保存するメモリが不要になる．したがって，ラドン変換をもとに検出を行う手法は，ハードウェアの実装の点で有効と考えられる．さらに，本章では，姿勢の検出において，一次元位相限定相関を用いたマッチングを使用することにより，振幅スペクトルを使用しない手

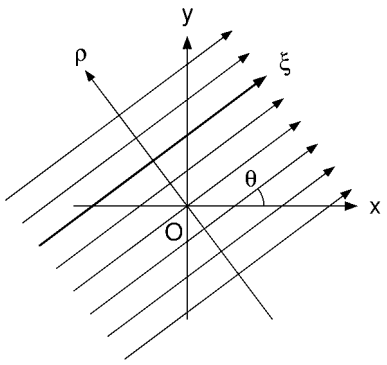


Fig. 2.1: Integral paths in Radon transform

法を提案する．姿勢の検出では，ラドン変換に一次元位相限定相関を用いたマッチングを行うことにより，複数の姿勢を検出することが可能である．また，検出した複数の姿勢から，対応する位置を検出することも可能である．したがって，提案する手法により，複数の対象物体の位置と姿勢を求めることができる．

本章の構成を以下に示す．第 2.2 節では，ラドン変換の性質について述べる．第 2.3 節では，一次元位相限定相関について述べる．第 2.4 節では，物体の位置と姿勢を検出するアルゴリズムについて述べる．第 2.5 節では，単数物体の合成画像を用いて検出実験を行い，その評価について述べる．第 2.6 節では，複数物体の合成画像を用いて検出実験を行い，その評価について述べる．第 2.7 節では，実画像による検出実験を行い，その検出結果について述べる．

2.2　ラドン変換とその性質

本章は，ラドン変換 [15] を用いて位置と姿勢の検出を行う．連続画像 $g(x, y)$ に対するラドン変換 $R(\rho, \theta)$ は，

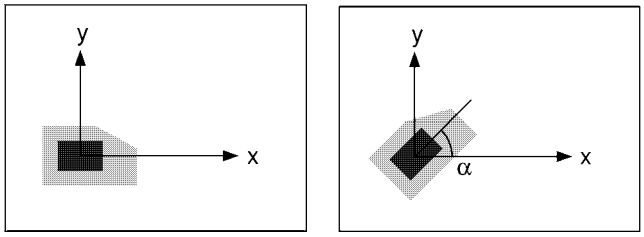
$$R(\rho,\theta)=\int_{-\infty}^{\infty} g(\xi C_\theta - \rho S_\theta, \xi S_\theta + \rho C_\theta)\,\mathrm{d}\xi \tag{2.1}$$

で与えられる [15]．ここで，$\cos\theta$ を C_θ，$\sin\theta$ を S_θ と略記する． ラドン変換 $R(\rho, \theta)$ は，Fig. 2.1-(a) に示すように，原点からの距離 ρ，x 軸からの角度が θ の直線に沿う画素値の線積分である．

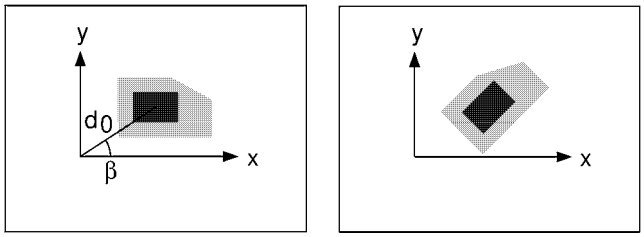
入力画像のラドン変換とパターン画像のラドン変換をマッチングすることにより，物体の位置と姿勢を計算することができる．Fig. 2.2-(a)，(b)，(c) に示すように，原画像を g_0，原画像 g_0 を原点まわりに角度 α 回転させた画像を g_1，原画像 g_0 を角度 β 方向に距離 d_0 並進移動させた画像を g_2 とする．すなわち，

$$g_1(x, y) = g_0(xC_\alpha + yS_\alpha, -xS_\alpha + yC_\alpha)$$
$$g_2(x, y) = g_0(x - d_0C_\beta, y - d_0S_\beta)$$

である．画像 g_0, g_1, g_2 のラドン変換をそれぞれ $R_0(\rho, \theta)$，$R_1(\rho, \theta)$，$R_2(\rho, \theta)$ で表すと，次



(a)original image g_0 (b) rotation g_1



(c) translation g_2 (d) planar motion g_3

Fig. 2.2: Planar motion of rigid object

式が成り立つ.

$$R_1(\rho,\theta) \equiv R_0(\rho,\theta+\alpha), \quad \forall\rho,\theta. \tag{2.2}$$

$$R_2(\rho,\theta) \equiv R_0(\rho - d_0 S_{\theta-\beta},\theta), \quad \forall\rho,\theta. \tag{2.3}$$

Fig. 2.2-(d) に示すように，原画像を角度 α 回転させ，方向 β，距離 d_0 並進させた画像を g_3 とする．このとき，g_3 のラドン変換 $R_3(\rho,\theta)$ に対して，次式が成り立つ.

$$R_3(\rho,\theta) \equiv R_0(\rho - d_0 S_{\theta-\beta},\theta+\alpha), \quad \forall\rho,\theta. \tag{2.4}$$

上式に示すように，並進移動のパラメータ d_0，β は，変数 ρ の部分にのみ現れ，回転移動のパラメータ α は，変数 θ の部分にのみ表れる.

2.3 一次元位相限定相関

前節で述べたように，ラドン変換を用いると，ρ 方向のマッチングにより並進移動のパラメータを，θ 方向のマッチングにより回転移動のパラメータを求めることができる．ただし，ラドン変換は画素値の線積分であり，照明の変動や背景の影響などによりラドン変換の値が変動するため，最小二乗法や相関法ではロバストな検出が困難である．そこで，マッチングに一次元位相限定相関を使用する.

一次元テンプレート信号を $a_0(t)$ とし，$a_0(t)$ を φ シフトした一次元入力信号を $a_1(t)$ とする．一次元位相限定相関では，シフト量 φ を以下の方法で求める．一次元信号 $a_0(t)$ と $a_1(t)$ の関係式は,

$$a_1(t) = a_0(t-\varphi) \tag{2.5}$$

と表せる．一次元信号 $a_0(t)$，$a_1(t)$ の一次元フーリエ変換 $A_0(\omega)$，$A_1(\omega)$ を振幅と位相に分けて，

$$A_0(\omega) = |A_0(\omega)|e^{j\phi_0(\omega)} \tag{2.6}$$
$$A_1(\omega) = |A_1(\omega)|e^{j\phi_1(\omega)} \tag{2.7}$$

と表す．ここで，$|A_0|$ と $|A_1|$ はフーリエ変換 A_0 と A_1 の振幅スペクトルを表し，$\phi_0(\omega)$ と $\phi_1(\omega)$ は，フーリエ変換 A_0 と A_1 の位相成分の偏角を表す．また，ω は周波数を表す．次に，位相スペクトルの商 $C(\omega)$ を計算する．

$$C(\omega) = \frac{e^{j\phi_1(\omega)}}{e^{j\phi_0(\omega)}}. \tag{2.8}$$

式（2.6）,（2.7）より，$C(\omega)$ は，

$$C(\omega) = \frac{A_1(\omega)A_0^*(\omega)}{|A_1(\omega)A_0^*(\omega)|} \tag{2.9}$$

となる．ただし，A_0^* は，A_0 の複素共役である．次に，関数 $C(\omega)$ を逆フーリエ変換し，一次元位相限定相関 $c(t)$ を求める．ここで，$|A_0(\omega)| = |A_1(\omega)|$，ならびに $\phi_1(\omega) = \phi_0(\omega) - \varphi\omega$ が成り立つことに注目すると，

$$c(t) = \delta(t - \varphi) \tag{2.10}$$

となり，位相限定相関関数は，デルタ関数を形成することがわかる．すなわち関数 $c(t)$ は，$t = \varphi$ で最大値を持つ．したがって，$c(t)$ の最大値を探索することで，シフト量 φ を求めることができる．

2.4　提案するビジョンアルゴリズム

本章で提案するアルゴリズムは，ラドン変換と一次元位相限定相関を用いたテンプレートマッチングで位置と姿勢の検出を行う．画像のラドン変換の一次元フーリエ変換は，画像の二次元フーリエ変換の局座標変換と一致することが知られている [10]．フーリエ変換を用いた検出法法 [4, 5, 6, 7, 8, 9] では，画像の二次元パワースペクトルの極座標変換をマッチングすることで姿勢を検出し，画像内の姿勢のずれを修正した後に，画像の二次元位相スペクトルをマッチングすることで位置を検出している．したがって，従来のフーリエ変換を用いた手法では，姿勢検出は振幅スペクトルを基にマッチングを行い，位置検出は位相スペクトルを基にマッチングを行っている．それに対し本章で提案する手法は，振幅スペクトルを基にしたマッチングは使用せず，位相スペクトルを基にしたマッチングのみで位置と姿勢の検出を行う．

2.4.1　単数物体の検出アルゴリズム

テンプレートとなる原画像のラドン変換を $R_t(\rho, \theta_t)$ とし，原画像を角度 α 回転させ，方向 β，距離 d_0 並進させた画像のラドン変換を $R_i(\rho, \theta_i)$ とする．ここで，テンプレート側

の角度を θ_t とし，入力側の角度を θ_i で表す．また，ラドン変換 $R_t(\rho,\theta)$，$R_i(\rho,\theta)$ を ρ 方向に一次元フーリエ変換を行った結果を $F_t(f,\theta)$，$F_i(f,\theta)$ とする．

姿勢検出においては，ラドン変換 $R_t(\rho,\theta)$，$R_i(\rho,\theta)$ の各 θ_t と θ_i に対して一次元位相限定相関を行う．姿勢検出で使用する位相限定相関関数 $p(\rho,\theta_t,\theta_i)$ は，次式の $P(f,\theta_t,\theta_i)$ を逆フーリエ変換することで得られる．

$$P(f,\theta_t,\theta_i)=\frac{F_i(f,\theta_i)F_t^*(f,\theta_t)}{|F_i(f,\theta_i)F_t^*(f,\theta_t)|}. \tag{2.11}$$

ここで，F_t^* は F_t の複素共役である．次に $p(\rho,\theta_t,\theta_i)$ より，各 θ_t，θ_i における ρ 方向の最大値を $peek(\theta_t,\theta_i)$ に記録する．すなわち，

$$peek(\theta_t,\theta_i)=\max_{\rho} p(\rho,\theta_t,\theta_i) \tag{2.12}$$

である．このとき，$peek(\theta_t,\theta_i)$ のなかで高い値をとるのは，式（2.4）より，

$$\alpha=\theta_t-\theta_i \tag{2.13}$$

となるときである．そこで，角度 a に対して，次式で表される $sum(a)$ を計算する．

$$sum(a)=\sum_{\theta_t=0}^{2\pi} peek(\theta_t,\theta_t-a). \tag{2.14}$$

関数 $sum(a)$ は，$a=\alpha$ で最大値をとる．したがって，$sum(a)$ の最大値を求め，そのときの角度を物体の姿勢とする．

次に位置の検出について述べる．まず，姿勢検出で求めた姿勢 α より，テンプレート側と入力側のラドン変換における θ 方向のずれを修正する．次に各 θ に対して一次元位相限定相関を行う．位置検出に使用する位相限定相関関数 $q(\rho,\theta)$ は，次式で与えられる $Q(f,\theta)$ を逆フーリエ変換することで得られる．

$$Q(f,\theta)=\frac{F_i(f,\theta)F_t^*(f,\theta+\alpha)}{|F_i(f,\theta)F_t^*(f,\theta+\alpha)|}. \tag{2.15}$$

式（2.4）より，各 θ に対して，位相限定相関関数 $q(\rho,\theta)$ は，$\rho=d_0S_{\theta-\beta}$ で，最大値を取ることがわかる．一方，点 $[x,y]=[d_0C_\beta,d_0S_\beta]$ を通る直線は，

$$\rho(\theta)=d_0\sin(\theta-\beta) \tag{2.16}$$

で表される．すなわち，x-y 平面内の点 $[d_0C_\beta,d_0S_\beta]$ をハフ変換すると，ρ-θ 平面内の曲線 $\rho(\theta)=d_0\sin(\theta-\beta)$ が得られる．したがって，ρ-θ 平面内の曲線 $\rho(\theta)=d_0\sin(\theta-\beta)$ を逆ハフ変換し，x-y 平面 $h_{inv}(x,y)$ 内に直線を投票すると，点 $[d_0C_\beta,d_0S_\beta]$ の位置で直線群が交差する．したがって，$h_{inv}(x,y)$ で，投票が最も多い点を求めることで，物体の位置を求めることができる．

2.4.2　ラドン変換と検出精度の関係

前項では，単数物体の位置と姿勢を検出するアルゴリズムについて述べた．本項では，ラドン変換と検出精度の関係について述べる．本章で提案するアルゴリズムは，ラドン変換をもとに位置と姿勢を検出しているため，検出精度はラドン変換の配列数と分解能の影響を受ける．ラドン変換の角度 θ のステップ数を K とし，距離 ρ のステップ数を L とする．ここで，角度 θ の分解能を $\Delta\theta$ とし，距離 ρ の分解能を $\Delta\rho$，逆ハフ変換で直線を投票するとする x-y 平面の分解能を Δd とする．また，ρ 軸側に一次元フーリエ変換を行うため，フーリエ変換の周波数 f のステップ数は，L となる．

姿勢検出の精度では，ラドン変換における角度 θ の分解能 $\Delta\theta$ が影響する．分解能 $\Delta\theta$ は，$\Delta\theta = 2\pi/K$ となるため，ステップ数 K によって分解能 $\Delta\theta$ は決定される．分解能 $\Delta\theta$ が小さいほど，精度の高い検出が可能となるため，ラドン変換における検出精度は，ステップ数 K に依存する．次に位置検出では，参照側と入力側の角度 θ のずれを修正した後，各 θ に対して一次元位相限定相関を行い，各 θ のピーク値に対して逆ハフ変換を行い，位置を求めている．そのため，式（2.16）より，逆ハフ変換の投票数は，ステップ数 K に依存する．また，ラドン変換における，距離 ρ の分解能 $\Delta\rho$ が小さいほど，逆ハフ変換の精度は向上するため，位置検出の精度は高くなる．ただし，位置検出では，直線を投票する x-y 平面の分解能 Δd も影響するため，分解能 Δd が大きいと位置検出の精度は悪くなる．

2.4.3　一次元位相限定相関の実行回数

本項では，アルゴリズムに用いた一次元位相限定相関の実行回数について述べる．

姿勢検出において，一次元位相限定相関を用いる式（2.11）の計算では，参照側と入力側のすべての θ に対して一次元位相限定相関を行う必要があるため，その実行回数は K^2 となる．位置検出において，一次元位相限定相関を用いる式（2.15）の計算では，角度 θ のステップ数だけ一次元位相限定相関を行うため，その実行回数は K となる．

2.5　静止画像を用いた位置と姿勢検出

2.5.1　実験環境

ここでは，検出実験を行う環境について述べる．実験に使用する静止画像のサイズは 320×240[pixel] であり，ラドン変換の ρ のステップ数を $L = 256$, θ のステップ数を $K = 256$ とし，$\Delta\rho = 2$ と設定した．また，位相限定相関に使用する高速フーリエ変換 (FFT) の周波数成分の数は，ρ のステップ数 L と同じ 256 である．一次元位相限定相関の最大値は，周波数成分の数と同じ 256 である．検出実験には Pentium4 1.8GHz，メモリ 512MB 搭載の PC を使用し，OS は Linux である．プログラムのコンパイラには，Intel C++ Compiler6.0 を使用した．

Table 2.1: Computation of Radon transform

initialize array $R(\theta_k, \rho_h)$
for all ξ do
  for all ρ_h do
    for $k = 0, 1, \cdots, K-1$ do
      $\theta_k = k\Delta\theta$, $C_k = \cos\theta_k$, $S_k = \sin\theta_k$
      compute $x = \xi C_\theta - \rho_h S_\theta$
      compute $y = \xi C_\theta + \rho_h C_\theta$
      add pixel value $g_{x,y}$ to $R(\theta_k, \rho_h)$
      end
    end
  end
end

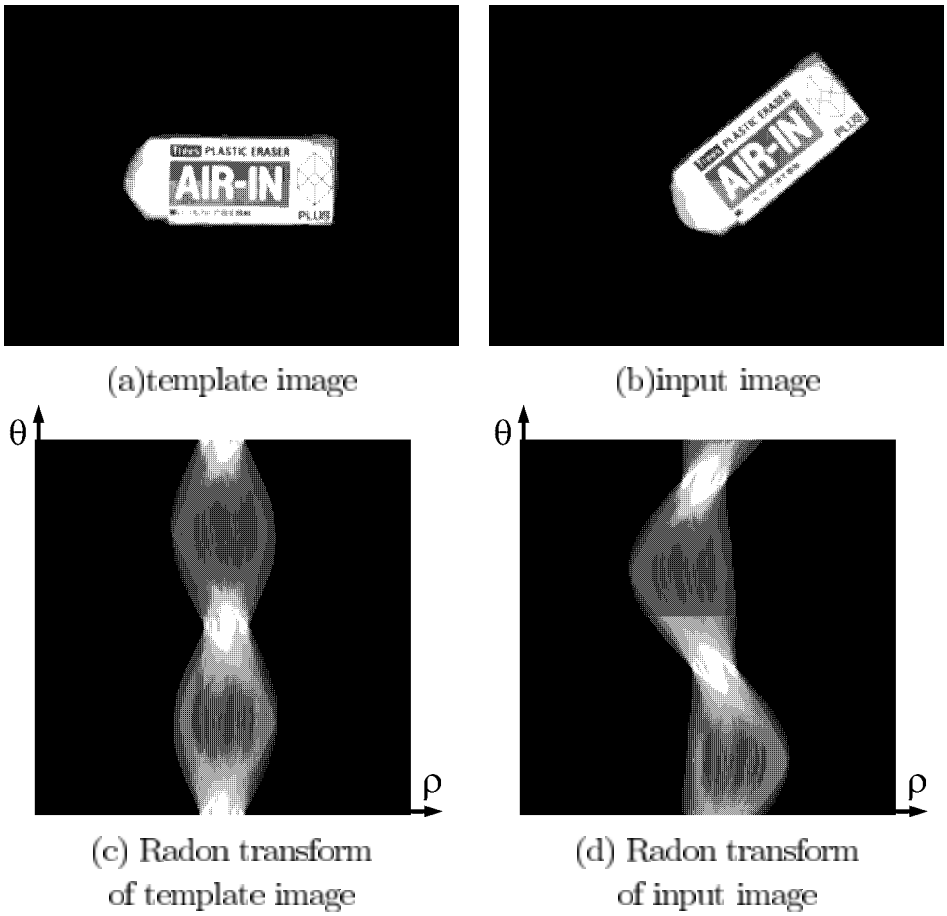
Table 2.2: Parallel computation of Radon transform

initialize array $R(\theta_k, \rho_h)$
for $g(x, y) = (0, 0), (0, 1), \cdots, (W-1, H-1)$ do
  for $k = 0, 1, \cdots, K-1$ do
    $\theta_k = k\Delta\theta$, $C_k = \cos\theta_k$, $S_k = \sin\theta_k$
    compute $\rho = -xS_k + yC_k$
    compute $\rho_h = \rho/\Delta\rho$
    add pixel value $g_{x,y}$ to $R(\theta_k, \rho_h)$
    end
  end
end

2.5.2 ラドン変換の並列化

本節では，ラドン変換の並列化について述べる．画像入力においては，画像の座標値 x，y と画素値 $g(x, y)$ が順次得られる．式（2.1）より，座標値は，$x = \xi C_\theta - \rho S_\theta$，$y = \xi S_\theta + \rho C_\theta$ と表される．式（2.1）の処理を擬似コードで表すと，Table 2.1 となる．ここで，$\Delta\rho$ は ρ の間隔を表し，$\rho_h = h\Delta\rho$ $(h = 0, \pm 1, \pm 2, \cdots)$ とする．ここで，$\theta_k = k\Delta\theta$ とする．

Table 2.1 より，パラメータ ρ，ξ，θ から，x，y を求めるラドン変換の方式では，キャプチャーした画像データを保存した後でなければ，ラドン変換を行うことができない．そこで，画像のキャプチャーと並行にラドン変換を行うようにするため，パラメータ x，y，θ から ρ を求める $\rho = yC_\theta - xS_\theta$ に変更する．変更した処理を擬似コードで表すと，Table



(a)template image　　(b)input image

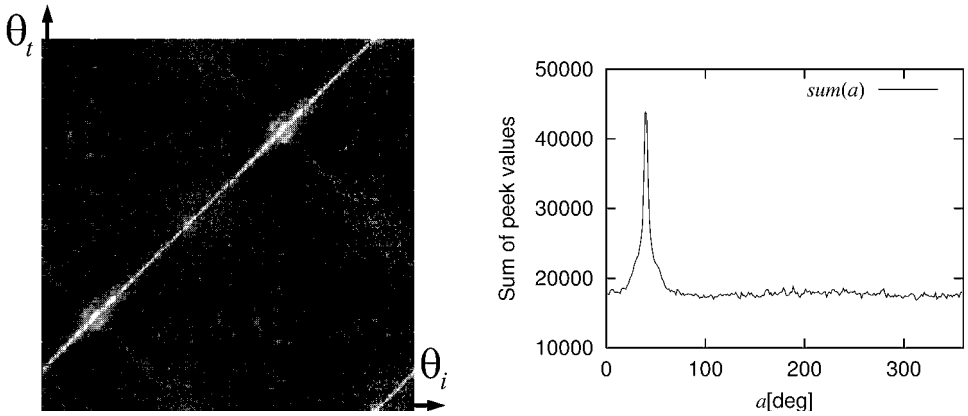
(c) Radon transform of template image　　(d) Radon transform of input image

Fig. 2.3: Images and their Radon transforms

Table 2.3: Computation time

処理名	処理時間 [msec]
ラドン変換	315
姿勢検出	1500
位置検出	12
合計	1900

2.2 となる．ここで，W は画像の横サイズ，H は画像の縦サイズとする．この方法では，キャプチャーした画像データを保存する必要がなくなるため，アルゴリズムを LSI 上に実装する場合，メモリ消費を抑えることができる．また，θ_k に対して並列処理化が可能なため，処理時間の短縮が見込まれる．



(a) $peek(\theta_t, \theta_i)$ (b) $sum(a)$

Fig. 2.4: Detection of rotation

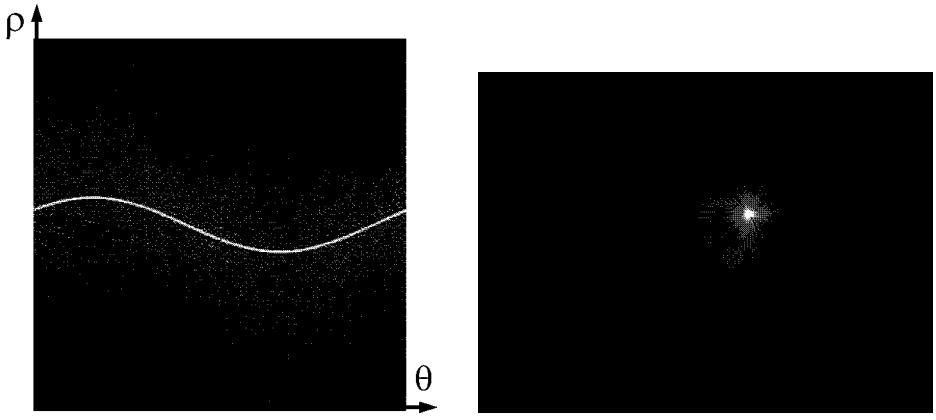
2.5.3 対象物体の検出

単数物体の検出実験では，原画像と原画像を幾何学的変換によって回転と並進移動させた入力画像を用いて，物体の位置と姿勢検出の評価を行った．この実験では，背景による影響を避けるため，背景の画素値を 0 にしている．検出実験に用いた原画像を Fig. 2.3 -(a) に示し，幾何学的変換により原画像を反時計回りに 40[deg] 回転させ，x 方向に 30[pixel], y 方向に 20[pixel] 並進移動させた入力画像を Fig. 2.3-(b) に示す．また，Fig. 2.3-(a)，(b) の画像のラドン変換を，それぞれ Fig. 2.3-(c)，(d) に示す．

提案するアルゴリズムによる検出結果は，回転角度 39.4[deg]，x 方向の移動距離 30[pixel], y 方向の移動距離 19[pixel] となった．また，検出実験の処理に要した時間を Table 3.4 に示す．ここで，姿勢検出と位置検出の評価を行う．姿勢検出においては，Fig. 2.4-(a) に示すように，$peek(\theta_t, \theta_i)$ の値は，特定の直線に高い値が表れる．このときの $sum(a)$ は，Fig. 2.4-(b) に示すように，該当する姿勢で最大値をとる．姿勢検出時の誤差は，θ の分解能が約 1.4[deg] であるため，計算誤差の範囲内であるといえる．次に位置検出の評価を行う．位置検出で使用した位相限定相関関数 $q(\rho, \theta)$ の結果を Fig. 2.5-(a) に示す．また，$q(\rho, \theta)$ において $\rho(\theta)$ の部分を逆ハフ変換した $h_{inv}(x, y)$ を Fig. 2.5-(b) に示す．逆ハフ変換 $h_{inv}(x, y)$ における x-y 平面の分解能は 1[pixel] であり，投票された直線はほぼ一点に集中して交差しているため，位置検出時の誤差は計算誤差の範囲内であるといえる．したがって，提案するアルゴリズムにより，物体の位置と姿勢を検出することが可能である．

2.5.4 対象外物体の判別

次に Fig. 2.6-(a) に示す対象外の物体画像と Fig. 2.3-(a) に示すテンプレート画像を用いて，提案するアルゴリズムによる検出実験を行った．この実験より，対象物体と対象外物体の選別方法についての考察を行う．対象外物体のラドン変換を Fig. 2.6-(b) に示し，姿勢検出結果を Fig. 2.6-(c)，(d) に示す．Fig. 2.6-(c) より，対象外の物体では，$peek(\theta_t, \theta_i)$ に高いピーク値が直線状になって表れない．また，Fig. 2.6-(d) より，$sum(a)$ のグラフはほぼ平坦になっており，ピークの山が表れない．したがって，姿勢検出時に関数 $sum(a)$



(a) $q(\rho, \theta)$　　(b) $h_{inv}(x, y)$

Fig. 2.5: Detection of position

の最大値と平均値を計算し，最大値と平均値の差が小さい場合は，対象外の物体として判別する．

2.6　複数物体の検出

これまでの検出実験では，単数の物体を対象としてきた．そこで，複数の物体を検出するため，複数の姿勢と複数の位置を検出する方法について検討し，検出実験を行った．実験環境は第 5 章と同じであり，実験に用いた画像は，原画像を幾何学的変換によって回転と並進移動させた画像を合成したものを使用している．この実験で使用した入力画像を Fig. 2.7 に示し，これらの画像のラドン変換を Fig. 2.8 に示す．

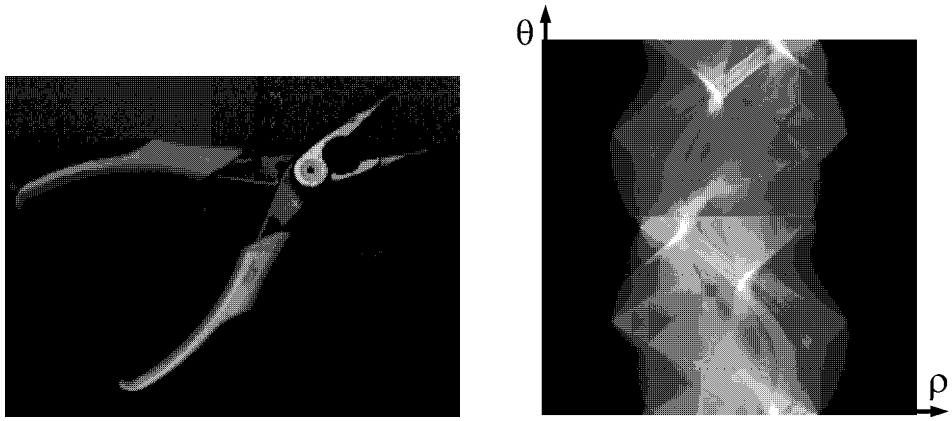
2.6.1　複数の姿勢検出

Fig. 2.3-(a) に示す画像をテンプレートとし，Fig. 2.7-(1)，(2)，(3)，(4) を入力画像として，姿勢検出実験を行った．検出実験の結果を Fig. 2.9 に示す．本論文では，位相限定相関を用いたマッチングを行うため，画像内に複数の異なる姿勢の対象物体があるとき，$sum(a)$ はそれぞれの該当する姿勢のところにピークが表れる．したがって，$sum(a)$ に表れる複数のピークを検出することで，複数の姿勢を検出することが可能になる．そこで，$sum(a)$ を微分し，ゼロ交差を検出することで，複数の姿勢を求める方法を提案する．まず，$sum(a)$ の平均値 ave_s を求め，$sum(a)$ から ave_s を引く．このとき，0 以下の値は 0 とする．すなわち，次式で定義される $sum^+(a)$ を計算する．

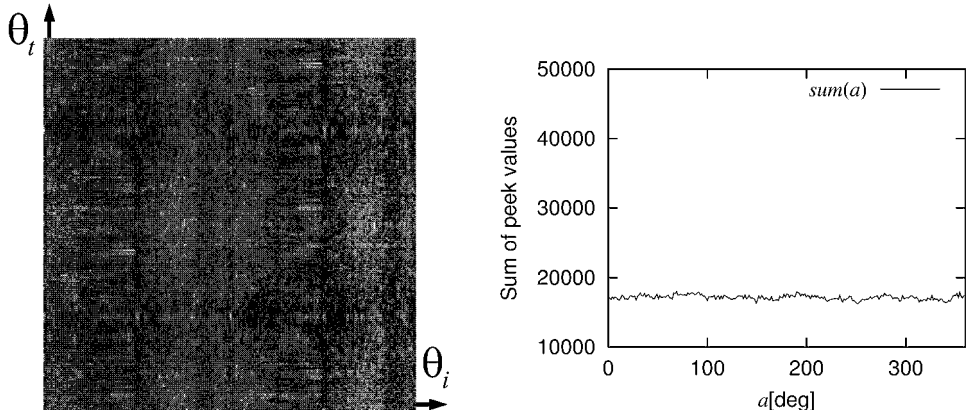
$$sum^+(a) = \max(sum(a) - ave_s, 0). \tag{2.17}$$

次に $sum^+(a)$ の a 方向に微分を行い，ゼロ交差を検出する．検出したゼロ交差から角度 a_z を求め，$sum^+(a_z)$ が閾値 t_s 以上のとき，a_z を該当する姿勢として検出する．

この姿勢検出方法では，該当する姿勢以外のところも検出する可能性がある．しかし，誤検出した姿勢では，位置検出の位相限定相関でピークが表れず，位置を検出できないた



(a) The other object image (b) Radon transform



(c) $peek(\theta_t, \theta_i)$ (d) $sum(a)$

Fig. 2.6: Matching with another object

め，位置検出時に誤検出の姿勢と判別することが可能である．ただし，該当する姿勢に近いところの姿勢では，ピーク値が高く位置検出可能な場合があるため，位置検出の後に検出候補を絞る必要がある．

2.6.2 複数の位置検出

前節と同じテンプレートを使用し，入力画像にはFig. 2.7-(1)，(5)，(6)を使用する．この二つの画像を用いて位置検出の実験を行った．画像内に同姿勢の対象物体が複数あるとき，該当する姿勢で位置検出を行うと，Fig. 2.10に示すように，$q(\rho, \theta)$内で複数のピークが表れる．また，位相限定相関$q(\rho, \theta)$の平均値をave_qとし，ave_q以上の$q(\rho, \theta)$に対して逆ハフ変換を行うと，$h_{inv}(x, y)$内で対象物体の位置に直線の交差が集中する．したがって，直線の交差が集中する点を見つけることで，同じ姿勢を持つ複数の物体を検出することが可能になる．そこで，複数の位置を検出する場合，$h_{inv}(x, y)$に対し，閾値T_l以上の直線交差数をラベリングする．直線交差数が閾値未満の場合は，このとき検出した姿勢は誤りとし，対象外の物体とする．次に，ラベリングごとに直線交差数が最大となる$h_{inv}(x, y)$を求めることで，複数の位置を検出する．

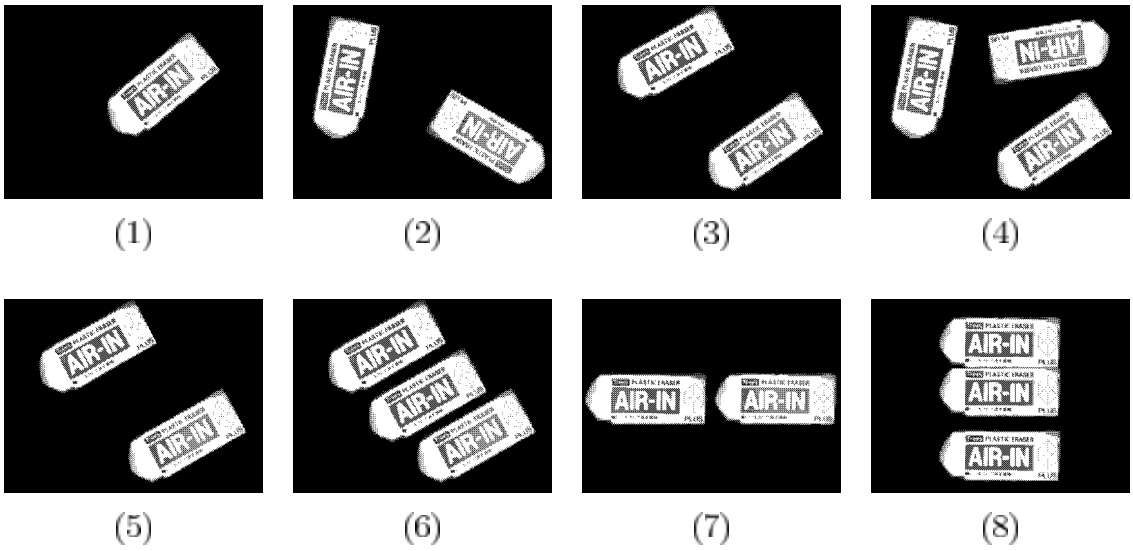


Fig. 2.7: Input images

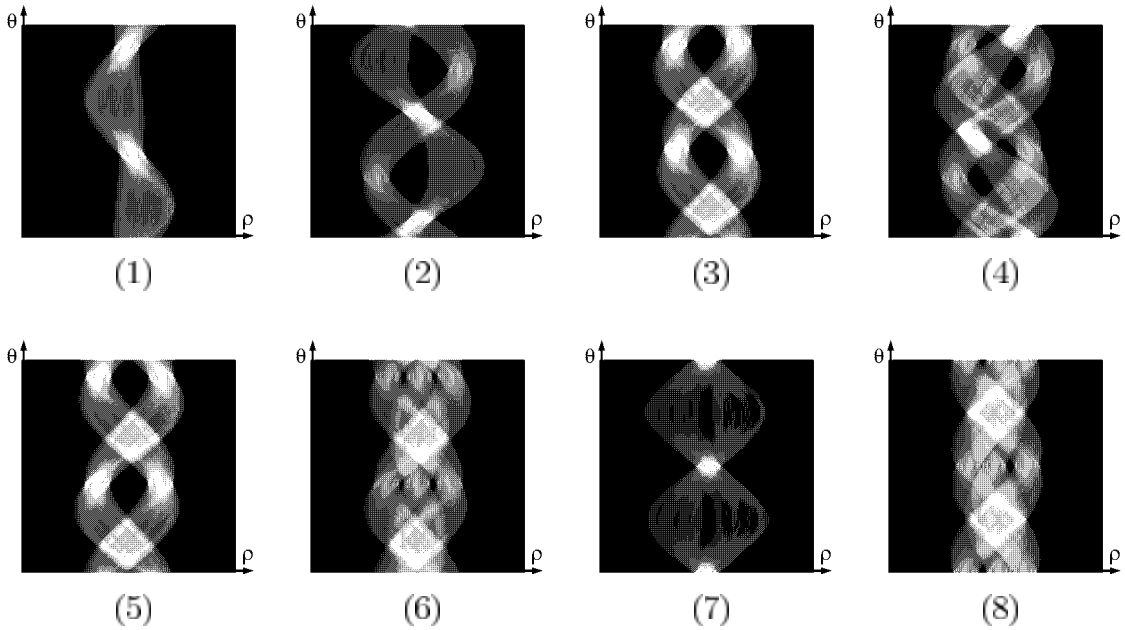


Fig. 2.8: Radon transform of input images

2.6.3　検出実験

姿勢検出の閾値 $t_s = 0.1$ と設定し，位置検出の閾値 $T_l = 32$ と設定して，検出実験を行った．入力画像とその検出結果の位置と姿勢を Table 2.4 に示す．ここで表内の括弧は，誤差を表している．この実験結果より，Fig. 2.7-(2)〜(4) に示す異なる姿勢での対象物体の検出実験では，異なる姿勢であっても物体の位置と姿勢を検出することができた．次に Fig. 2.7-(5)〜(8) に示す同姿勢での対象物体の場合でも，物体の位置と姿勢を検出することができた．ここで，Fig. 2.7-(7)，(8) は，ラドン変換の水平もしくは垂直方向の線積分上に同姿勢の物体が並んだ場合である．これら複数物体の位置と姿勢を検出できるのは，ラドン変換があらゆる角度で線積分の投影を行っており，Fig. 2.8-(7)，(8) に示すように

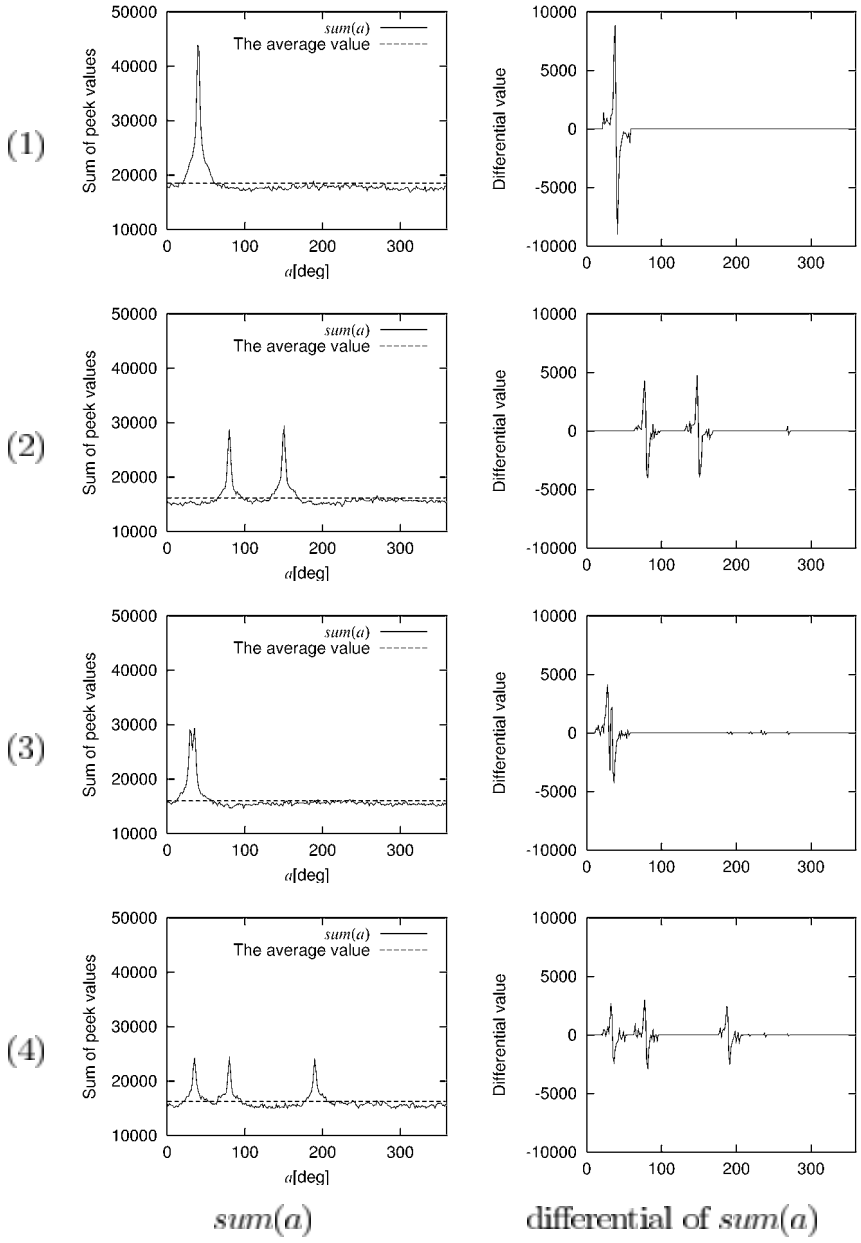


Fig. 2.9: Detection of diffrent rotations

ρ軸側に投影されたデータには画像内に含まれる物体の数だけ位相情報が含まれているからである．次に，この実験で位置と姿勢の検出に要した時間について述べる．検出に要した時間は，最大で約 3.2[sec] であった．理想的な検出に要する時間は，Table 3.4 の処理時間に異なる姿勢の数だけ位置検出時間が追加されるはずである．しかし，今回の実験では，姿勢検出でのゼロ交差が該当する姿勢以外にも多数存在しているため，さらに位置検出の処理時間を要する結果となっている．したがって，無駄のない検出処理を行うためには，なるべく該当する姿勢のみのゼロ交差を残すように閾値を設定する必要がある．

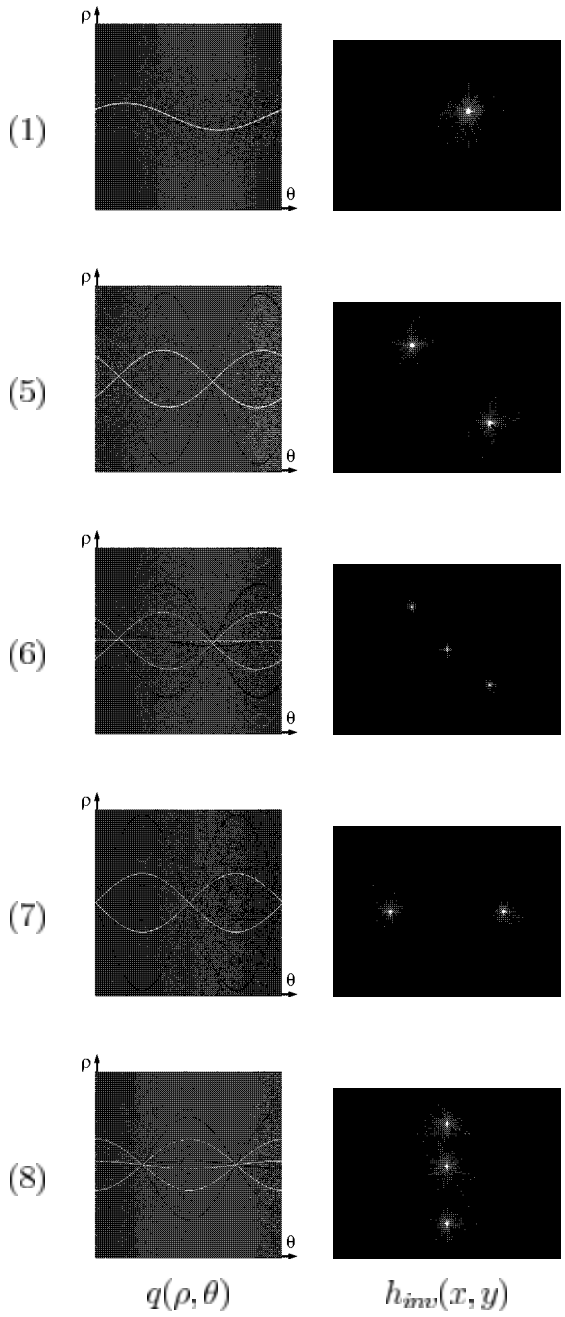


Fig. 2.10: Detection of different positions

2.7　ラドン変換の分解能の違いによる検出実験

2.7.1　アルゴリズム内の数値の正規化

前節の検出実験では，一次元位相限定相関の値は正規化していなかったため，一次元位相限定相関の最大値は，ρ のステップ数 L に等しかった．しかし，正規化を行わないでラドン変換のステップ数を変更したとき，一次元位相限定相関の最大値が変動してしまう．そのため，ラドン変換のステップ数を変更するたびに，検出における閾値設定をやり直さねばならなくなる．そこで，ρ のステップ数 L で一次元位相限定相関値を割り，位相限定

Table 2.4: Experimentally detcted position and orientation

入力画像			
画像番号	姿勢 [deg]	変位量 x [pixel]	変位量 y [pixel]
(1)	40	30	20
(2)	80	-100	30
	150	80	-50
(3)	30	-50	60
	35	60	-50
(4)	35	60	-50
	80	-100	30
	190	60	60
(5)	30	60	-50
	30	-50	-60
(6)	30	0	0
	30	60	50
	30	-50	60
(7)	0	80	0
	0	-80	0
(8)	0	0	70
	0	0	10
	0	0	-70

検出結果			
画像番号	姿勢 [deg]	変位量 x [pixel]	変位量 y [pixel]
(1)	39.4(-0.6)	30(0)	19(-1)
(2)	80.2(+0.2)	-100(0)	30(0)
	150.5(+0.5)	79(-1)	-47(+3)
(3)	29.5(-0.5)	-50(0)	61(+1)
	35.2(+0.2)	60(0)	-49(+1)
(4)	35.2(+0.2)	60(0)	-49(+1)
	80.2(+0.2)	100(0)	30(0)
	189.8(-0.2)	59(-1)	61(+1)
(5)	29.5(-0.5)	60(0)	-49(+1)
	29.5(-0.5)	-50(0)	60(0)
(6)	29.5(-0.5)	0(0)	0(0)
	29.5(-0.5)	60(0)	-49(+1)
	29.5(-0.5)	-50(0)	60(0)
(7)	0.0(0.0)	79(-1)	0(0)
	0.0(0.0)	-80(0)	0(0)
(8)	0.0(0.0)	0(0)	70(0)
	0.0(0.0)	0(0)	10(0)
	0.0(0.0)	0(0)	-69(+1)

相関の最大値が 1 になるように正規化する．

また，姿勢検出で使用する式（2.16）では，最大値が K となるため，最大値が 1 になるように正規化を行う．

$$sum(a) = \frac{\sum_{\theta_t=0}^{2\pi} peek(\theta_t, \theta_t - a)}{K}. \tag{2.18}$$

これらの正規化により，ラドン変換のステップ数の変動によって姿勢検出で使用する閾値 t_s を変更する必要がなくなる．

2.7.2 ラドン変換の分解能に対する考察

本項では，ラドン変換の分解能を変更したときの検出結果について考察する．本検出実験では，テンプレート画像は Fig. 2.3-(a) を使用し，入力画像は Fig. 2.7 を使用する．

ラドン変換の配列数は，

1. θ のステップ数 $K = 256$，ρ のステップ数 $L = 256$
2. θ のステップ数 $K = 256$，ρ のステップ数 $L = 512$
3. θ のステップ数 $K = 512$，ρ のステップ数 $L = 256$

4. θ のステップ数 $K = 512$，ρ のステップ数 $L = 512$

の四つに分類して検出実験を行った．ここで，$K = 256$ のとき $\Delta\theta = 1.40[\mathrm{deg}]$ であり，$K = 512$ のとき $\Delta\theta = 0.70[\mathrm{deg}]$ である．$L = 256$ のときは $\Delta\rho = 2[\mathrm{pixel}]$ と設定し，$L = 512$ のときは，$\Delta\rho = 1[\mathrm{pixel}]$ と設定している．また，姿勢検出の閾値 $t_s = 0.1$ と設定し，位置検出の閾値 T_l は，$T_l = K/8$ と設定した．ラドン変換のステップ数が $(K = 256,\ L = 256)$ のときの検出結果を Table 2.5 に示し，$(K = 256,\ L = 512)$ のときの検出結果を Table 2.6 に示し，$(K = 512,\ L = 256)$ のときの検出結果を Table 2.7 に示し，$(K = 512,\ L = 512)$ のときの検出結果を Table 2.8 に示す．

この検出結果より，分解能 $\Delta\theta$ を小さくすることで，姿勢検出の精度は向上した．しかし，位置検出では，分解能 $\Delta\theta$ と分解能 $\Delta\rho$ を小さくしても，検出精度は，ほとんど向上していない．また，同姿勢の複数物体よりも異なる姿勢の複数物体のほうが，位置検出の誤差頻度が多い．位置検出の精度が上がらなかった原因として，逆ハフ変換を実行するときの x-y 平面の分解能が粗いということが考えられる．また，逆ハフ変換により x-y 平面に直線を投票するとき，直線に重み付けを行っていないためジャギーが発生し，直線の交差点がぶれてしまったことが考えられる．

Table 2.5: Detcted result ($\theta : 256, \rho : 256$)

検出結果			
画像番号	姿勢 [deg]	変位量 x [pixel]	変位量 y [pixel]
(1)	39.4(-0.6)	29(-1)	20(0)
(2)	80.2(+0.2)	-98(+2)	31(+1)
	150.5(+0.5)	79(-1)	-47(+3)
(3)	29.5(-0.5)	-50(0)	61(+1)
	35.2(+0.2)	60(0)	-49(+1)
(4)	35.2(+0.2)	60(0)	-49(+1)
	80.2(+0.2)	-99(1)	31(+1)
	189.8(-0.2)	59(-1)	61(+1)
(5)	29.5(-0.5)	60(0)	-49(+1)
	29.5(-0.5)	-50(0)	60(0)
(6)	29.5(-0.5)	0(0)	0(0)
	29.5(-0.5)	60(0)	-49(+1)
	29.5(-0.5)	-50(0)	60(0)
(7)	0.0(0.0)	79(-1)	0(0)
	0.0(0.0)	-80(0)	0(0)
(8)	0.0(0.0)	0(0)	70(0)
	0.0(0.0)	0(0)	10(0)
	0.0(0.0)	0(0)	-69(+1)

Table 2.6: Detcted result ($\theta : 256, \rho : 512$)

検出結果			
画像番号	姿勢 [deg]	変位量 x [pixel]	変位量 y [pixel]
(1)	39.4(-0.6)	30(0)	20(0)
(2)	80.2(+0.2)	-99(+1)	31(+1)
	150.5(+0.5)	80(0)	-47(+3)
(3)	29.5(-0.5)	-50(0)	60(0)
	35.2(+0.2)	60(0)	-49(+1)
(4)	35.2(+0.2)	60(0)	-49(+1)
	80.2(+0.2)	-99(1)	31(+1)
	189.8(-0.2)	59(-1)	62(+2)
(5)	29.5(-0.5)	60(0)	-49(+1)
	29.5(-0.5)	-50(0)	60(0)
(6)	29.5(-0.5)	0(0)	0(0)
	29.5(-0.5)	60(0)	-49(+1)
	29.5(-0.5)	-50(0)	60(0)
(7)	0.0(0.0)	79(-1)	0(0)
	0.0(0.0)	-80(0)	0(0)
(8)	0.0(0.0)	0(0)	70(0)
	0.0(0.0)	0(0)	10(0)
	0.0(0.0)	0(0)	-69(+1)

Table 2.7: Detcted result ($\theta : 512, \rho : 256$)

検出結果			
画像番号	姿勢 [deg]	変位量 x [pixel]	変位量 y [pixel]
(1)	40.1(+0.1)	29(-1)	20(0)
(2)	80.2(+0.2)	-99(+1)	31(+1)
	149.8(-0.2)	80(0)	-48(+2)
(3)	30.2(+0.2)	-50(0)	60(0)
	35.2(+0.2)	60(0)	-49(+1)
(4)	35.2(+0.2)	60(0)	-49(+1)
	80.2(+0.2)	-99(1)	31(+1)
	189.8(-0.2)	59(-1)	62(+2)
(5)	30.2(+0.2)	60(0)	-49(+1)
	30.2(+0.2)	-49(+1)	61(+1)
(6)	30.2(+0.2)	0(0)	0(0)
	30.2(+0.2)	60(0)	-49(+1)
	30.2(+0.2)	-50(0)	60(0)
(7)	0.0(0.0)	79(-1)	0(0)
	0.0(0.0)	-80(0)	0(0)
(8)	0.0(0.0)	0(0)	70(0)
	0.0(0.0)	0(0)	10(0)
	0.0(0.0)	0(0)	-69(+1)

Table 2.8: Detcted result ($\theta : 512, \rho : 512$)

検出結果			
画像番号	姿勢 [deg]	変位量 x [pixel]	変位量 y [pixel]
(1)	40.1(+0.1)	30(0)	20(0)
(2)	80.2(+0.2)	-99(+1)	31(+1)
	149.8(-0.2)	80(0)	-47(+3)
(3)	30.2(+0.2)	-49(+1)	60(0)
	35.2(+0.2)	60(0)	-49(+1)
(4)	35.2(+0.2)	60(0)	-49(+1)
	80.2(+0.2)	-99(1)	31(+1)
	189.8(-0.2)	59(-1)	62(+2)
(5)	30.2(+0.2)	60(0)	-49(+1)
	30.2(+0.2)	-49(+1)	60(0)
(6)	30.2(+0.2)	0(0)	0(0)
	30.2(+0.2)	60(0)	-49(+1)
	30.2(+0.2)	-50(0)	60(0)
(7)	0.0(0.0)	79(0)	0(0)
	0.0(0.0)	-80(0)	0(0)
(8)	0.0(0.0)	0(0)	70(0)
	0.0(0.0)	0(0)	10(0)
	0.0(0.0)	0(0)	-69(+1)

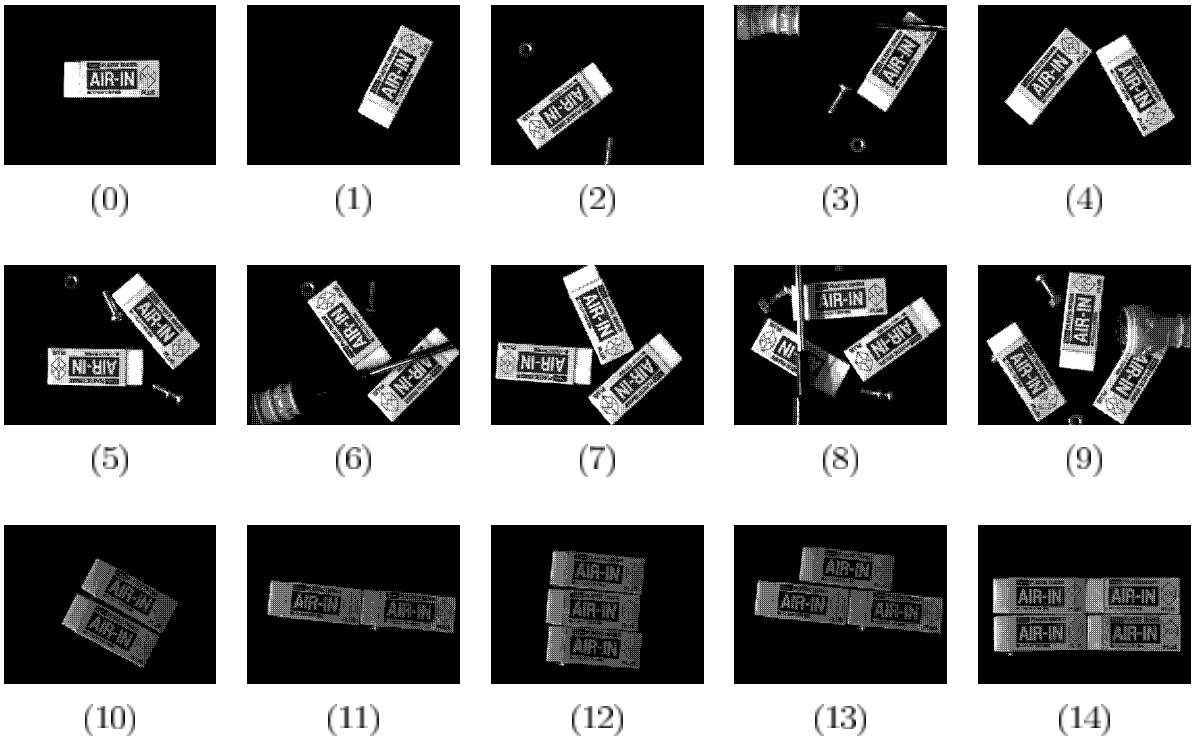


Fig. 2.11: Captured images

2.8 実画像を用いた複数物体検出

これまでの検出実験では，画像処理ソフトによって合成した画像を用いてきたため，本実験では，実際に CCD からキャプチャーした画像を用いて実験を行う．CCD カメラからキャプチャーした画像を Fig. 2.11 に示す．また，検出実験に使用するテンプレート画像は，Fig. 2.11-(0) とした．なお，姿勢検出の閾値は $t_s = 0.1$ とし，位置検出の閾値は $T_l = 32$ に設定した．Fig. 2.11 の検出結果を Fig. 2.12 に示す．ここで，検出結果を白枠線で表示するようにしている．この実験結果より，Fig. 2.11-(1)〜(9) に示す姿勢が異なるときの対象物体検出では，対象物体が 1，2 個のときではほぼ正確に検出している．しかし対象物体が 3 個のときでは，すべて検出できるとは限らない．Fig. 2.11-(10)〜(14) に示す同姿勢で並んだ対象物体の場合では，対象物体をすべて検出することができた．第 6 章の実験で示したように，対象物体が増えるたびに位相限定相関値が低くなっている．また，対象物体の検出ができなかった Fig. 2.11-(6)，(8)，(9) の画像では，対象外の物体によって対象物体の一部が隠蔽されてしまっているため，さらに位相限定相関値が低くなる．そのため，検出できなかった原因として，姿勢検出で該当する姿勢の $sum^+(a_z)$ が閾値 t_s 未満であった場合と位置検出で平均値 ave_q を超える位相限定相関 $q(\rho,\theta)$ が減少してしまい，直線交差数が閾値 T_l 未満だった場合が考えられる．そこで，閾値設定を変更して,，Fig. 2.11-(6)，(8)，(9) 画像に対して検出実験を行ったところ，$t_s = 0.04$，$T_l = 20$ のとき，Fig. 2.13 に示すように，画像内のすべての対象物体を検出することができた．したがって，検出できなかった原因は，閾値設定ということが判明した．このことから，複数の物体を検出する場合は，事前に検出する画像環境で対象物体の位相限定相関値を計測

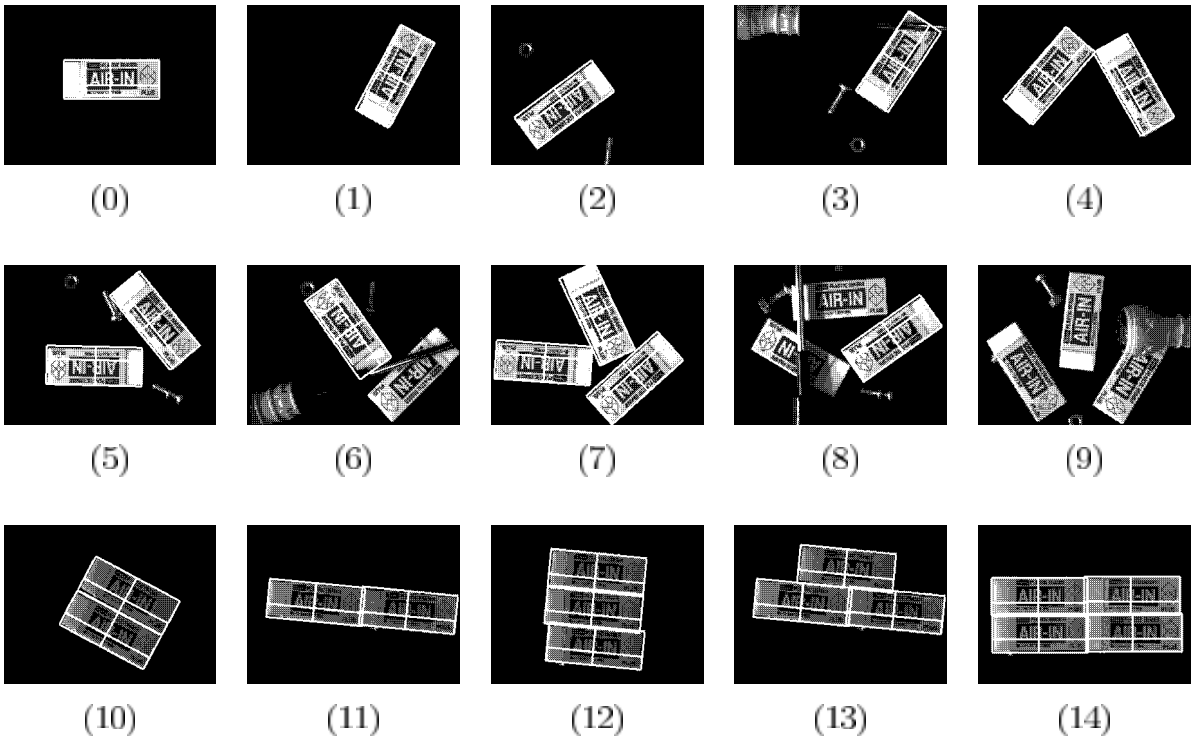


Fig. 2.12: Detection locations in captured images

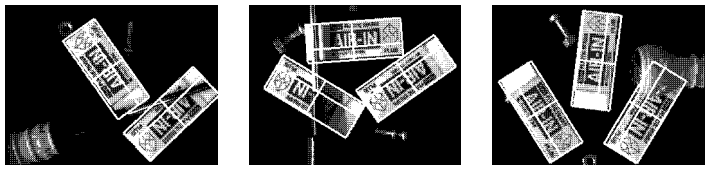


Fig. 2.13: Re-detection of captured images ($T_s = 0.04, T_l = 20$)

し，その計測値をもとに閾値を設定する必要がある．

2.9　結言

本章では，ラドン変換と一次元位相限定相関を用いた対象物体の位置と姿勢を検出する手法を提案した．提案する手法では，ラドン変換により回転変位と並進変位を分離し，一次元位相限定相関を用いたマッチングにより，複数の対象物体を検出する．合成画像を用いた実験では，姿勢の誤差は ± 0.6[deg] 以内に入っており，位置の誤差は ± 3[pixel] に入っている．実画像を用いた実験では，閾値の設定しだいで三つの対象物体を検出することが可能である．以上のことから，提案する手法が，対象物体の位置と姿勢を検出する方法として有効であることが示された．

今回の実験では，検出処理に秒単位の時間を要している．NTSC 信号のビデオフレームレートが周期 33[msec] であることから，本章で提案するアルゴリズムは，ソフトウェアによる実装でビデオフレームレートでの検出は困難である．したがって，本章で提案する

アルゴリズムを用いて高速な検出を行う場合，アルゴリズムをハードウェアに実装する必要がある．

第3章　片側ラドン変換を用いた平面運動検出

3.1　緒言

本章では，平面運動を検出することができ，ソフトウェアによる実装で，ビデオフレームレート内での処理が可能となるビジョンアルゴリズムを提案する．

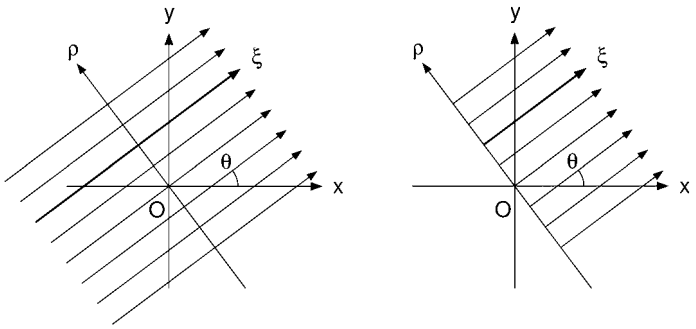
コンピュータ技術の発達により，PC の高性能化が進み，画像処理専用のハードウェアを用いなくても，PC 上で画像処理による物体検出が行えるようになってきている [11]．しかし，ソフトウェア上で高速な検出処理を考慮した場合，計算量の多いアルゴリズムでは高速な検出が困難であるため，第 2 章の考察から少ない計算量のアルゴリズムを用意する必要がある．計算量の少ない平面運動物体の位置と姿勢の検出法として，

1. フーリエ記述子法 [12, 13]
2. 重心モーメント法 [14]

が挙げられる．フーリエ記述子法では，参照画像と入力画像内の物体の輪郭を比較する．フーリエ記述子法は，検出に要する計算量は少ない．しかし，検出できる物体は，輪郭を表す関数を定義できる物体のみであるため，一般に穴を有する物体や同じ形状で模様のある物体の識別は困難である．重心モーメント法では，重心からの正規化されたモーメント特徴量を用いて物体の姿勢を検出する．重心モーメント法は，検出に要する計算量は少ない．しかし，異なる形状で同一の二次モーメントを有する場合があり，物体の識別に制約が多い．

平面運動物体の検出法は，ロボットの目としてロボットビジョンへの適用が可能である．たとえば物体ハンドリングでは，ベルトコンベア等の平面運動で対象が運ばれることが多い．このような場合，二次元の視覚情報で対象の位置と姿勢を検出し，物体ハンドリングを行うことが可能である．また，物体ハンドリングを行う場合，物体の位置と姿勢を高速に計算し，ハンドリングを行うコントローラにその情報を伝える必要がある．したがって，物体ハンドリングにおいては，多様な対象物に対処でき検出時間が短い手法が望まれる．一方，物体ハンドリングにおいては，前処理として物体が重ならないよう分離するなど，物体の状態に制約をつけることが可能である．したがって，多くの物体ハンドリングでは，分離されたハンドリング物体の位置と姿勢を高速に求めることができれば十分である場合が多い．

本章では，片側ラドン変換を用いた新しいアルゴリズムを提案する．提案するアルゴリズムにより，分離されたハンドリング物体を識別し，その位置と姿勢を求めることができる．第 3.2 節では，片側ラドン変換の定義とその性質について述べる．第 3.3 節では，片



(a)　(b)

Fig. 3.1: Integral paths in Radon transform and in one-sided Radon transform

側ラドン変換を用いて，物体の位置と姿勢を検出するビジョンアルゴリズムを構成する．第3.4節では，平面運動検出アルゴリズムの検証について述べる．第3.5節では，平面運動物体の位置・姿勢を計測する実験を行い，ビジョンアルゴリズムを評価する．

3.2　片側ラドン変換

3.2.1　ラドン変換とその性質

前章では，ラドン変換と一次元位相限定相関を用いた検出法について述べた．しかし，前章の角度検出は，計算量が多いため，本章の姿勢検出には向かない．そこでハフ・フーリエ変換法[10]で用いられたラドン変換のρ軸側のパワースペクトルを用いた検出法を使用する．

ラドン変換$R(\rho,\theta)$のρに関するパワースペクトルは，次式によって与えられる．

$$F(f,\theta)=\left|\int_{-\infty}^{\infty}R(\rho,\theta)e^{-j2\pi f\rho}d\rho\right|^2 \tag{3.1}$$

$R_0(\rho,\theta)$，$R_3(\rho,\theta)$のρに関するパワースペクトルをそれぞれ$F_0(f,\theta)$，$F_3(f,\theta)$とすると，式(2.4)より，

$$F_0(f,\theta)\equiv F_3(f,\theta+\alpha),\quad \forall f,\theta \tag{3.2}$$

が得られる．これは，回転角度αがパワースペクトルF_0とF_3の1次元マッチングで計算できることを意味する．ただし，ラドン変換が$R(\rho,\theta)\equiv R(-\rho,\theta+\pi)$を満たすので，パワースペクトル$F(f,\theta)$は周期$\pi$の関数であり，$F(f,\theta)\equiv F(f,\theta+\pi)$が成り立つ．したがって，この手法で回転角度$\alpha$と$\alpha+\pi$を区別するには，追加的な処理が必要である．

また，物体の位置を2値化重心計算により求め，物体の姿勢は重心をラドン変換の座標原点としたラドン変換から求める手法が考えられる．この手法では，ρ=0に対応するラドン変換を比較して，物体の姿勢を計算する．ただし，ラドン変換においては，$R_0(0,\theta)\equiv R_0(0,\theta+\pi)$が成り立つため，角度$\alpha$と$\alpha+\pi$を区別するには$\rho\neq 0$の変換結果を調べなければならない．

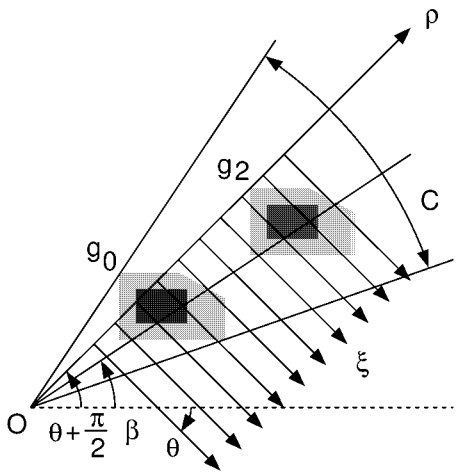


Fig. 3.2: Direction of projection axis ρ

3.2.2 片側ラドン変換とその性質

ラドン変換においては，積分路が Fig.3.1-(a) に示すように直線であるため，角度 θ と $\theta+\pi$ を区別することが困難である．そこで，積分路を直線から，Fig.3.1-(b) に示すように半直線に変更する．すなわち，

$$U[g](\rho,\theta)=\int_0^\infty g(\xi\cos\theta-\rho\sin\theta,\xi\sin\theta+\rho\cos\theta)\mathrm{d}\xi \tag{3.3}$$

とする．これを片側ラドン変換と呼ぶ.

原画像 g_0 と回転画像 g_1 の片側ラドン変換をそれぞれ $U_0(\rho,\theta)$，$U_1(\rho,\theta)$ で表すと，次式が成り立つ.

$$U_0(\rho,\theta)\equiv U_1(\rho,\theta+\alpha),\quad \forall\rho,\theta. \tag{3.4}$$

並進移動画像 g_2 の片側ラドン変換を $U_2(\rho,\theta)$ で表す．画像 g_0，g_2 のラドン変換 R_0，R_2 は，任意の θ に対して ρ 方向に $-d_0\sin(\theta-\beta)$ シフトすることにより互いに一致する．一方，片側ラドン変換 U_0，U_2 は，任意の θ に対して一致するとは限らない．たとえば，Fig.3.2 に示す ρ 軸に対して，片側ラドン変換 U_0，U_2 を計算する．このとき，積分路である半直線は，図に示す ξ 軸である．このとき，U_0 と U_2 は，ρ 方向にどのようにシフトさせても一致しない．そこで，ρ 方向のシフトで一致する角度 θ を求める．

座標原点から始まり，x 軸との角度が θ の半直線を $\ell(\theta)$ で表す．Fig.3.2 に示すように，角度 θ の集合 C を定義する．すなわち，任意の $\theta\in C$ においては，直線 $\ell(\theta)$ は，画像 g_0 の物体もしくは画像 g_2 の物体と交わる．任意の $\theta\notin C$ においては，直線は g_0 と g_2 のどちらの物体にも干渉しない．射影軸 ρ の方向が，集合 C に含まれる場合，射影軸 ρ の方向線が画像 g_0 と画像 g_2 の物体の同じ箇所を通るとき，片側ラドン変換 $U_0(\rho,\theta)$ と $U_2(\rho,\theta)$ は，ρ 方向に $-d_0\sin(\theta-\beta)$ シフトすることにより一致する．また，ρ 軸の方向が，画像の移動方向 β に一致するとき，片側ラドン変換 $U_0(\rho,\theta)$ と $U_2(\rho,\theta)$ は，ρ 方向に $-d_0$ シフトすることにより一致する．射影軸 ρ の方向は，角度 $\theta+\pi/2$ で与えられる．したがって，次

式が得られる．

$$U_0(\rho,\theta) \equiv U_2(\rho - d_0\sin(\theta-\beta),\theta), \quad \forall\rho, \quad \theta+\frac{\pi}{2} \notin C \ \text{ or } \ \theta+\frac{\pi}{2}=\beta. \tag{3.5}$$

座標原点を原画像の物体内部に選ぶ．このときすべての半直線 $\ell(\theta)$ は，原画像の物体と交わる．したがって，$C=[0,2\pi]$ が成り立つ．このとき，並進移動画像において，射影軸 ρ の方向線が座標原点を含む物体内部を通るとき，

$$U_0(\rho,\theta) \equiv U_2(\rho - d_0\sin(\theta-\beta),\theta), \quad \forall\rho, \quad \theta+\frac{\pi}{2}=\beta \tag{3.6}$$

が成り立つ．平面運動画像 g_3 の片側ラドン変換を $U_3(\rho,\theta)$ とすると，次式が成り立つ．

$$U_0(\rho,\theta-\alpha) \equiv U_3(\rho - d_0\sin(\theta-\beta),\theta), \quad \forall\rho, \quad \theta+\frac{\pi}{2}=\beta. \tag{3.7}$$

片側ラドン変換 $U(\rho,\theta)$ の ρ 方向のパワースペクトルを $\mathcal{F}(f,\theta)$ とする．片側ラドン変換 $U(\rho,\theta)$ は，一般に $U(\rho,\theta+\pi)$ と一致しないので，片側ラドン変換のパワースペクトル $\mathcal{F}(f,\theta)$ は，周期 2π の関数となる．片側ラドン変換 $U_0(\rho,\theta)$，$U_3(\rho,\theta)$ のパワースペクトルをそれぞれ $\mathcal{F}_0(f,\theta)$，$\mathcal{F}_3(f,\theta)$ とすると，式（3.7）より次式が得られる．

$$\mathcal{F}_0(f,\theta-\alpha) \equiv \mathcal{F}_3(f,\theta), \quad \forall f, \quad \theta+\frac{\pi}{2}=\beta. \tag{3.8}$$

3.3 片側ラドン変換を用いた平面運動物体検出アルゴリズム

本節では，片側ラドン変換の性質に基づき，平面運動検出アルゴリズムを構築する．参照画像を g_{sample}，入力画像を g_{input} とする．また，これらの画像の片側ラドン変換をそれぞれ $U_{sample}(\rho,\theta)$ と $U_{input}(\rho,\theta)$ とし，片側ラドン変換の ρ 方向のパワースペクトルをそれぞれ $\mathcal{F}_{sample}(\rho,\theta)$ と $\mathcal{F}_{input}(\rho,\theta)$ とする．

まず，式（3.7）と式（3.8）を用いて，入力画像の物体の位置と姿勢を計算する．回転角度 α と並進移動の角度 β を検出するために，パワースペクトル $\mathcal{F}_{sample}(\rho,\theta)$ と $\mathcal{F}_{input}(\rho,\theta)$ を比較する．そして，並進移動距離 d_0 を求めるために片側ラドン変換 $U_{sample}(\rho,\theta)$ と $U_{input}(\rho,\theta)$ を比較する．以上のアルゴリズムを次にまとめる．

Algorithm 1

Step 1 $U_{sample}(\rho,\theta)$ と $U_{input}(\rho,\theta)$ を計算する．

Step 2 $\mathcal{F}_{sample}(f,\theta)$ と $\mathcal{F}_{input}(f,\theta)$ を計算する．

Step 3 $\| \mathcal{F}_{sample}(f,\theta_{sample}) - \mathcal{F}_{input}(f,\theta_{input}) \|$ の最小値を求め，
θ_{sample} と θ_{input} を検出する．

Step 4 $\alpha = \theta_{input} - \theta_{sample}$
$\beta = \theta_{input} + \pi/2$
を求める．

Step 5 $U_{sample}(\rho, \theta_{sample}) \equiv U_{input}(\rho + d_0, \theta_{input})$
$\forall\rho$ を満たす d_0 を求める．

片側ラドン変換のパワースペクトルが，2π 周期であることから，パワースペクトルを計算することで回転角度を計算することができる．

並進移動を検出する別の方法として，物体の重心を用いる方法がある．物体と背景を分離でき，物体の重なりがないと仮定すると，物体の重心を計算することにより，物体の並進移動を検出することができる．参照画像と入力画像の物体の重心を，それぞれ G_{sample}, G_{input} とする．物体の姿勢は，式（3.8）を用いることで検出できる．以上のアルゴリズムを次にまとめる．

Algorithm 2

Step 1 重心 G_{sample} と G_{input} を計算する．

Step 2 それぞれの重心を原点とし，
$U_{sample}(\rho, \theta)$ と $U_{input}(\rho, \theta)$ を計算する．

Step 3 $\mathcal{F}_{sample}(f, \theta)$ と $\mathcal{F}_{input}(f, \theta)$ を計算する．

Step 4 $\mathcal{F}_{sample}(f, \theta - \alpha) \equiv \mathcal{F}_{input}(f, \theta) \quad \forall f, \theta$
を満たす α を求める．

次に，重心計算から位置を求め，重心を原点とし $\rho = 0$ に対応する片側ラドン変換を比較することにより，姿勢を求める．重心計算によって位置が求められ，物体の回転角度 α は，式 (3.4) を用いることで求められる．このように，重心計算後に片側ラドン変換を直接比較することで，物体の回転角度を検出することができる．以上のアルゴリズムを次にまとめる．

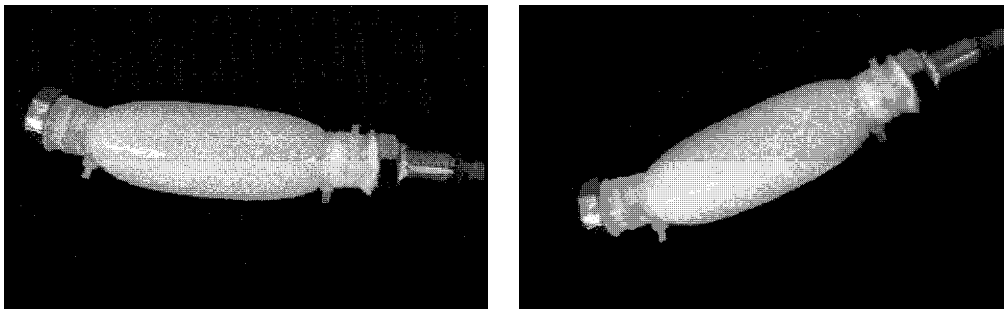
Algorithm 3

Step 1 重心 G_{sample} と G_{input} を計算する．

Step 2 それぞれの重心を原点とし，$U_{sample}(\rho, \theta)$ と $U_{input}(\rho, \theta)$ を計算する．

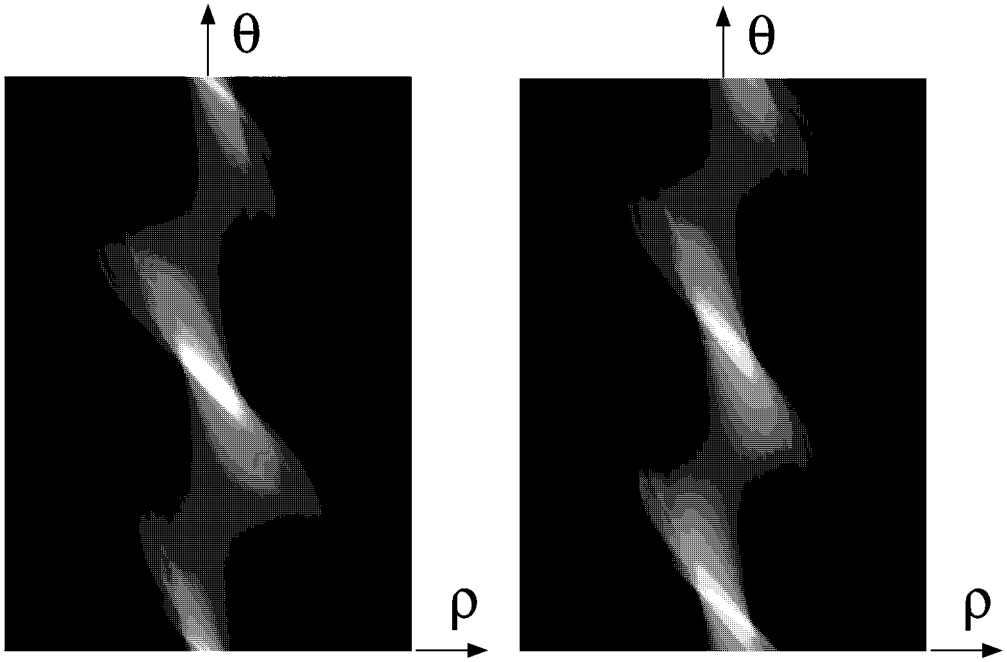
Step 3 $U_{sample}(0, \theta - \alpha) \equiv U_{input}(0, \theta) \quad \forall\theta$
を満たす α を求める．

片側ラドン変換 $U(0, \theta)$ は 2π 周期の関数であるため，片側ラドン変換 $U_0(0, \theta)$ と $U_0(0, \theta+\pi)$ の値は，一般に異なる．このことは，2 つの片側ラドン変換 $U_{sample}(0, \theta)$ と $U_{input}(0, \theta)$ のマッチングで角度 α と $\alpha + \pi$ を区別できることを意味する．



(a) (b)

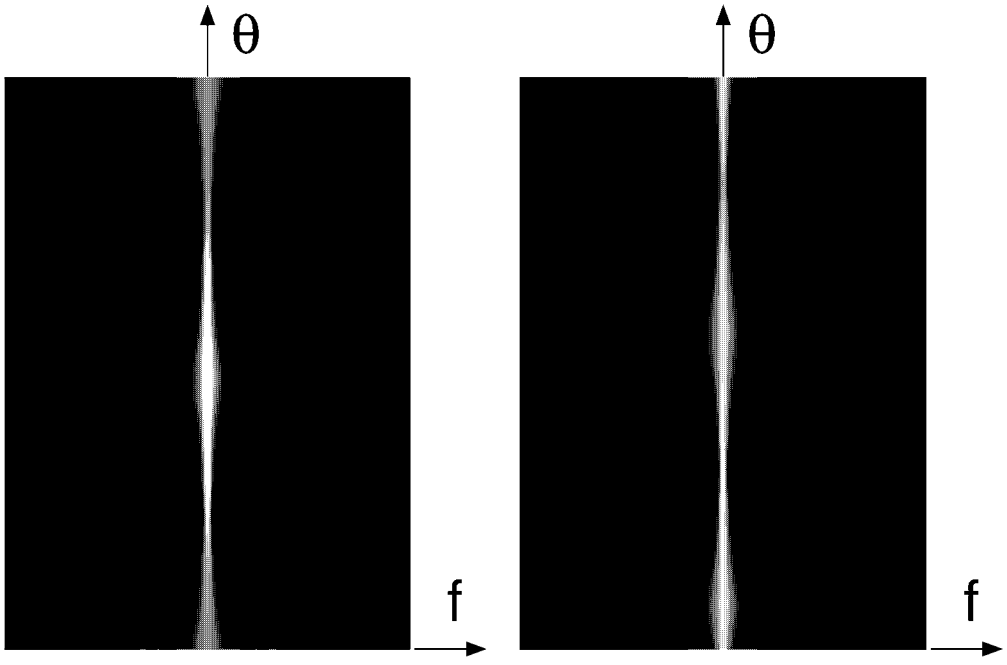
Fig. 3.3: Original image and moved image



(a) (b)

Fig. 3.4: Computed one-sided Radon Transform

次に，これらアルゴリズムの片側ラドン変換と姿勢マッチングに要する計算量について比較する．画像のサイズを $M \times M$ [pixel] とし，角度 θ のステップ数を N とする．距離 ρ のステップ数を M に一致させ，また片側ラドン変換の積分距離を M に一致させる．このとき，フーリエ変換の周波数 f のステップ数は M である．片側ラドン変換の計算量は，Algorithm 1 と Algorithm 2 では M^2N となり，Algorithm 3 では MN になる．次に姿勢マッチングの計算量について述べる．Algorithm 1 では，ノルム $|| \mathcal{F}_{sample}(f, \theta_{sample}) - \mathcal{F}_{input}(f, \theta_{input}) ||$ の計算に，計算量 M を要する．各 θ_{sample}，θ_{input} に対してノルムを計算するので，全体の計算量は MN^2 である．Algorithm 2 では，$\mathcal{F}_{sample}(f, \theta - \alpha)$ と $\mathcal{F}_{input}(f, \theta)$ の比較に，計算量 MN を要する．各 α に対して比較を行なうので，全体の計算量は MN^2 である．Algorithm 3 では，$U_{sample}(0, \theta - \alpha)$ と $U_{input}(0, \theta)$ の比較に計算量 N を要する．各 α に対して比較を行なうので，全体の計算量は N^2 である．結局，Algorithm 1 と Algorithm 2 の計算量は，$M^2N + MN^2$，Algorithm 3 の計算量は，$MN + N^2$ である．



(a) (b)

Fig. 3.5: Computed power spectrums of one-sided Radon Transform

Table 3.1: Detected position and orientation for Fig.3.3

	α°	x[pixel]	y[pixel]
true values	30	20	10
Algorithm1	30	19.6	9.99
Algorithm2	30	13.6	18.9
Algorithm3	30	13.6	18.9

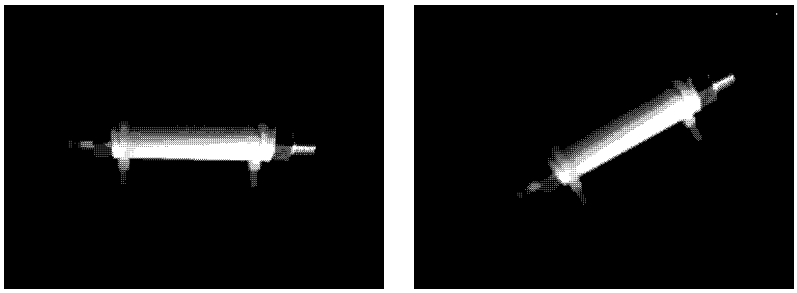
3.4 平面運動物体検出アルゴリズムの検証

第 3 章で提案した三つのアルゴリズムを PC 上に実装し，評価する．PC は自作 PC/AT 互換機（Pentium III 800MHz，メモリ 256MB 搭載），コンパイラは GCC を使用し，最適化オプションは"-O2"である．この評価に使用する原画像を Fig.3.3-(a) に示す．Fig.3.3-(a) の中央を原点として 30° 回転し，x 方向に 20[pixel]，y 方向に 10[pixel] 移動させた画像を Fig.3.3-(b) に示す．原画像と移動画像の片側ラドン変換をそれぞれ Fig.3.4-(a),(b) に示す．また，パワースペクトルをそれぞれ Fig.3.5-(a),(b) に示す．また，Algorithm 1 と Algorithm 2 での片側ラドン変換の座標原点は，画像中心に設定している．距離 ρ のステップ数 M は，Algorithm 1，Algorithm 2 ともに 256 である．角度 θ のステップ数 N は，Algorithm 1，Algorithm 2，Algorithm 3 ともに 360 である．各アルゴリズムにより検出された回転角度と並進移動量を Table 3.1 に示す．

Algorithm 2 と Algorithm 3 において重心を正確に計算できない理由として，1）画像内のノイズ，2）物体が画像内に収まらないという点が挙げられる．そこで，ノイズが無

Table 3.2: Detected position and orientation for Fig.3.6

	α°	x[pixel]	y[pixel]
true values	30	20	10
Algorithm1	30	19.6	9.99
Algorithm2	30	20.1	10.1
Algorithm3	30	20.1	10.1



(a)　　(b)

Fig. 3.6: Original image and moved image within frame

く，物体が画像内に収まる画像を用意し，同様の実験を行った．このとき使用した画像を Fig.3.6 に示す．この画像を用いて検出された回転角度と並進移動量を Table 3.2 に示す．これらの実験結果より，Algorithm 2 と Algorithm 3 は，物体が画像内に収まっておりかつ重心計算に影響を及ぼすノイズを含まないようにしなければ，正確な物体検出ができないことがわかる．Algorithm 1 においては，画像全体の情報を含む片側ラドン変換やそのパワースペクトルの比較により，物体の位置と姿勢を求めるため，ノイズの影響を受けにくいと考えられる．各アルゴリズムの計算時間を Table 3.3 に示す．ただし，原画像の片側ラドン変換とパワースペクトルの計算に費やした時間は除いてある．片側ラドン変換の計算時間が，全体の計算時間の大部分を占めることがわかる．

3.5　ビデオフレームレート運動検出アルゴリズムの構成

3.5.1　平面運動計測システムの構成

前章で構成した三つのアルゴリズムの内，Algorithm 3 は，PC 上でビデオフレームレートで実行できる．また Algorithm 3 は，穴や凹部を有する形状に対応でき，同じ形状で違う模様の物体を識別できるため，重心モーメント法やフーリエ記述子法と比較して対応できる物体の範囲が広いと考えられる．そこで本章では，Algorithm 3 を基礎として，平面運動計測システムを構築し，その性能を評価する．

PC の環境は，第 4 章と同じであり，ビデオキャプチャーボード I O DATA 製 GV-VCP2/PCI を通して CCD カメラから画像を取り込む．画像取り込み API は Video4Linux

Table 3.3: Computed time of algorithms

Algorithm 1 [msec]	
one-sided Radon transform	5600
power spectrum	32
matching	432
total	6100

Algorithm 2 [msec]	
gravity center	1.3
one-sided Radon transform	5200
power spectrum	30
matching	634
total	5900

Algorithm 3 [msec]	
gravity center	1.6
one-sided Radon transform	27
matching	1.3
total	29.9

を使用する．CCD カメラから取り込む画像は，幅 320[pixel]，高さ 240[pixel] のグレースケール画像である．なお，離散画像から連続画像への補間には，共一次内挿法を用いる．

次に，入力画像の生成について述べる．あらかじめ，背景画像 $g_{back}(x,y)$ を取得しておく．カメラからの画像 $g_{camera}(x,y)$ と背景画像との差分の絶対値を入力画像とする．すなわち，

$$g_{input}(x,y) = \left| g_{camera}(x,y) - g_{back}(x,y) \right| \tag{3.9}$$

これにより，移動する物体の画素値のみを抽出することができる．なお，参照画像に対しても，同様の処理をあらかじめ行い，$g_{sample}(x,y)$ を取得しておく．重心の座標値は，重心の計算式で得られた値の整数値とする．すなわち，重心は格子点上にある．Step 2 における片側ラドン変換の計算においては，線積分の上限を ξ_d とし，台形則で積分値を求める．Step 3 におけるマッチングにおいては，相関関数

$$S(\tau) = \int_0^{2\pi} \left\{ U_{sample}(0, \theta+\tau) - U_{input}(0,\theta) \right\}^2 \mathrm{d}\theta \tag{3.10}$$

を計算し，$S(\tau)$ の最小値 $S_{min} = S(\tau_{min})$ を求める．最小値 S_{min} が閾値以内であれば，入力画像はパターン画像の形状に一致すると判定する．このとき τ_{min} が物体の姿勢である．式 (3.10) の値が閾値を超えた場合，別の参照画像を用いて式（3.10）を計算し，マッチングを行う．

3.5.2 処理時間

本報告では，物体を構成する画素の違いによる検出比較を行うため，物体が物体の重心からの片側ラドン変換の積分距離 $\xi_d = 32$[pixel] に収まる場合と $\xi_d = 64$[pixel] に収まる場

Table 3.4: Computation time for different integral intervals

	$\xi_d = 32$	$\xi_d = 64$
separation of object and background	1.65[msec]	1.20[msec]
gravity center	1.30[msec]	1.50[msec]
one-sided Radon transform	7.88[msec]	15.44[msec]
matching	1.34[msec]	1.36[msec]
total	12.02[msec]	19.50[msec]

Table 3.5: Standard deviation

	x position [pixel]	y position [pixel]	angle [degree]
$\xi_d = 32$	0.82	0.29	3.48
$\xi_d = 64$	0.44	0.30	1.09

合に対して，位置と姿勢の計測結果，処理時間，認識率を評価する．片側ラドン変換の θ のステップ数は，512 に設定している．提案するビジョンアルゴリズム Algorithm 3 に含まれる処理に要する時間を，Table 3.4 に示す．物体と背景の分離とは，式（3.9）により背景を除去する処理である．パターンマッチングとは，式（3.10）を用いて，物体を識別し，回転角を求める処理である．各処理を 100 回実行し，その平均値を表に示してある．この結果より，片側ラドン変換の処理時間が全体の処理の大半を占めており，片側ラドン変換の積分の区間距離が同じであれば，片側ラドン変換の処理時間は積分距離に比例していることがわかる．このことから，全体の処理時間を短くするには，片側ラドン変換の高速化による効果が一番大きいことがわかる．NTSC 方式のビデオ信号を入力として使用するときには，物体の種類，位置・姿勢の計算を 33[msec] 以内で行う必要がある．Table 3.4 より，物体の位置・姿勢検出処理は，ビデオフレームレートで可能である．

3.5.3　平面運動計測

構築した平面運動計測システムを評価するため，エアテーブル上で平面運動計測の実験を行った．エアテーブルと物体の間の摩擦は無視できるため，エアテーブル上で物体は等速度・等角速度運動を行う．このことから提案するアルゴリズムの計測結果が，等速度・等角速度を示せば提案するアルゴリズムの位置・姿勢計測の正当性を証明することができる．そこで，アルゴリズムによる位置・姿勢の計測結果とその結果から最小二乗法によって求めた式の比較を行う．片側ラドン変換のサイズを $\xi_d = 32$ のときで計測した結果を Fig.3.7 と Fig.3.8 に示し，$\xi_d = 64$ のときで計測した結果を Fig.3.9 と Fig.3.10 に示す．実験に用いた物体では，画像中心と力学的な重心が一致しており，画像重心は直線軌道を描くと考えられる．計測結果の値と最小二乗法の式の値によって求めた標準偏差を Table

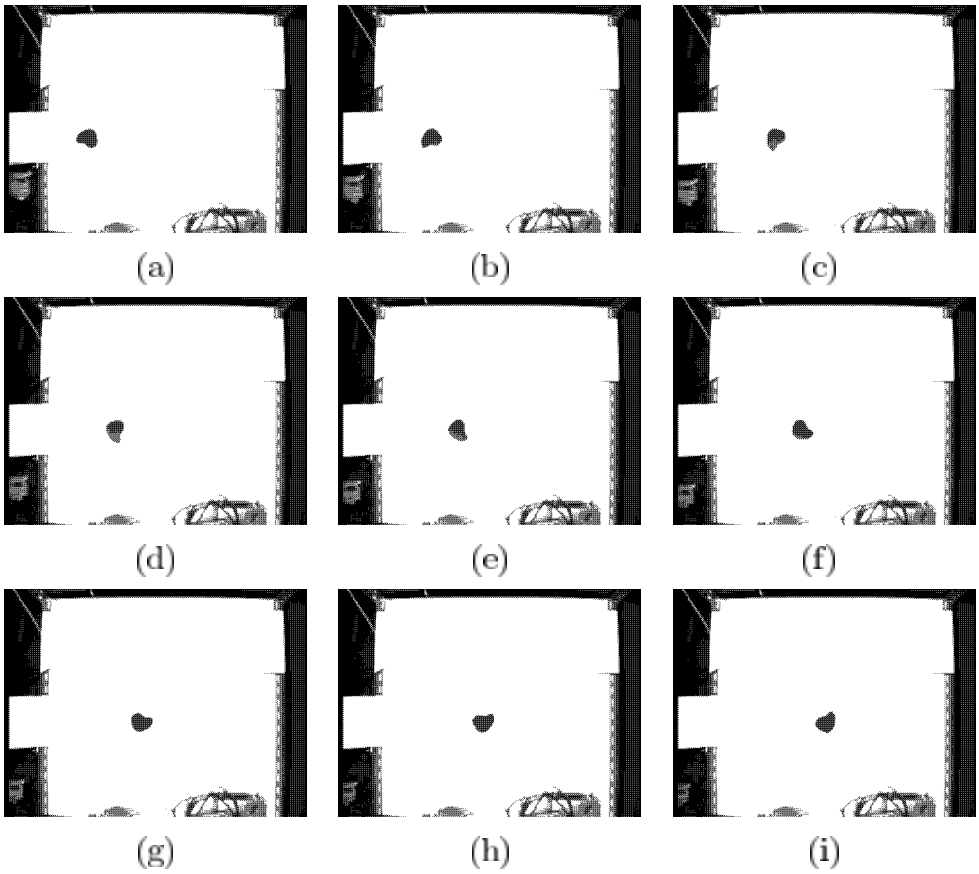
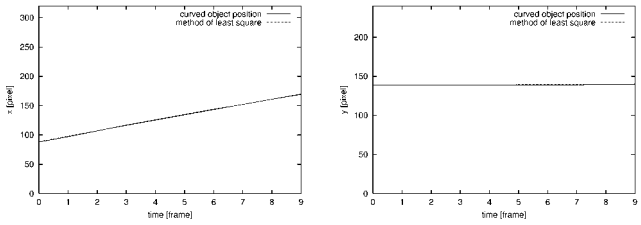


Fig. 3.7: Successive images in planar motion of curved object ($\xi_d = 32$)

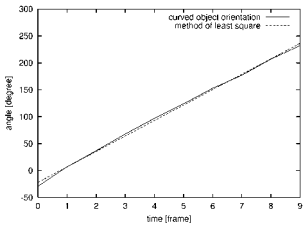
3.5 に示す．これらの結果より，計測結果と最小二乗法によって求めた式はほぼ一致しており，標準偏差のばらつきも小さいことがわかる．したがって，提案するアルゴリズムで計測した位置・姿勢の速度・角速度は一定であると考えられる．このことからビジョンアルゴリズムの位置・姿勢計測の正当性を証明することができた．なお，計測精度は $\xi_d = 64$ のほうが高い．

3.5.4 平面運動物体の認識評価

正三角形，正方形，曲線形，円形の四つの物体を使用して，平面運動する物体の認識実験を行った．物体がその重心から $\xi_d = 32$ に収まるときの物体画像を Fig.3.11 に示し，$\xi_d = 64$ に収まるときの物体画像を Fig.3.12 に示す．この実験では，正三角形，正方形，曲線形の三つをテンプレートとし，四つの物体を単体で平面運動させたときの物体認識を評価する．パターンマッチングでは，式（3.10）で与えられる相関関数の最小値を評価し，閾値以内であればテンプレートの物体であると判断する．正三角形をテンプレートとして，四つの物体の平面運動を計測したときの式（3.10）の最小値を Fig.3.13 に示す．また，正方形をテンプレートにしたときの結果を Fig.3.14 に示し，曲線形をテンプレートにしたときの結果を Fig.3.15 に示す．Fig.3.13，Fig.3.14，Fig.3.15 の横軸は式（3.10）の値を表し，縦軸は頻度を表している．



(a) position



(b) orientation

Fig. 3.8: Computed position and orientation of curved object ($\xi_d = 32$)

Table 3.6: Detection of objects

	$\xi_d = 32$	$\xi_d = 64$
triangle	100[%]	100[%]
square	100[%]	100[%]
curved	95[%]	100[%]
circle	98[%]	94[%]

同じ物体に対する相関関数の最小値が小さく，異なる物体に対する相関関数の最小値が大きいほど，物体を識別する閾値を設定しやすい．このことから，Fig.3.13，Fig.3.14 では，物体に含まれる画素数が多い $\xi_d = 64$ の方が閾値を設定しやすいことがわかる．しかし，Fig.3.15 に示すように，$\xi_d = 32$ のときでは同じ物体に対する値と異なる物体に対する値が重なる部分があり，$\xi_d = 64$ のときでは同じ物体に対する値と異なる物体に対する値の差が小さいため，閾値を設定が難しい．このようになった原因として，曲線形が正三角形と円形に類似していることが考えられる．

次に Fig.3.13，Fig.3.14，Fig.3.15 の結果をもとにテンプレートごとの閾値を設定し，平面運動する物体の認識率を評価する．物体の識別においては，正三角形，正方形，曲線形の順にマッチングを行う．いずれともマッチングしないときは，円形と判断する．平面運動物体の認識結果を表 3.6 に示す．表 3.6 における曲線形の認識失敗はすべて円形と判断したものであり，円形の認識失敗はすべて曲線形と判断したものである．これは，Algorithm 3 が $\rho = 0$ に対応する片側ラドン変換のみを用いることに起因する．認識の精度を上げるためには，Algorithm 1 あるいは Algorithm 2 を用いる必要がある．また，物体認識に要した時間は，$\xi_d = 32$ のとき最大で約 20[msec]，$\xi_d = 64$ のとき最大で約 29[msec] である．

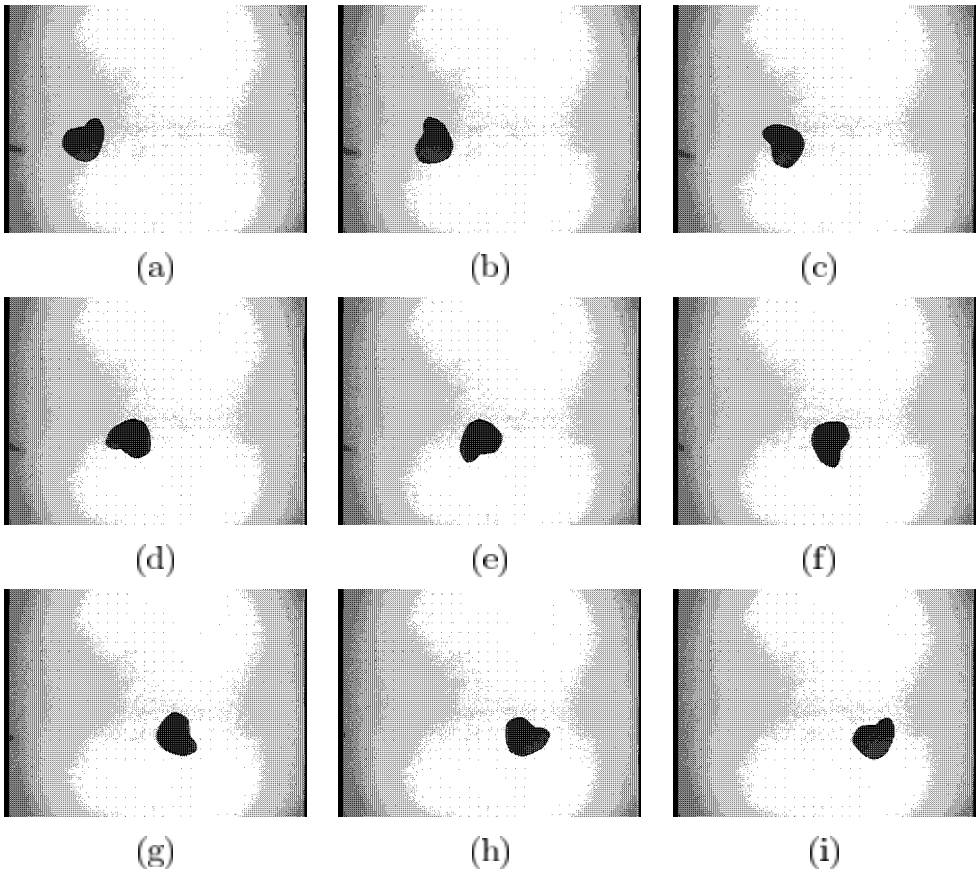


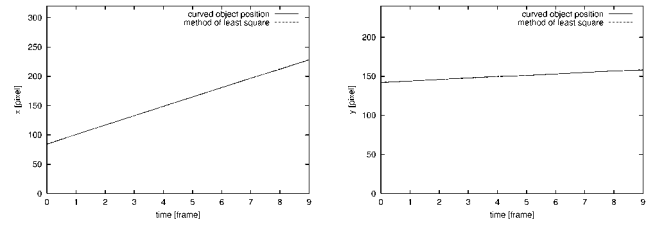
Fig. 3.9: Successive images in planar motion of curved object ($\xi_d = 64$)

3.5.5 模様つき物体の平面運動計測

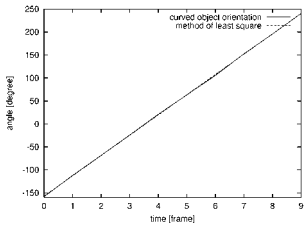
片側ラドン変換は画素値の線積分であるため，表面上の模様を用いて物体の位置・姿勢を検出できる．そこで円形に模様をつけ，$\xi_d = 64$ と設定して平面運動計測を行った．計測時の結果を Fig.3.16 と Fig.3.17 に示す．今回の実験に用いた物体では，画像重心と力学的な重心は一致しないが，その違いは小さい．したがって，画像重心は，ほぼ直線軌道を描くと考えられる．Fig.3.17 に示すように，物体の衝突前と衝突後に物体は，等速等角速度運動を行っている．したがって，平面運動計測は正当であると判断する．以上のことから，模様つき物体の位置・姿勢の検出にも有効であると考えられる．なお，画像重心と力学的な重心のずれが大きいときには，画像重心から力学的な重心を計算する必要がある．以上に示すように，物体と背景を分離できれば，平面運動物体の衝突前と衝突後の位置と姿勢を検出することが可能である．

3.6 結言

本論文では，片側ラドン変換を用いて平面運動物体の位置と姿勢を検出する手法を提案した．まず，片側ラドン変換とそのパワースペクトルの性質を明らかにし，片側ラドン変換とパワースペクトルを計算することにより，平面運動物体の位置と姿勢を検出するア



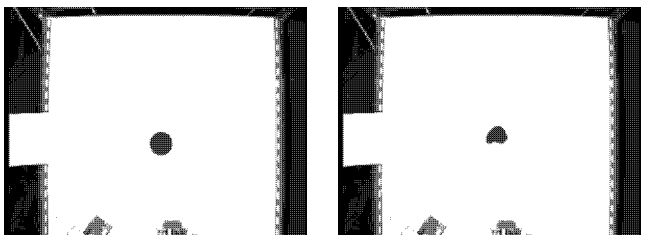
(a) position



(b) orientation

Fig. 3.10: Computed position and orientation of curved object ($\xi_d = 64$)

(a) triangle　(b) square

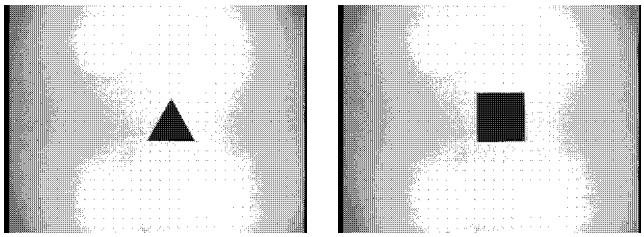


(c) circle　(c) curved

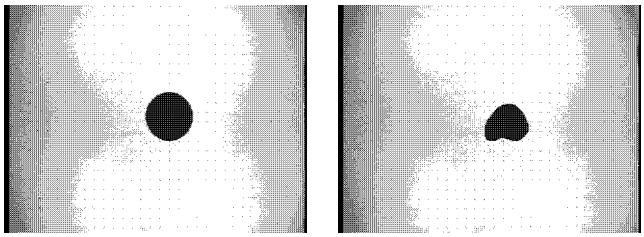
Fig. 3.11: Objects in experiment ($\xi_d = 32$)

ルゴリズムを構成した．このアルゴリズムは，平面運動物体の位置と姿勢を検出することができるが，計算時間が多く，ビデオフレームレートでは実行できないことがわかった．次に，重心と片側ラドン変換を計算することにより，平面運動物体の位置と姿勢を検出するアルゴリズムを構成した．このアルゴリズムは，物体が分離されている必要があるものの，ビデオフレームレートで実行できることがわかった．そこで，このアルゴリズムを基礎として，ビデオフレームレートで平面運動物体の位置と姿勢を検出し，物体の種類を識別するシステムを構築した．エアテーブル上で平面運動物体の位置と姿勢の検出を行い，構築したシステムが精度良く位置と姿勢を検出できることを示した．

本章で提案するアルゴリズムにより，ソフトウェアによる手法でビデオフレームレート

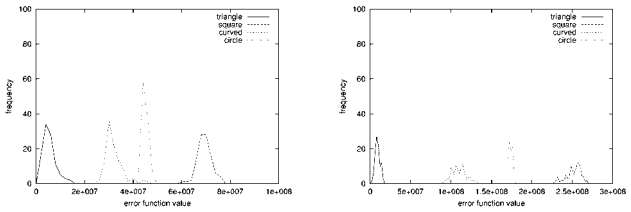


(a) triangle (b) square



(c) circle (c) curved

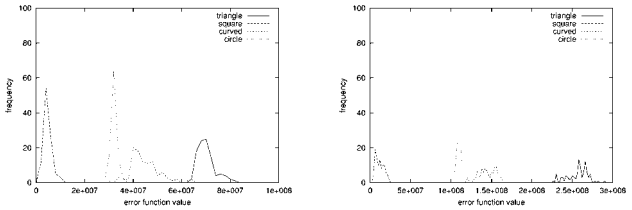
Fig. 3.12: Objects in experiment ($\xi_d = 64$)



(a) $\xi_d = 32$ (b) $\xi_d = 64$

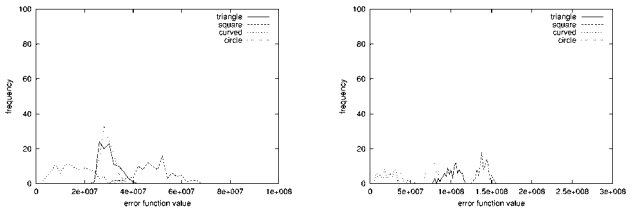
Fig. 3.13: Minimum value of error function (triangle)

での検出が可能となった．第 4 章では，本章のアルゴリズムをもとにハードウェアによる実装を検討する．



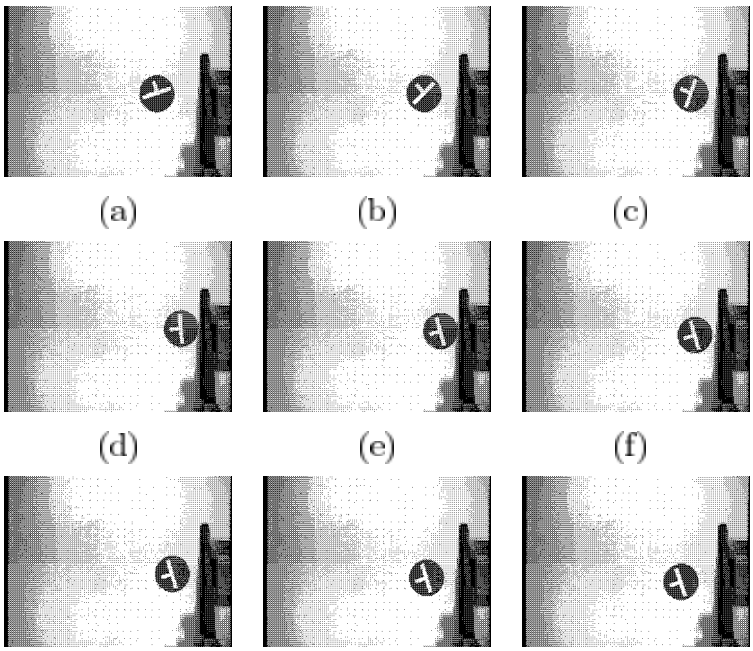
(a) $\xi_d = 32$　　(b) $\xi_d = 64$

Fig. 3.14: Minimum value of error function (square)



(a) $\xi_d = 32$　　(b) $\xi_d = 64$

Fig. 3.15: Minimum value of error function (curved)

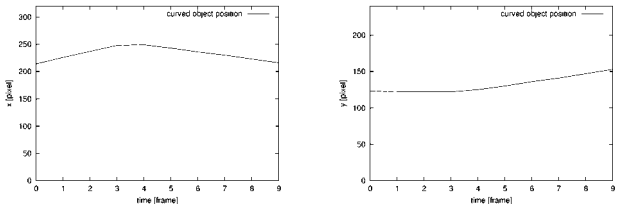


(a)　(b)　(c)

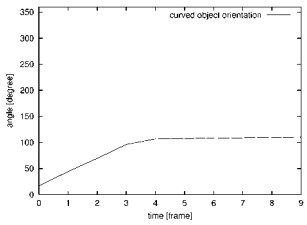
(d)　(e)　(f)

(g)　(h)　(i)

Fig. 3.16: Successive images in planar motion of circle object ($\xi_d = 64$)



(a) position



(b) orientation

Fig. 3.17: Computed position and orientation of circle object ($\xi_d = 64$)

第4章　FPGAベースリアルタイムビジョン

4.1　はじめに

本章では，論理回路の再構成が可能なFPGA (Field Programmable Gate Array) を基にリアルタイムビジョンシステムを構築し，FPGAベースリアルタイムビジョンシステムの性能を，実験的に評価する．

現在のビジョンシステムは，大きく二つの手法に分類できる．ひとつは，ソフトウェアによる手法で，もうひとつはASICによる手法である．ソフトウェアによる手法では，ビジョンアルゴリズムはプログラムの形で記述され，汎用のCPU上で実行される．この手法では，様々なアルゴリズムを，低いコストで実装することができる．一方，アルゴリズムによっては，多くの計算時間を要し，リアルタイムで実行することができない．ASICによる手法では，ビジョンアルゴリズムは論理回路の形で記述され，アルゴリズムに特化して設計，製造されたLSI上で実行される．すでに，フーリエ変換[16]，ハフ変換[17]，正規化相関[18, 19]，ステレオビジョンアルゴリズム[20]など，多くのアルゴリズムがASIC上に実装されている．また，論理回路とアナログセンサ回路から成るLSIも試作されている[21, 22]．ASICによる手法では，高速な演算が可能である反面，ASIC上の回路設計と製造に，多大の労力とコストを要する．一般に，万単位以上のマーケットが見込めないと，ASICを設計，製造するメリットはないと言われる．結局，ソフトウェアによる手法は，高いフレキシビリティを持つがリアルタイム性に劣り，ASICによる手法はリアルタイム性に優れるがフレキシビリティとコストパフォーマンスに劣る．

フレキシビリティとリアルタイム性とのこのようなジレンマを解決するために，本論文ではFPGAベースビジョンシステムを構築する．FPGAとは，ユーザが論理回路を書き込み，動作させることが可能なLSIである．ユーザは，ビジョンアルゴリズムに特化して設計された論理回路をFPGA上に実装し，FPGA上で回路を動作させることにより，ビジョンアルゴリズムを実行することができる．ビジョンアルゴリズムは論理回路の形で実行されるので，リアルタイム性は確保される．論理回路を再度設計し，FPGAに実装することができるので，フレキシビリティも確保される．すでに，いくつかのビジョンアルゴリズムがFPGAに実装されている．たとえば，二次元離散コサイン変換[23]，Parzen Window法による画像識別[24]，医療診断における集中度フィルタ[25]，畳み込みによる画像復元[26]，CORDICを用いたハフ変換[27]が実装されている．しかし，実装例では，フィルタリングなど局所演算に基づくビジョンアルゴリズムが多く，二次元フーリエ変換やハフ変換に代表される大域演算に基づくビジョンアルゴリズムの実装例は少ない．その原因として，大域演算を含むビジョンアルゴリズムを実行する論理回路を，ハードウェア

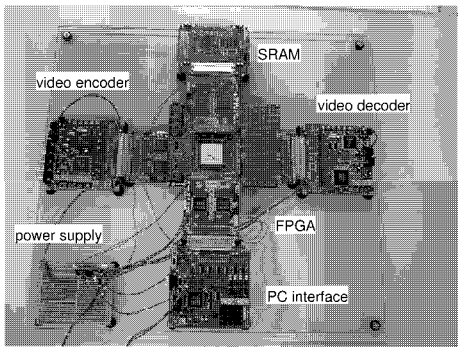


Fig. 4.1: Prototype of FPGA-based vision system

記述言語を用いて設計することが，容易ではない点が挙げられる．大域演算を含むビジョンアルゴリズムの回路設計においては，大域演算の結果や最終的な結果を画像で評価しながら，デバッグを進める必要がある．ハードウェア記述言語を用いた設計では，このようなデバッグに手間と時間を要するため，大域演算に基づくビジョンアルゴリズムの実装が困難となっている．

以上の考察から，本章では，FPGA ベースリアルタイムビジョンシステムを構築するとともに，C 言語ベースの回路設計を用いてビジョンアルゴリズムの回路設計を試みる．まず，構築した FPGA ベースビジョンシステムのハードウェアを紹介する．次に，FPGA ビジョンシステムに三つのアルゴリズム，画像重心の計算，放射状射影による姿勢の検出，ならびにハフ変換の計算を実装する．最後に，FPGA ベースビジョンシステムで，これらのアルゴリズムを実行した結果を述べる．

4.2　FPGA ベースビジョンシステム

4.2.1　ハードウェア構成

本節では，FPGA を用いたビジョンシステムについて述べる．物体操作や人間識別など多くのアプリケーションにおいてビジョンシステムは，連続する一連の画像をリアルタイムで処理することが要求される．さらに，様々な使用環境に適合できるフレキシビリティを有することが求められる．多くのビジョンアルゴリズムは並列性が高く，並列処理を実現することにより計算時間を短縮することが期待できる．一方，FPGA 上に論理回路を並列に配置することにより，並列処理を容易に実現できる．したがって，FPGA ベースビジョンシステムに並列性の高いビジョンアルゴリズムを実装することにより，リアルタイム性とフレキシビリティの双方を達成することができる．

Fig. 4.1 に FPGA ベースビジョンシステムのプロトタイプを示す．このプロトタイプは，FPGA ボード，ビデオデコーダボード，ビデオエンコーダボード，SRAM ボード，PC インタフェースから構成される．FPGA ボードは，エスケーエレクトロニクス社製 CM69 であり，FPGA としてザイリンクス社製 Vertex 2000-E が搭載されている．この FPGA 上には，200 万ゲート相当の論理回路を構築できる．ビデオデコーダボード，エンコーダボードは，三菱電機マイコン機器ソフトウエア社製 MU200-VDEC，MU200-VENC であ

り，ビデオエンコード，デコードチップとして Brooktree 社製 Bt812 と Bt856 が使われている．SRAM ボード MU200-SRAM は，1MB のメモリを提供する．NTSC 信号で送られる CCD カメラからの連続画像は，ビデオデコーダボードでデジタル信号に変換され，FPGA 上に実装される画像処理回路に送られる．画像処理回路は，必要ならば SRAM を記憶回路として使いながら，画像処理を進める．画像処理回路は PC 上で設計される．設計された回路は，PC インターフェース MU200-EX40 を通って，FPGA 上にダウンロードされる．

4.2.2 C 言語ベース回路設計

ビジョンアルゴリズムの回路設計においては，ハードウェア記述言語 (HDL) の代わりに，C/C++言語ベースの回路設計を導入する．従来より，多くのビジョンアルゴリズムが，C/C++言語を用いて開発されている．したがって，C/C++言語を用いた回路設計を導入することにより，アルゴリズムの開発と論理回路の設計を，シームレスに遂行することができる．また，ビジョンシステムの開発では，設計と評価を繰り返すことが多く，回路規模が大きいため論理合成に時間を要する．C/C++言語による回路設計では，C/C++言語の段階で設計を評価することができるため，論理合成の回数を減らし，開発時間を短縮することが期待できる．以上の理由から，C/C++言語を用いた論理回路設計を導入する．

本論文では，C/C++で記述したソースを HDL へ変換するソフトである SystemCompiler を使用する．SystemCompiler では，論理回路の記述に CycleC を用いる．CycleC は，C/C++のサブセットである．CycleC で設計した論理回路は，PC 上でコンパイル，実行することができるので，論理合成することなく論理回路を検証することができる．さらに，テストプログラムやテストパターンを PC 上で容易に構成できるため，効率的な回路の検証が可能である．以上の点より，SystemCompiler を導入することにより，設計した論理回路を短時間で検証することができる．CycleC による記述は，ハードウェア記述言語 VerilogHDL あるいは VHDL に変換される．ハードウェア記述言語に変換された設計を論理合成し，FPGA 上に実装する．以上のように，C/C++言語ベースの論理設計を導入することにより，ビジョンアルゴリズムを短期間で FPGA 上に実装することができる．Table 4.1 に，CycleC によるセレクタの記述例を示す．表に示すように，C/C++言語のクラスとメソッドを用いて，論理回路を記述する．

ビデオデコーダボード，エンコーダボード，SRAM ボードとのインターフェイスは，あらかじめ HDL で記述されている．ビジョンアルゴリズムを CycleC で記述し，SystemCompiler で HDL に変換した後に，インターフェイスの HDL 記述と接続する．この手法により設計者は，ビジョンアルゴリズムそのものの論理回路記述に専念することができ，効率の良い開発が期待できる．

Table 4.1: Description of selector in SystemCompiler

```
class Selecter {
public:
  uint1 reg;
  Selecter ( void ){ reg = 0;}
  void run ( uint1, uint1,
       uint1, uint1, uint1, uint1& );
};

void Selecter::run(uint1 clk, uint1 rst,
     uint1 a, uint1 b, uint1 s, uint1&y)
{
  uint1 temp = s ? a : b;
  if ( infer_clock(clk) ||
       infer_reset(!rst) ) {
    if (!rst) { reg = 0; }
    else { reg = temp; }
    y = reg;
  }
}
```

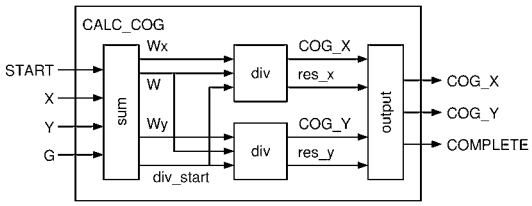


Fig. 4.2: Circuit to compute gravity center

4.3　ビジョンアルゴリズムの FPGA 実装

本節では，三つのビジョンアルゴリズムを，FPGA 上に実装する．すなわち，画像重心の計算，放射状射影による姿勢の検出 [28]，ならびにハフ変換の計算である．

4.3.1　画像重心の計算

まず，画像重心の計算を FPGA 上に実装する．計算アルゴリズムを簡単に説明する．グレースケール画像の幅を W，高さを H，格子点 (x, y) におけるピクセル値を $g(x, y)$ とす

る．画像重心の座標は，

$$\begin{bmatrix} c_x \\ c_y \end{bmatrix} = \begin{bmatrix} W_x/W \\ W_y/W \end{bmatrix} \tag{4.1}$$

ただし

$$\begin{aligned} \begin{bmatrix} W_x \\ W_y \end{bmatrix} &= \sum_{x=0}^{W-1}\sum_{y=0}^{H-1} \begin{bmatrix} x \\ y \end{bmatrix} g(x,y), \\ W &= \sum_{x=0}^{W-1}\sum_{y=0}^{H-1} g(x,y) \end{aligned} \tag{4.2}$$

で与えられる．上式に示すように，画像重心の計算は，三つの総和と二つの除算から成る．総和 W_x, W_y, W の計算は，画像のキャプチャーに並行して実行することができる．すると，画像のキャプチャーが終った時点で，除算を実行することができる．さらに，二つの除算を並列に実行することができる．結果として，画像重心の計算は，画像キャプチャーの終了後，除算一回分の計算時間で終了する．Fig. 4.2 に，画像重心を計算する論理回路を示す．論理回路の入力信号は，`START`，`X`，`Y`，`G` であり，出力信号は，`COG_X`，`COG_Y`，`COMPLETE` である．入力信号 `START` は，計算開始を指示する．信号 `X` と `Y` は画像の座標値を表し，信号 `G` は画素値 $g(x,y)$ を表す．出力信号 `COG_X` と `COG_Y` は，画像重心の座標値を表す．信号 `COMPLETE` は，計算完了を知らせる．モジュール `SUM` は，画像のキャプチャーと並行して，(4.2) 式で与えられる W_x，W_y，W を計算する．総和の計算が終了した後，信号 `div_start` が除算開始を指示する．

4.3.2 放射状射影

次に，放射状射影の計算を，FPGA 上に実装する．放射状射影に関して簡単に説明する．グレースケール画像上に座標系 $O-xy$ を固定する．画像上の点 (x,y) における画素値を $g(x,y)$ とする．画像重心から放射状に伸びる半直線に沿って，画素値を積分する．半直線が x 軸と成す角度を θ で表す．このとき，画素値の積分は，パラメータ θ の関数であり，次式で与えられる．

$$R(\theta) = \int_0^\infty g(c_x + \xi\cos\theta, c_y + \xi\sin\theta)d\xi. \tag{4.3}$$

ここで，c_x と c_y は，画像重心の座標値を表す．区間 $[0, 2\pi]$ で定められる積分値 $R(\theta)$ 全体を，放射状射影とよぶ．

参照画像を $g_{template}$，入力画像を g_{input} で表す．入力画像中の物体の姿勢を角度 α で表す．参照画像と入力画像における放射状射影を，$R_{template}$ と R_{input} で表す．二つの射影 $R_{template}$ と R_{input} は，次に示す関係を満たす．

$$R_{template}(\theta) = R_{input}(\theta + \alpha), \quad \forall\theta. \tag{4.4}$$

Table 4.2: Voting in computation of projection

initialize array $R(\theta)$
for $(x, y) = (0, 0), (0, 1), \cdots, (W-1, H-1)$ do
　$\Delta x = x - c_x$, $\Delta y = y - c_y$
　compute $\theta = \arctan(\Delta y / \Delta x)$
　increase $R(\theta)$ by pixel value $g(x, y)$
end

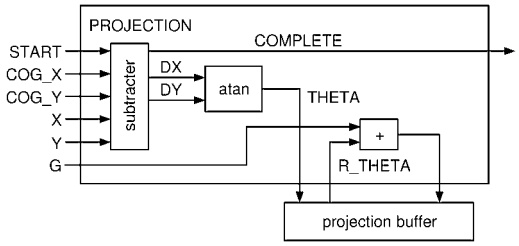


Fig. 4.3: Circuit to compute projection

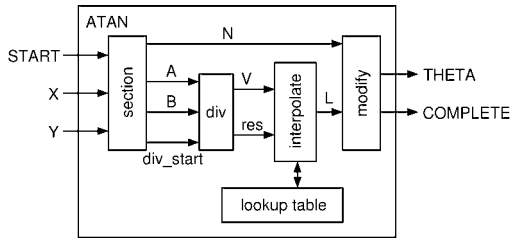


Fig. 4.4: Circuit to compute function *atan*

二つの射影間に，次式で与えられる誤差関数を導入する．

$$S(\tau) = \int_0^{2\pi} \left| R_{template}(\theta) - R_{input}(\theta + \tau) \right| d\theta. \tag{4.5}$$

この誤差関数は，$\tau = \alpha$ で最小値 0 を取る．したがって，上式で与えられる誤差関数を最小化することにより，姿勢角 α を算出することができる．

(4.3) 式で与えられる計算は，画像のキャプチャーが終了するまで開始できない．すなわち，この計算は，画像のキャプチャーと並行に計算することができず，リアルタイム処理の妨げになる．そこで，ハフ変換における投票の考えを導入し，画像のキャプチャーと並行に射影を計算できるように，アルゴリズムを構成する．(4.3) 式より，

$$x = c_x + \xi \cos\theta,$$
$$y = c_y + \xi \sin\theta.$$

パラメータ ξ を消去すると，

$$\tan\theta = \frac{y - c_x}{x - c_y}. \tag{4.6}$$

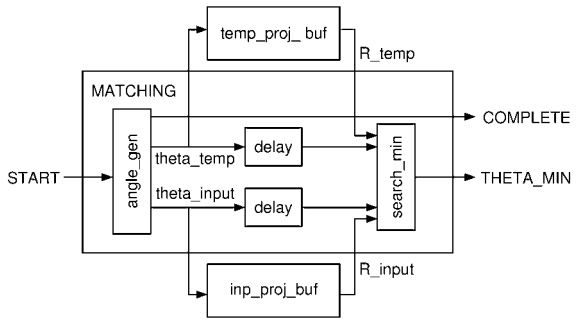


Fig. 4.5: Circuit to compare projections

上式は，点 (x, y) における画素値が，角度 θ で定められる積分に寄与することを意味する．すなわち，(4.6) 式で計算された角度 θ に対応する積分 $R(\theta)$ に，画素値 $g(x, y)$ を加算すればよい．画像のキャプチャーにおいて，画素値は論理回路に順次与えられる．したがって，画像のキャプチャーと並行して，すべての画素値を対応する積分値に加算することができる．以上の投票を，Table 4.2 に示す．

Fig. 4.3 に，投票により放射状射影を計算する回路を示す．この回路は，六本の入力信号 `START`, `COG_X`, `COG_Y`, `X`, `Y`, `G` と一本の出力信号 `COMPLETE` を持つ．モジュール `subtracter` は，$\Delta x = x - c_x$ と $\Delta y = y - c_y$ を計算する．モジュール `atan` は，$\theta = \arctan(\Delta y/\Delta x)$ を計算する．射影 $R(\theta)$ の値は，バッファに保存される．信号 `THETA` を指定し，角度 θ における積分値をバッファから読み出す．読み出された積分値 `R_THETA` に，信号 `G` で表される画素値 $g(x, y)$ が加算され，再び積分値をバッファに書き込む．

モジュール `atan` を，Fig. 4.4 に示す．あらかじめ，区間 $[0, 1]$ 内にいくつかの代表値 v を選び，$\theta = \arctan(v)$ を計算する．代表値における v と θ の組を，ルックアップテーブルに保存しておく．点 (x, y) が $0 \leq y \leq x$ を満たすとき，$\arctan(y/x)$ を計算する回路を構成しよう．まず，y/x を計算する．商は，区間 $[0, 1]$ に含まれる．したがって，関数値 $\arctan(y/x)$ は，ルックアップテーブル内の代表値を補間することで，計算することができる．モジュール `interpolate` は，この補間を実行する回路である．任意の点 (x, y) における関数値は，この補間を用いて計算することができる．たとえば，点 (x, y) が $0 \leq -x \leq y$ を満たす場合，関数値 $\arctan(y/x)$ の値は，$\pi - \arctan(-x/y)$ に等しい．商 $(-x/y)$ の値は区間 $[0, 1]$ に含まれるので，$\arctan(-x/y)$ の値はモジュール `interpolate` により計算することができる．そこで，x と y の符号，絶対値 $|x|$ と $|y|$ の大小関係により，x-y 平面を 8 個の領域に分割すると，どの領域に対しても，モジュール `interpolate` を用いて，関数値 $\arctan(y/x)$ の値を求めることができる．モジュール `section` は，点 (x, y) に対応する領域を求める．信号 `N` に領域番号，信号 `A` に $|x|$ と $|y|$ の小さい方，信号 `B` にもう一方を出力する．モジュール `modify` は，$\arctan(A/B)$ と N の値から，関数値 $\arctan(y/x)$ を構成する．

Fig. 4.5 に，(4.5) 式で与えられる誤差関数を最小化する論理回路を示す．モジュール `angle_gen` は，二つの信号 `theta_temp`, `theta_input` を生成する．それぞれ，(4.5) 式中の角度 θ と $\theta + \tau$ に対応する．対応する放射状射影の値 $R_{template}(\theta)$ と $R_{input}(\theta + \alpha)$ は，バッファ `temp_proj_buf` とバッファ `inp_proj_buf` から読み込まれる．モジュール `search_min` は，射影値の絶対差分の総和を計算し，計算値が最小となる角度 α を求める．

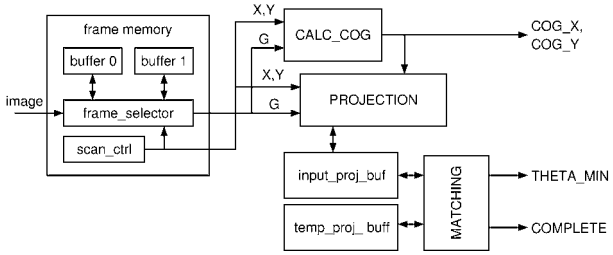


Fig. 4.6: Circuit to detect planar motion

物体と背景が明確に分離でき，画像内に複数の物体が存在しない場合，画像重心を計算することにより，平面運動物体の位置を求めることができる．さらに，参照画像と入力画像の放射状射影を比較することにより，平面運動物体の姿勢を算出することができる．Fig. 4.6 に，重心計算と放射状射影の比較により，平面運動物体の位置と姿勢を計算する回路を示す．ビデオデコーダボードでは，ダブルバッファリングにより，画像を連続的にキャプチャーする．モジュール `frame_selector` は，二つの画像バッファを切替える．モジュール `scan_ctrl` は，画面全体を巡回する座標値 (x, y) を生成する．参照画像の射影は，あらかじめ計算され，バッファ `temp_proj_buf` に格納されている．信号 `COG_X` と `COG_Y` が平面運動物体の位置を，信号 `THETA_MIN` が姿勢を表す．

4.3.3　ハフ変換

ハフ変換は，ビジョンの基本的なアルゴリズムの一つであり，応用範囲が広い．たとえば，ハフ変換を用いることにより，平面運動物体の位置と姿勢を，照明変動やオクルージョンに対してロバストに検出できることが知られている [29]．ハフ変換は時間を要する処理であるが，並列性が高いため，FPGA ベースビジョンに実装することにより，高速な実行が期待できる．また，ハフ変換の投票値を画素値に置き換えることで，第 2 章で述べたラドン変換の並列化と同じになる．そこで本項では，画素値を投票するハフ変換を FPGA ベースビジョンに実装する．

ハフ変換は，離散画像 $g(x, y)$ からハフ平面 $H(\rho, \theta)$ への変換である．まず，格子点 (x, y) に対して，次式を満たす ρ と θ の組を求める．

$$(-\sin\theta)x + (\cos\theta)y = \rho.$$

次に，上式を満たす ρ と θ の組に対して，$H(\rho, \theta)$ の値を画素値 $g(x, y)$ だけ増加させる．すべての画素値に対して以上の計算を行った結果が，ハフ変換である．上式は，点 (x, y) を通る直線群を表し，ρ は直線の原点からの符号付き距離を，θ は直線と x 軸が成す角を表す．以上の計算過程を，Table 4.3 に示す．

Fig. 4.7 に，ハフ変換を計算する回路の概略を示す．制御信号は省略する．区間 $[0, 2\pi]$ を K 分割し，$\Delta\theta = 2\pi/K$ とする．このとき，角度 θ の値は，$k\Delta\theta$ $(k = 0, 1, \cdots, K-1)$ である．角度 $k\Delta\theta$ に対する計算を，Fig. 4.8 に示すモジュール `voting(`k`)` で行う．角度 $k\Delta\theta$ におけるハフ変換の結果は，バッファ `hough_buf(`k`)` に格納される．モジュール `voting(`k`)` の入力信号は画像の座標値 `X`，`Y`，ならびに画素値 `G` である．モジュールには，$c_k = \cos(k\Delta\theta)$ と

Table 4.3: Voting in computation of Hough transform

initialize array $H(\rho, \theta)$
for $(x, y) = (0, 0), (0, 1), \cdots, (W-1, H-1)$ do
  for $k = 0, 1, \cdots, K-1$ do
    compute $\theta = k\Delta\theta$
    compute $\rho = (-\sin\theta)x + (\cos\theta)y$
    increase $H(\rho, \theta)$ by pixel value $g(x, y)$
end

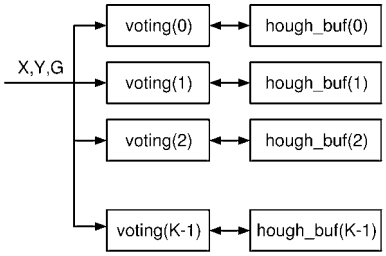


Fig. 4.7: Circuit to compute Hough transform

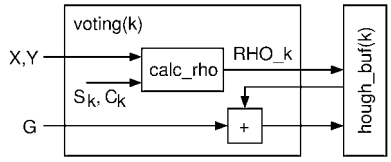


Fig. 4.8: Circuit to perform voting

$s_k = \sin(k\Delta\theta)$ の値が，あらかじめ記憶されている．モジュール `calc_rho` で $\rho_k = c_k y - s_k x$ の値を計算し，信号 `RHO_k` に出力する．距離 ρ_k の計算が完了後，バッファ `hough_buf(`k`)` を更新する．まず，信号 `RHO_k` を指定し，距離 ρ_k に対する積分値をバッファから読み出す．読み出された積分値に，信号 `G` で表される画素値 $g(x, y)$ を加算し，再び積分値をバッファに書き込む．以上の計算は，角度 $k\Delta\theta$ $(k = 0, 1, \cdots, K-1)$ に対して，並列に実行することができる．したがってハフ変換の計算は，モジュール `voting(`k`)` とバッファ `hough_buf(`k`)` を，角度の分割数 K 個並列に並べた，Fig. 4.7 に示す回路で実現される．

4.4 実験的評価

4.4.1 画像重心の計算

画像重心を計算する回路を，FPGA ベースビジョンシステムに実装した．画面中央部 256×256[pixel] の領域に対して画像重心を計算する．このとき FPGA は，2%のゲートを消費した．また，回路を 58MHz で駆動できることが，シミュレーションで判明した．実

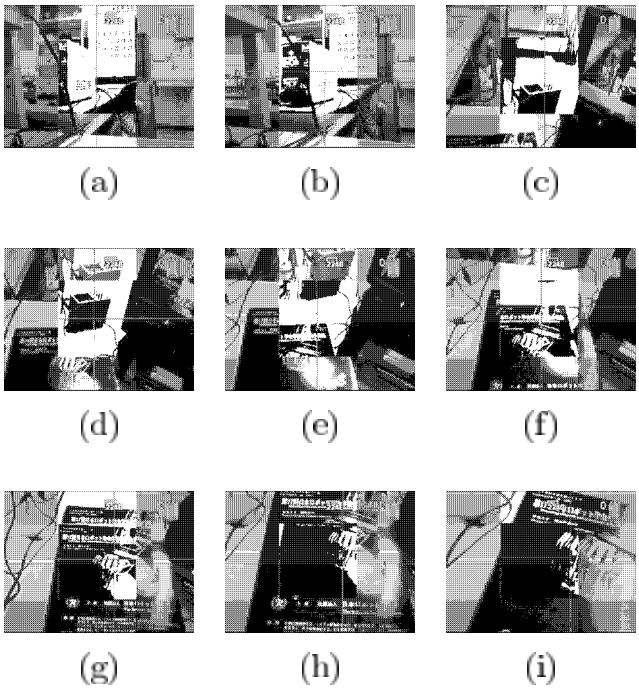


Fig. 4.9: Detection of image gravity center

験では，画像のキャプチャに合わせて，回路を 12MHz で駆動した．除算一回は 32 クロックで終了するので，画像重心の計算はキャプチャ後約 2.7[μsec] で完了する．Fig. 4.9 に，画像重心の計算例を示す．水平線と垂直線が，計算された画像重心の座標を表す．図に示すように，画像重心をリアルタイムで計算することができる．

4.4.2　放射状射影

重心計算と放射状射影の比較により，物体の平面運動を検出する論理回路を，FPGA ベースビジョンに実装した．画面中央部 256×256[pixel] の領域に対して画像重心と放射状射影を計算し，放射状射影のマッチングを行う．角度は，64 段階で表される．このとき FPGA は，13%のゲートを消費した．Fig. 4.10 に，実装回路で消費された FPGA のゲートを示す．回路を 24MHz で駆動できることが，シミュレーションで判明した．実験では，画像のキャプチャに合わせて，回路を 12MHz で駆動した．一画素に対する投票には 1 クロック要するので，放射状射影の計算には，1×256×256 クロックが必要である．これは，5.46[msec] に相当する．ただし，画像のキャプチャと並行して実行されるので，放射状射影の計算は画像のキャプチャとともに完了する．一方，射影間のマッチングに 64×64 クロックを要しているため，位置と姿勢の検出は，キャプチャ後約 341[μsec] で完了する．Fig. 4.11 に，平面運動物体の位置と姿勢の検出例を示す．水平線と垂直線が物体の位置を，斜めの直線が算出された姿勢を表す．図に示すように，物体の位置と姿勢をリアルタイムで計算することができる．

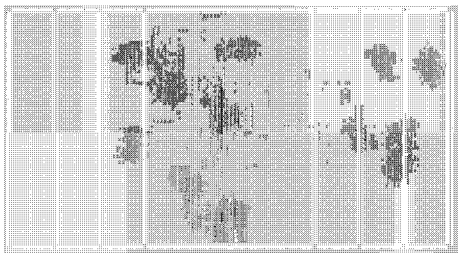


Fig. 4.10: Implementation of logic circuit to detect planar motion object

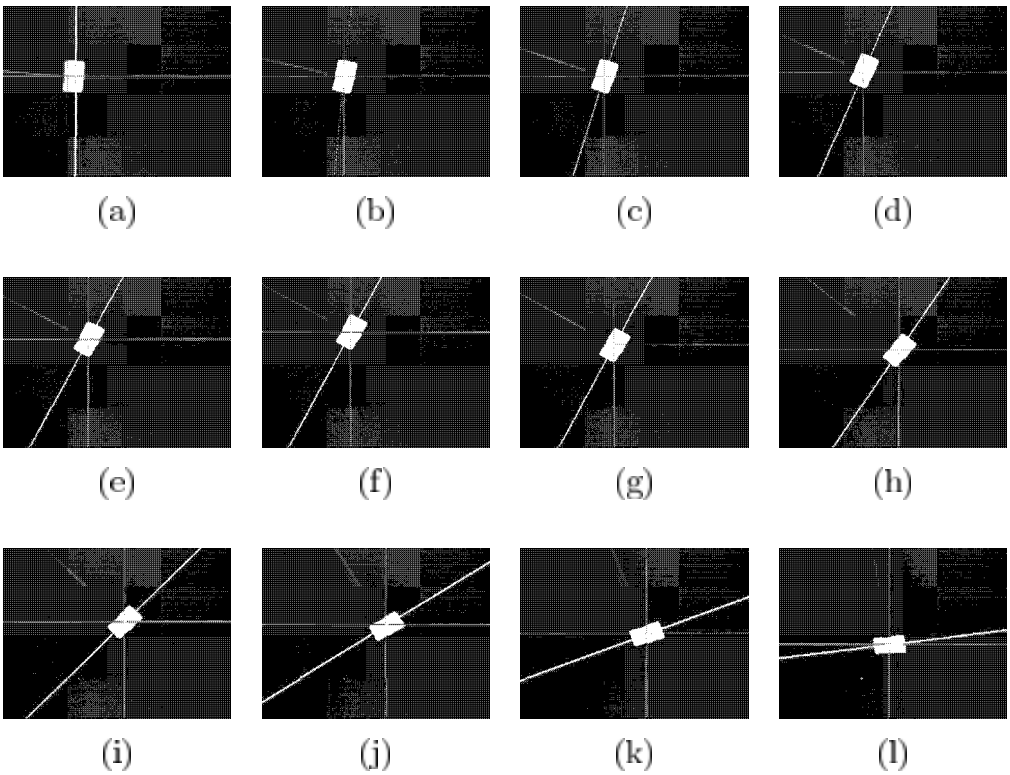
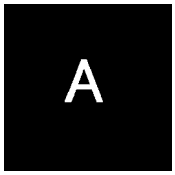


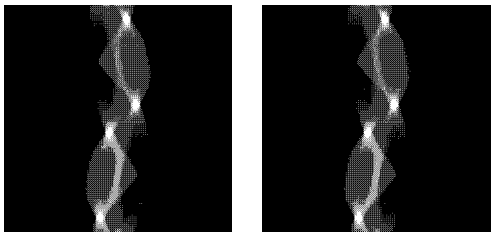
Fig. 4.11: Detection of position and orientation

4.4.3 ハフ変換

ハフ変換を計算し，変換結果を出力する論理回路を FPGA 上に実装した．ハフ変換の結果は，FPGA 内の特別な領域であるブロック RAM に書き込む．回路設計においては，CycleC による回路記述に対してシミュレーションを実行することにより，回路記述を検証する．Fig. 4.12 に，C 言語で記述したソフトウェアによりハフ変換を実行した結果と，ハフ変換の回路記述のシミュレーションによりハフ変換を実行した結果を示す．双方がほぼ一致しており，CycleC による回路記述が正しいことを示す．ハフ変換回路による変換結果の出力例を，Fig. 4.13 に示す．画面中央部 256×256[pixel] の領域に対してハフ変換を計算し，その結果を画面中央下部に表示する．角度 θ は，144 段階で表される．すなわち，角度の分解能は 2.5° である．距離 ρ は，−127[pixel] から 127[pixel] まで 128 段階で計算する．すなわち，距離の分解能は 1[pixel] である．このとき FPGA は，77%のゲートを使用した．Fig. 4.14 に，実装回路で消費された FPGA のゲートを示す．回路は，24MHz



(a) input



(b) software　　(c) simulation

Fig. 4.12: Hough transfroms by software and by simulation of logic circuit

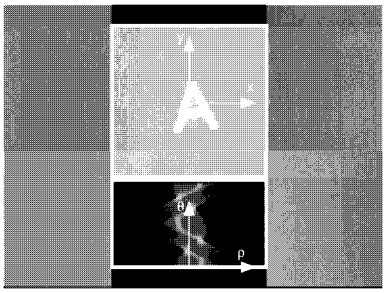


Fig. 4.13: Image and its Hough transform

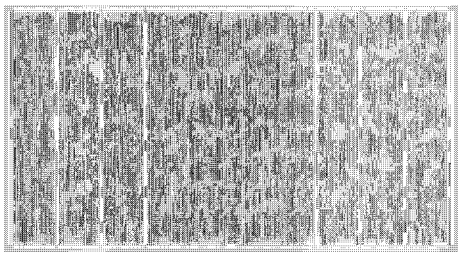


Fig. 4.14: Implementation of logic circuit to compute Hough transform

で駆動することができた．一画素に対する投票には 2 クロック要するので，ハフ変換の計算には，2×256×256 クロックが必要である．これは，5.46[msec] に相当する．ただし，画像のキャプチャと並行して実行されるので，ハフ変換の計算は画像のキャプチャとともに完了する．Fig. 4.15 に，ハフ変換の計算例を示す．図に示すように，入力画像に対するハフ変換をリアルタイムで計算することができる．

なお，平面運動を検出する回路，ハフ変換を計算する回路ともに，LSI 回路設計を専門としない設計者が，約半年で完成させている．

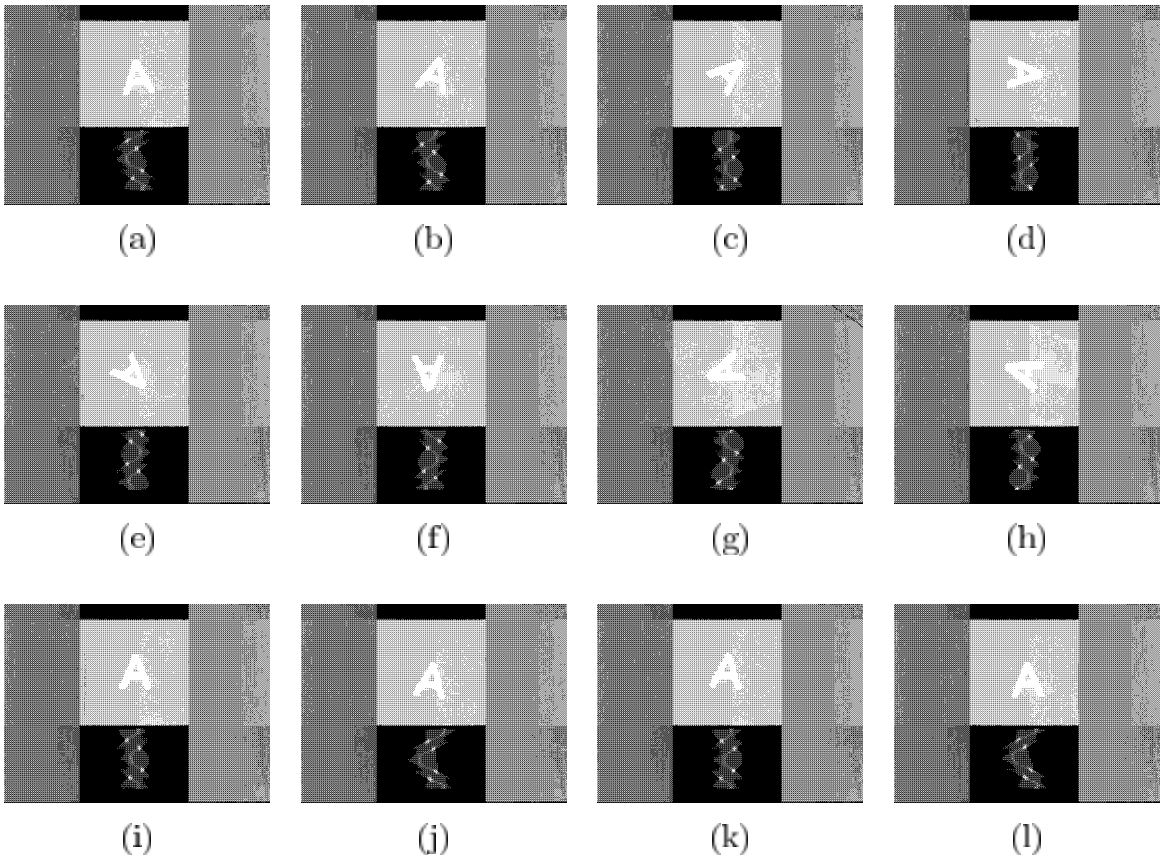


Fig. 4.15: Realtime computation of Hough transform

4.5 おわりに

本章では，FPGA ベースリアルタイムビジョンシステムを紹介するとともに，画像重心と放射状射影を用いて平面運動物体の位置と姿勢を算出するアルゴリズムを実装した結果を述べた．その結果，200 万ゲート相当の FPGA の 15%以下のゲート数で，アルゴリズムを実装できることがわかった．計算時間は約 5[msec] である．ただし，画像のキャプチャと並行して計算が実行されるため，キャプチャ後約 0.3[msec] でアルゴリズムが完了する．また，ハフ変換の計算を実装した結果，200 万ゲート相当の FPGA の 80%程度のゲート数を消費することが判明した．これは，バッファを FPGA 内に構築したためであり，バッファを SRAM 内に構築することによりゲート数を削減することが可能である．計算時間は約 5[msec] である．ただし，画像のキャプチャとともに，ハフ変換の計算が完了する．以上のように，FPGA ベースリアルタイムビジョンシステムを用いることにより，計算時間の短縮のみならず，時間遅れの短縮を実現することができる．また，アルゴリズムの FPGA への実装では，C/C++言語ベースの回路設計を用いた．結果として，平面運動検出回路やハフ変換計算回路を，それぞれ約半年で完成させることができた．設計者は LSI 設計を専門としていないため，ハードウエア記述言語のみによる設計では，同じ期間でアルゴリズムを実装することは不可能であったと考える．

現在，バッファを含めてすべての回路を FPGA 内に実装している．バッファを外部の SRAM に実装することにより，計算量がさらに多いアルゴリズムを実装することができ

る．整合フィルタ [30] や位相限定相関法 [7]，ハフフーリエ変換法 [10]，一般化ハフ変換 [2, 3] は，平面運動物体の位置と姿勢をロバストに検出でき，照明変動や対象物体の欠けなどに対応できることが知られている．しかしながら，これらのアルゴリズムは，一次元/二次元フーリエ変換，ハフ変換，極座標変換など，計算量の多い演算を必要とし，アルゴリズムをリアルタイムに実行することを困難にしている．これらのアルゴリズムをFPGAベースビジョンに実装し，リアルタイムで実行することが今後の課題である．また，ビジョンアルゴリズムの計算時間を短縮し，CCDカメラ以上のフレームレートが実現できるCMOSイメージセンサからの画像を用いて位置と姿勢を検出すること，CPU内蔵のFPGAを用いて，認識や判断を含むビジョンアルゴリズムを実装することも今後の課題である．

第5章　結論

本論文では，平面運動物体の位置と姿勢の検出を目指し，ラドン変換を用いたビジョンアルゴリズムを提案した．そして，提案するビジョンアルゴリズムをソフトウェアに実装し，検出実験による評価を行った．また，ビジョンアルゴリズムをハードウェアに実装し，ハードウェア上での実行性能について検証した．

本論文で得られた研究成果を以下にまとめる．

(1) ラドン変換と一次元位相限定相関を用いたビジョンアルゴリズムを提案し，位相スペクトルを基にしたマッチングで対象の位置と姿勢を検出する方法を示した．さらに，このアルゴリズムによって，画像内の複数の対象を検出することが可能であることを示した．

(2) ソフトウェアによる実装で，ビデオフレームレート内での検出処理が可能なビジョンアルゴリズムを提案した．次にPCを用いたビジョンシステムを構築し，提案するアルゴリズムをソフトウェアに実装した．実験結果より，ビデオフレームレートで平面運動物体の位置と姿勢を検出が可能であることを示した．

(3) ビジョンアルゴリズムをFPGAに実装し，ハードウェア上での実行したときの処理時間とFPGAの使用領域の結果を示した．また，ハードウェアに実装する有効性と問題点について確認された．

以上が本論文によって明らかにされた内容である．

謝辞

本研究の遂行にあたり，非常に多くの方々からご支援をいただきました．ここにその方達への感謝の意を表します．

まずはじめに，著者の指導教員である立命館大学理工学部 平井慎一教授には，本研究の方針，実験方法，研究の遂行方法などに対して終始有益な御指導，御鞭撻，御意見を受け賜りました．ここに改めて深い敬意と感謝の意を表します．誠に有難うございました．

本研究を遂行するにあたり，平井研究室所属の院生，学部生の方々には，多大なるご協力を頂きました．特に当時の院生であった増渕 章洋さん，座光寺 正和さんには，同じビジョン研究のグループとして，様々なご協力を賜りました．ここに，深く感謝の意を表します．

本研究の一部は，新エネルギー・産業技術総合開発機構 (NEDO) 地域新生コンソーシアムの援助を受けたことを付記し感謝の意を表します．

最後に，暖かく見守っていただいた私の家族に心から感謝致します．

2004年6月，　坪井 辰彦

参考文献

[1] D.I. Barnea and H.F. Silverman, “A Class of Algorithms for Fast Digital Image Registration”, IEEE Trans. Compt., vol.C-21, no.2, pp.179-186, 1972.

[2] D.H. Ballard, “Generalizing the Hough Transform to Detect Arbitrary Shapes”, Pattern Recognit., vol.13, no.2, pp.111–122, 1981.

[3] 木村彰男，渡辺孝志，“アフィン変換に不変な任意図形検出法として拡張された一般化ハフ変換”，信学論 (D-II)，vol.J84-D-II，no.5，pp.789–798，2001.

[4] B.S. Reddy and B.N. Chatterji, “An FFT-based Technique for Translation, Rotation, and Scale-invariant Image Registration”, IEEE Trans. Image Processing, vol.IP-5, no.8, pp.1266–1271, 1996.

[5] D. Casasent and D. Psaltis, “Position, Rotation, and Scale Invariant Optical Correlation”, Applied Optics, vol.15, no.7, pp.1795–1799, 1976.

[6] D. Casasent and D. Psaltis, “New Optical Transform for Pattern Recognition”, Proc. IEEE., vol.65, no.1, pp.77–84, 1977.

[7] Q. Chen, M. Defries and F. Deconinck, “Symmetric Phase-only Matched Filterring of Fourier-Mellin Transforms for Image Registration and Recognition”, IEEE Trans. PAMI, vol.16, no.12, pp.1156–1166, 1994.

[8] 佐々木宏，小林孝次，青木孝文，川又政征，樋口龍雄，“回転不変位相限定相関による画像の回転角度計測について”，映像情報メディア学会技術報告，vol.20，no.41，pp.1–6，1996.

[9] 佐々木宏，野村和正，“位相限定相関を用いた画像のサブピクセル精度の位置ずれ検出”，信学技報，vol.101，no.141(CAS2001 1-28)，pp.79–86，2001.

[10] 大西弘之，鈴木寿，有本卓，“ハフおよびフーリエ変換を用いた拡大・回転・平行移動検出法の部品位置決めへの応用”，日本ロボット学会誌，vol.16，no.2，pp.232–240，1998.

[11] 三浦純：“PC を用いた画像処理”，日本ロボット学会誌，vol.16，no.8，pp.1046–1049，1998

[12] G. H. Grunlund: "Fourier Preprocessing for Hand Print Character Recongnition", IEEE Trans. Comput., vol.C-21, no.2, pp.195–201, 1972.

[13] C. T. Zahn and R Z. Roskies: "Fourier Descriptors for Plane Closed Curves", IEEE Trans. Comput., vol.C-21, no.3, pp.269–281, 1972.

[14] M. Hu: "Visual Pattern Recognition by Moment Invariants", IRE Trans. Info. Theory, vol.IT-8, pp.179–187, 1962.

[15] 斎藤恒雄：画像処理アルゴリズム．pp.124–126，近代科学社，1993

[16] Thompson, C. D., *Fourier Transforms in VLSI*, IEEE Trans. on Computers, Vol.C-32, No.11, pp.1047–1057, 1983.

[17] Maresca, M., Lavin, M., and Li, H., *Parallel Hough Transform Algorithms on Polymorphic Torus Architecture*, Levialdi, S. eds., *Multicomputer Vision*, Academic Press, pp.9–21, 1988.

[18] Inoue, H., Tachikawa T., and Inaba, M., *Robot Vision System with a Correlation Chip for Real-time Tracking, Optical Flow and Depth Map Generation*, Proc. IEEE Int. Conf. on Robotics and Automation, pp.1621–1626, Nice, May, 1992.

[19] Bugeja, A. and Yang, W., *A Reconfigurable VLSI Coprocessing System for the Block Matching Algorithm*, IEEE Trans on VLSI Systems, Vol.5, No.3, pp.329–337, 1995.

[20] Hariyama, M., Takeuchi, T., and Kameyama, M., *VLSI Processor for Reliable Stereo Matching Based on Adaptive Window-Size Selection*, Proc. 2001 IEEE Int. Conf. on Robotics and Automation, pp.1168–1173, Seoul, May, 2001.

[21] Eklund, J.-E., Svensson, C., and Aström, A., *VSLI Implementation of a Focal Plane Image Processor – A Realization of the Near-Sensor Image Processing Concept*, IEEE Trans. on VLSI Systems, No.4, Vol.3, pp,322–335, 1996.

[22] Ishii, I., Nakabo, Y., and Ishikawa, M., *Target Tracking Algorithm for 1ms Visual Feedback System using Massively Parallel Processing Vision*, Proc. 1996 IEEE Int. Conf. on Robotics and Automation, pp.2309–2314, Mineapolis, May, 2001.

[23] Woods, R., Trainor, D., and Heron, J.-P., *Applying an XC6200 to Real-Time Image Processing*, IEEE Design & Test of Computers, Vol.15, No.1, pp.30–38, 1998.

[24] 安永, 高見, 吉原, FPGA を用いたナノ秒オーダ画像処理ハードウェア, 電子通信情報学会誌, Vol.J84-D-II, No.10, pp.2280–2292, 2001.

[25] 永渕, 吉永, 横田, 大津, 馬場, 並列 FPGA システムによる医療用画像処理の高速化, 電子情報通信学会技術研究報告 CPSY2001, Vol.101, No.217, pp.1–6, 2001.

[26] Memik, S. O., Katsaggelos, A. K., and Sarrafzadeh, M., *Analysis and FPGA Implementation of Image Restortion under Resource Constraints*, IEEE Trans. on Computers, Vol.52, No.3, pp.390–399, 2003.

[27] Deng, D. D. S. and ElGindy, H., *High-speed Parameterisable Hough Transform Using Reconfigurable Hardware*, Proc. Pan-Sydney area Workshop on Visual Information Processing, pp.51–57, 2001.

[28] Tsuboi, T., Masubuchi, A., Hirai, S., Yamamoto, S., Ohnishi, K., and Arimoto, S., *Video-frame Rate Detection of Position and Orientation of Planar Motion Objects using One-sided Radon Transform*, Proc. IEEE Int. Conf. on Robotics and Automation, Vol. 2, pp.1233–1238, Seoul, May, 2001.

[29] Onishi, H. and Suzuki, H., *Detection of Rotation and Parallel Translation Using Hough and Fourier Transforms*, Proc. 1996 Int. Conf. on Image Processing, Vol.3, pp.827–830, 1996.

[30] Turin, G. L., *An Introduction to Digital Matched Filters*, Proceedings of the IEEE, Vol. 64, No. 7, pp.1093–1112, 1977.

研究関連

学会誌論文

1. 坪井辰彦，平井慎一，“片側ラドン変換を用いた平面運動検出”，日本ロボット学会誌，vol. 22，no. 2，pp. 207–214，2004

2. 平井慎一，座光寺正和，増渕章洋，坪井辰彦，“FPGA ベースリアルタイムビジョン”，日本ロボット学会誌，vol. 22，no. 7，2004(掲載予定)

3. 坪井辰彦，平井慎一，“ラドン変換と一次元位相限定相関を用いた複数物体の検出”，電子情報通信学会論文誌 (掲載予定)

査読付き国際学会論文

1. Tatsuhiko Tsuboi, Akihiro Masubuchi, Shinichi Hirai, Shinya Yamamoto, Kazuhiko Ohnishi, and Suguru Arimoto,“Video-frame Rate Detection of Position and Orientation of Planar Motion Objects using One-sided Radon Transform”, Proc. of the 2001 IEEE Int. Conf. on Robotics and Automation, pp. 1233–1238, Seoul, Korea, May 2001.

2. Tatsuhiko Tsuboi, Masakazu Zakouji, Akihiro Masubuchi, and Shinichi Hirai, “Parallel Processing of One-sided Radon Transform for the Realtime Detection of Position and Orientation of Planar Motion Objects”, Proc. of the 2002 IEEE Int. Conf. on Robotics and Automation, pp. 2925–2930 Washington, D.C, U.S.A, May 2002

口頭発表

1. 坪井辰彦，平井慎一，“片側ラドン変換を用いたビデオフレームレートでの物体認識と平面運動計測”，第 18 回日本ロボット学会学術講演会予稿集 (第 2 分冊)，pp. 645–646，滋賀，(2000.9)

2. 坪井辰彦，平井慎一，“片側ラドン変換を用いた平面運動物体の識別と位置・姿勢検出”，ロボティクス・メカトロニクス講演会’01，2A1-C10，香川，(2001.6)

3. 坪井辰彦，平井慎一，“片側ラドン変換を用いたビデオフレームレートでの平面運動計測とその実験的評価”，2001年電子情報通信学会情報・システムソサイエティ大会講演論文集，pp. 225，東京，(2001.9)

4. 坪井辰彦，平井慎一，“複数の座標中心上の片側ラドン変換を用いた位置・姿勢検出”，ロボティクス・メカトロニクス講演会'02講演論文集，2P1-J12，島根, (2002.6)

5. 坪井辰彦，平井慎一，“ラドン変換とフーリエ位相限定相関を用いた平面運動物体の位置・姿勢検出”，(SI2002) 第3回SICEシステムインテグレーション部門　講演会論文集 (II)，pp. 117–118, 兵庫，(2002.12)

6. 坪井辰彦，平井慎一，“ラドン変換と一次元位相限定相関を用いた平面運動物体の位置と姿勢の検出アルゴリズム”，ロボティクス・メカトロニクス講演会'03講演論文集, 2P1-1F-C5，北海道, (2003.5)

7. 坪井辰彦，平井慎一，“ラドン変換と位相限定相関を用いたビジョンアルゴリズムのロバスト性評価”，(SI2003) 第4回SICEシステムインテグレーション部門　講演会, 1I2-2, 東京，(2003.12)