片側ラドン変換に基づく平面運動検出アルゴリズムの並列化

○平井 慎一, 坪井 辰彦, 増渕 章洋, 座光寺 正和
立命館大学 ロボティクス学科

Parallel Processing of One-sided Radon Transform for the Detection of the Position and the Orientation of Planar Motion Object

○ Shinichi Hirai, Tatsuhiko Tsuboi, Akihiro Masubuchi, and Masakazu Zakouji
Dept. of Robotics, Ritsumeikan Univ.

Abstract - *In this report, we will develop parallel processing of one-sided Radon transform for the detection of the position and the orientation of a planar motion object. Detection of planar motion based on one-sided Radon transform can be applied to arbitrary shapes and is robust against overlapping of objects but it requires much computation time. Thus, we will construct a parallel computation of one-sided Radon transform to reduce the computation time. The developed parallel algorithm is evaluated by comparing it with the original algorithm.*
keywords: vision, realtime, planar motion detection, one-sided Radon transform, FPGA

1. はじめに

運動物体の位置と姿勢を検出することは，様々な場所に置かれている物体のハンドリングや変形を伴う物体のハンドリングなど，高度なハンドリングの基盤となる技術である．たとえば，食品産業においては，食品の衛生を保つために，ハンドリングの自動化が望まれている．しかしながら，食品はランダムな位置や姿勢で置かれていることが多く，ハンドリングの自動化のために，食品の位置や姿勢を検出することが要求される．また，リサイクルにおいては，分解部品の分別の多くが人手により成されており，分別精度の向上のために自動化が望まれている．このとき，分解部品の位置と姿勢を検出することが要求される．

著者らは，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三つのアルゴリズムを提案した [1]．これらのアルゴリズムは，CCD カメラからの入力画像とあらかじめ撮影した参照画像を比較することにより，任意形状の物体の位置と姿勢を算出することができる．アルゴリズムの一つは物体の画像重心の計算を必要とせず，他の二つは画像重心の計算を必要とする．前者のアルゴリズムにおいては，対象物体を背景から分離する必要がないため，対象物体が他の物体と重なっていたり，対象物体の一部が画像の外にある場合でも，位置と姿勢の検出が可能である．しかしながら，このアルゴリズムは，片側ラドン変換の計算，パワースペクトルの計算，パワースペクトルの二次元マッチングから構成されており，多くの計算時間を要する．一方，これらの計算は並列性が高く，並列演算により計算時間を短縮できる可能性が高い．

コンピュータビジョンにおける並列計算に関して，多くの研究が成されている．画像における変換は，計算時間を要する処理であり，フーリエ変換の並列演算 [2]，ハフ変換の並列演算 [3] に関する研究が進められた．画像ブロック間の相関演算は，コンピュータビジョンの基本的な処理の一つであり，その並列演算が研究され，専用 VLSI が開発されている [4, 5]．また，適応ウインドウに基づくステレオビジョンの並列化と VLSI 化が検討されている [6]．近年，センシングと信号処理の並列処理に関する研究が進展している．Near-sensor image processing の概念に基づき，フォトディテクタと信号処理部を二次元的に配置したビジョンチップが開発されている [7]．また，ビジョン信号の並列演算のために

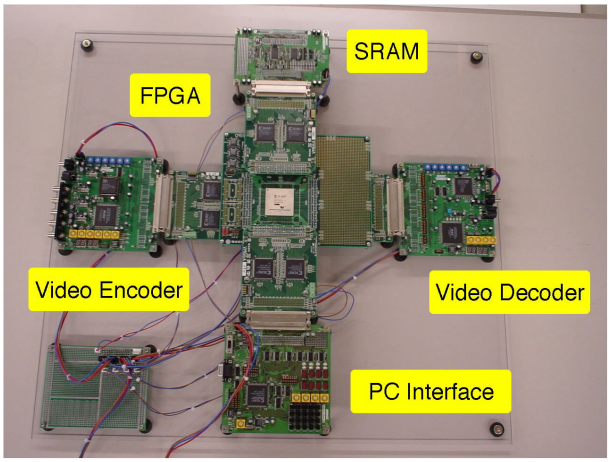


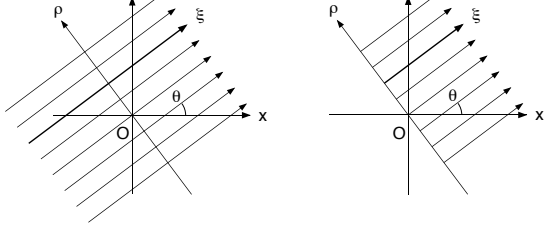
Fig.1: FPGA board for realtime vision

S^3PE アーキテクチャが提案され，高速物体追跡に適用されている [8]．

本報告では，平面運動物体の位置と姿勢を検出するために，片側ラドン変換の並列演算を開発する．まず，FPGA を用いたビジョンシステムを構築する．次に，片側ラドン変換を用いて，物体の位置と姿勢を検出するアルゴリズムについて説明する．次に，片側ラドン変換の並列演算について考察し，投票の概念に基づく並列アルゴリズムを提案する．最後に，オリジナルアルゴリズムと比較することにより，並列アルゴリズムを実験的に評価する．

2. FPGA を用いたビジョンシステム

本研究では，実際のハンドリングで使用されるビジョンシステムを構築することを目的とする．物体ハンドリングにおいては，高いハンドリングスピードが要求されるため，ビジョンシステムには高速性が要求される．一方，対象とする物体や環境は多様であり，多様な物体ハンドリングに適用できる汎用性が要求される．高速性と汎用性を満たすために，FPGA を用いたビジョンシステムを構築する．FPGA においては，論理回路を書き換えることができるため，多様な物体ハンドリングに適用できる汎用性を実現できる．一方，ビジョ



(a) (b)

Fig.2: Integral paths in Radon transform and those in one-sided Radon transform

ンアルゴリズムを並列に実行することにより，高速性を実現することができる．試作した FPGA ボードを図1に示す．画像の入力には，CCD カメラと Bt シリーズに代表されるビデオキャプチャチップを用いる．このとき，CCD カメラからの NTSC 信号は，順次 A/D 変換され，後段の画像処理部に送られる．画像処理部においては，ディジタル画像を扱うので，論理回路が構成できれば十分である．そこで，FPGA 上に，画像処理用のハードウェアを実現する．ただし，ラドン・フーリエ変換法では，ラドン変換やフーリエ変換の結果を保存しておくメモリが必要である．そこで，SRAM 内に変換の結果を保存し，必要に応じて参照する．使用している FPGA は Xilinx Vertex-E 2000 である．ビデオデコーダ，ビデオエンコーダ，SRAM が周辺に配置されている．論理回路の記述には，CycleC を用いる．Cycle C は，Visual C/C++のサブセットであり，コンパイルすることにより PC 上でアルゴリズムを評価することができる．また，Cycle C による記述は，HDL に変換することができるので，FPGA の論理回路を構成することができる．

3. 片側ラドン変換

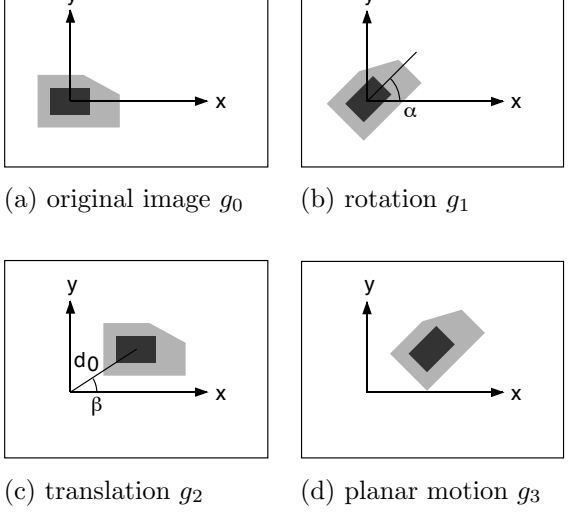
本節では，片側ラドン変換を用いて，平面運動物体の位置と姿勢を検出するアルゴリズムを紹介する．グレースケール画像に座標系 $O-xy$ を設定する．点 (x, y) における画素値を $g(x, y)$ で表す．Fig.1-(a) に示すように，原点からの距離 ρ，x 軸と成す角 θ の直線に沿って，画素値 $g(x, y)$ を線積分する．この積分を，ラドン変換とよぶ．ラドン変換においては，直線群に沿う線積分を計算する．原点回りに直線群を角度 π 回転させると，もとの直線群に一致する．これにより，ラドン変換では，角度 α の回転と角度 $\alpha + \pi$ の回転を区別することが困難である．そこで，角度 α の回転と角度 $\alpha + \pi$ の回転を区別するために，積分路として直線の代わりに半直線を導入する．すなわち，Fig.1-(b) に示すように，原点からの距離 ρ，x 軸と成す角 θ の半直線に沿って，画素値 $g(x, y)$ を線積分する．この積分を，片側ラドン変換とよび，次式で表される．

$$U[g](\rho, \theta) = \int_0^\infty g(\xi \cos\theta - \rho \sin\theta, \xi \sin\theta + \rho \cos\theta) d\xi. \quad (1)$$

Fig.3-(a)，(b) に示すように，原画像を g_0，原画像を原点回りに角度 α 回転させて得られる画像を g_1 とする．画像 g_0 の片側ラドン変換 $U_0(\rho, \theta)$ と g_1 の片側ラドン変換 $U_1(\rho, \theta)$ は，次式を満たす．

$$U_0(\rho, \theta) \equiv U_1(\rho, \theta + \alpha), \quad \forall \rho, \theta. \quad (2)$$

Fig.3-(c) に示すように，原画像を角度 β の方向に距離 d_0 並進移動させて得られる画像を g_2，その片側ラドン



(a) original image g_0 (b) rotation g_1

(c) translation g_2 (d) planar motion g_3

Fig.3: Planar motion of rigid object

変換を $U_2(\rho, \theta)$ とする．原画像に写る物体の内部にあるように，座標原点を選ぶと，

$$U_0(\rho, \theta) \equiv U_2(\rho - d_0 \sin(\theta - \beta), \theta), \quad \forall \rho, \quad \theta + \frac{\pi}{2} = \beta. \quad (3)$$

Fig.3-(d) に示すように，原画像に回転角 α，移動方向 β，移動距離 d_0 で与えられる平面運動を与えた画像を g_3，その片側ラドン変換を $U_3(\rho, \theta)$ とする．このとき，

$$U_0(\rho, \theta - \alpha) \equiv U_3(\rho - d_0 \sin(\theta - \beta), \theta), \quad \forall \rho, \quad \theta + \frac{\pi}{2} = \beta. \quad (4)$$

片側ラドン変換 $U(\rho, \theta)$ の ρ に沿うパワースペクトルを $\mathcal{F}(f, \theta)$ で表す．ラドン変換のパワースペクトルは周期 π の周期関数である．一方，片側ラドン変換 $U(\rho, \theta)$ と $U(\rho, \theta + \pi)$ は一般に異なるので，片側ラドン変換のパワースペクトルは周期 2π の周期関数である．片側ラドン変換 $U_0(\rho, \theta)$ と $U_3(\rho, \theta)$ 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，次式が成り立つ．

$$\mathcal{F}_0(f, \theta - \alpha) \equiv \mathcal{F}_3(f, \theta) \quad \forall f, \quad \theta + \frac{\pi}{2} = \beta. \quad (5)$$

片側ラドン変換の性質を利用して，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する．参照画像と入力画像を比較して，運動を検出する．参照画像を g_{sample}，入力画像を g_{input} で表す．それらの片側ラドン変換をそれぞれ $U_{sample}(\rho, \theta)$，$U_{input}(\rho, \theta)$ で表す．片側ラドン変換のパワースペクトルをそれぞれ $\mathcal{F}_{sample}(f, \theta)$，$\mathcal{F}_{input}(f, \theta)$ で表す．まず，パワースペクトル $\mathcal{F}_{sample}(f, \theta)$ と $\mathcal{F}_{input}(f, \theta)$ を比較し，回転角 α と並進移動方向 β を算出する．次に，片側ラドン変換 $U_{sample}(\rho, \theta)$ と $U_{input}(\rho, \theta)$ を比較し，並進移動距離 d_0 を求める．以上のアルゴリズムは，次のようにまとめられる．

Algorithm

Step 1 Compute $U_{sample}(\rho, \theta)$ and $U_{input}(\rho, \theta)$.

Step 2 Compute $\mathcal{F}_{sample}(f, \theta)$ and $\mathcal{F}_{input}(f, \theta)$.

Step 3 Find θ_{sample} and θ_{input} that minimize $\| \mathcal{F}_{sample}(f, \theta_{sample}) - \mathcal{F}_{input}(f, \theta_{input}) \|$.

$$\beta = \theta_{input} + \pi/2$$

Step 5 Find d_0 that satisfies

$$U_{sample}(\rho, \theta_{sample}) \equiv U_{input}(\rho + d_0, \theta_{input}) \quad \forall\rho.$$

片側ラドン変換のパワースペクトルは周期 2π であるので，パワースペクトルの比較により，回転角を求めることができる．ただし，このアルゴリズムは，パワースペクトルの二次元マッチングを含んでおり，多くの計算時間を要する．

4. 片側ラドン変換の並列処理

本節では，片側ラドン変換の並列計算について考察する．グレースケール画像が CCD カメラにより撮影され，NTSC 形式で処理部に送られるとする．画像の幅を W，高さを H で表す．グレースケール画像は，格子点 $P_{i,j}$ $(i = 0, 1, \cdots, W-1, j = 0, 1, \cdots, H-1)$ における画素値 $g_{i,j}$ により与えられる．オリジナルのアルゴリズムでは，画素値 $g(x, y)$ の値を，周囲の格子点における画素値を補間することにより求める．すなわち，

$$g(x,y) = (1-r_x)(1-r_y)g_{i,j} + r_x(1-r_y)g_{i+1,j} + (1-r_x)r_y g_{i,j+1} + r_x r_y g_{i+1,j+1}$$

ここで $i = [x/W]$，$j = [y/H]$，$r_x = (x - iW)/W$，$r_y = (y - jH)/H$ である．(1) 式で与えられる片側ラドン変換の計算においては，格子点へのアクセスに規則性がない．したがって，格子点の画素値をメモリに保存しなければならない上に，メモリへのランダムアクセスが必要である．これは，計算時間とメモリ領域の増加を引き起こす．

計算時間とメモリ領域を減らすために，片側ラドン変換の並列計算を構成する．まず，格子点へのアクセス順序を変更する．格子点における画素値は，処理部に一つずつ順に送られる．そこで，ハフ変換における投票の概念を導入する．格子点 $P_{i,j}$ の座標を $(x_{i,j}, y_{i,j})$ で表す．角度の範囲 $[0, 2\pi]$ を K 分割し，$\Delta\theta = 2\pi/K$，$\theta_k = k\Delta\theta$ $(k = 0, 1, \cdots, K-1)$ とする．パラメータ ρ の間隔を $\Delta\rho$ で表し，$\rho_h = h\Delta\rho$ $(h = 0, \pm1, \pm2, \cdots)$ とする．片側ラドン変換 $U(\rho, \theta)$ を，$\theta = \theta_k$，$\rho = \rho_h$ のみで，離散的に求める．積分路を定めるパラメータ ρ, θ, ξ を用いて，座標 x, y は次式のように表される．

$$x = \xi\cos\theta - \rho\sin\theta,$$
$$y = \xi\sin\theta + \rho\cos\theta.$$

ただし，ξ の値は正または 0 でなくてはならない．上式より

$$\xi = x\cos\theta + y\sin\theta,$$
$$\rho = -x\sin\theta + y\cos\theta.$$

座標 x, y と角度 θ の値が与えられると，上式により ξ と ρ の値を計算することができる．計算した ξ の値が正または 0 である場合，画素値 $g(x, y)$ の値が積分値 $U(\rho, \theta)$ に加算されている．したがって，ξ の値が正または 0 であるならば，積分値 $U(\rho, \theta)$ を画素値 $g(x, y)$ の値だけ増加させればよい．結局，片側ラドン変換における投票は，Table 1に示すようにまとめられる．

Table 1に示す投票の過程では，格子点の画素値は一つずつ順番にアクセスされる．CCD カメラは，格子点の画素値を一つずつ処理部に送る．したがって，各々の画素値が順番に処理部に送られている限り，画素値をメモリ上に記憶させる必要はなく，Table 1に示す投票により，片側ラドン変換を計算することができる．

Table 1: Voting in computation of one-sided Radon transform

initialize array $U(\theta_k, \rho_h)$
for $(i, j) = (0, 0), (0, 1), \cdots, (W-1, H-1)$ do
  compute $x_{i,j}$ and $y_{i,j}$
  for $k = 0, 1, \cdots, K-1$ do
    $\theta_k = k\Delta\theta$, $C_k = \cos\theta_k$, $S_k = \sin\theta_k$
    $\xi = x_{i,j}C_k + y_{i,j}S_k$
    $\rho = -x_{i,j}S_k + y_{i,j}C_k$
    if $\xi \geq 0$ then
      $h = \rho/\Delta\rho$
      increase $U(\theta_k, \rho_h)$ by pixel value $g_{i,j}$
    end
  end
end

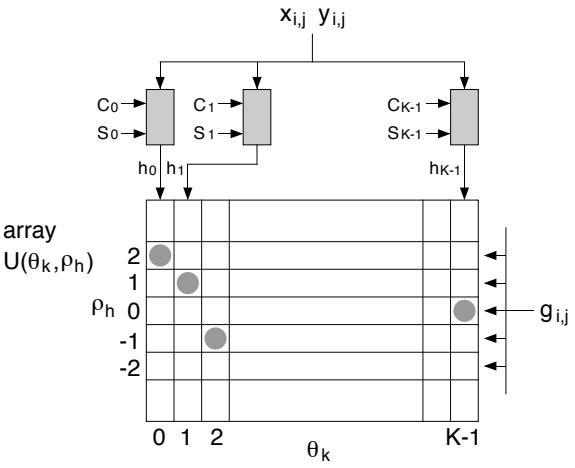


Fig.4: Parallel voting of one-sided Radon transform

また，各々の角度に関して並列に，投票を実行することができる．角度 θ_k に関する投票においては，配列 $U(\rho, \theta)$ は，$\theta = \theta_k$ のおいてのみアクセスされる可能性がある．すなわち，配列 $U(\rho, \theta)$ へのアクセスにおいては，異なる角度に関する投票はたがいに干渉しない．座標 $x_{i,j}, y_{i,j}$ と $C_k = \cos\theta_k$，$S_k = \sin\theta_k$ から，角度 θ_k に対応する添字 h を並列に計算することができる．画素値 $g_{i,j}$ を増やす操作も，角度 $\theta_0, \theta_1, \cdots, \theta_{K-1}$ に対して並列に実行することができる．したがって，Fig.4 に示すように，投票を並列に実行することができる．

並列アルゴリズムにおけるデータの流れを，Fig.5に示す．パワースペクトルの二次元マッチングにより回転角と移動方向を，片側ラドン変換の一次元マッチングにより移動距離を算出する．

5. 評価実験

本節では，オリジナルアルゴリズムと比較することにより，並列アルゴリズムを評価する．二つのアルゴ

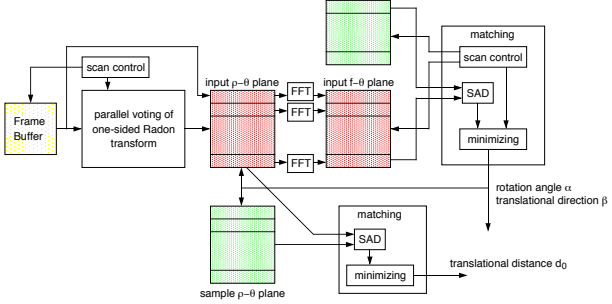


Fig.5: Data flow in parallel algorithm

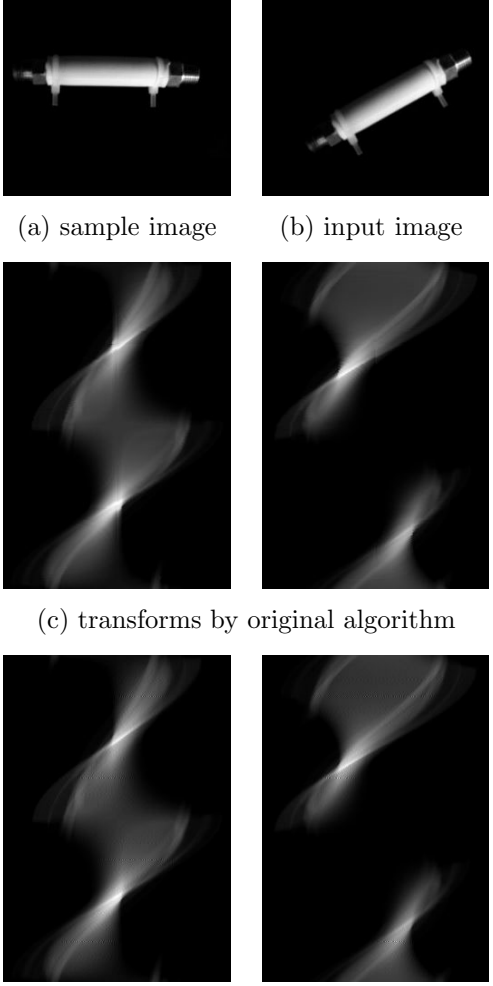
Table 2: Computed rotation angles

	original		parallel	
actual	computed	error	computed	error
0°	−1°	−1°	−2°	−2°
30°	30°	0°	30°	0°
60°	64°	4°	64°	4°
90°	93°	3°	93°	3°
120°	122°	2°	119°	−1°
150°	148°	−2°	149°	−1°
180°	181°	1°	180°	0°
210°	219°	9°	218°	8°
240°	238°	−2°	245°	5°
270°	266°	−4°	277°	7°
300°	294°	−6°	299°	−1°
330°	324°	−6°	326°	−4°

リズムを，CPU が Pentium III 800 MHz，メモリが 256MB の PC 上に実装した．PC の OS は Vine Linux である．プログラムは C で記述し，GCC ver.2.7.2.3 でコンパイルした．Fig.6-(a)，(b) に参照画像と入力画像の例を示す．画像は 8 [bit] グレースケール画像である．サイズは 256 [pixel] × 256[pixel] であり，画像の中央を座標原点とする．Fig.6-(c) に，それぞれの画像に対してオリジナルアルゴリズムで計算した片側ラドン変換を示す．Fig. 6-(d) に，それぞれの画像に対して並列アルゴリズムで計算した片側ラドン変換を示す．パラメータ θ の間隔を $\Delta\theta = 1$ [degree] とする．パラメータ ρ の範囲を $[-125, 125]$ [pixel] とし，その間隔を $\Delta\rho = 1$ [pixel] とする．Fig. 6-(c) と (d) を比較すると，並列アルゴリズムは片側ラドン変換を正確に計算していることがわかる．

物体の位置と姿勢を算出するために，片側ラドン変換のパワースペクトルを計算する．この例では，周波数 $f_h = h/125$ $(h = -125, -124, \cdots, 125)$ におけるパワースペクトルを求めている．周波数 f_0 に近い低周波の領域では，パワースペクトルが安定に計算できるのに対し，高周波の領域ではノイズが多く，数値的に不安定である．そこで，パワースペクトルのマッチングにおいては，次式のノルムを用いる．

$$\| \mathcal{F}_{sample}(f_h, \theta_{sample}) - \mathcal{F}_{input}(f_h, \theta_{input}) \| = \sum_{h=0}^{40} |\mathcal{F}_{sample}(f_h, \theta_{sample}) - \mathcal{F}_{input}(f_h, \theta_{sample})|.$$

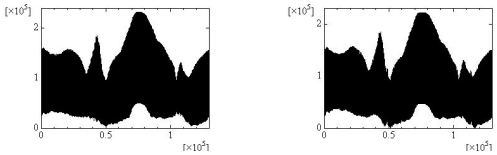


(a) sample image (b) input image

(c) transforms by original algorithm

(d) transforms by parallel algorithm

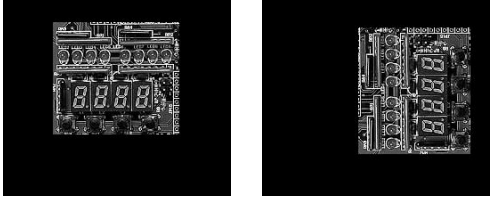
Fig.6: Computed transforms for tube image



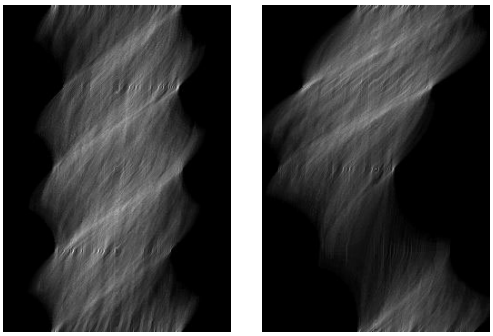
(a) original algorithm (b) parallel algorithm

Fig.7: Matching between power spectrums corresponding to tube images

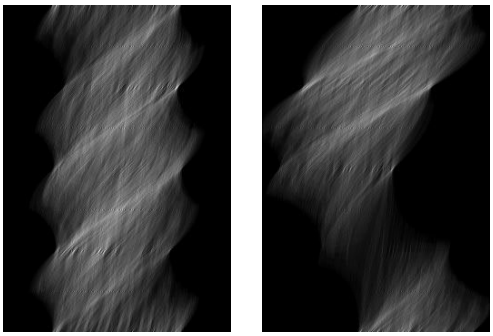
Fig.6は，参照画像に対応するパワースペクトルと入力に対応するパワースペクトルのノルムを表す．Fig.6-(a) においては，オリジナルアルゴリズムによる計算結果を用い，Fig.6-(b) においては並列アルゴリズムによる計算結果を用いている．ノルム $\| \mathcal{F}_{sample}(f_h, \theta_{sample}) - \mathcal{F}_{input}(f_h, \theta_{input}) \|$ を，$\theta_{sample} = 0, 1, \cdots, 359$ [degree]，$\theta_{input} = 0, 1, \cdots, 359$ [degree] に対して計算する．図は，得られた 360×360 個の計算結果を一列に並べて得られるグラフである．Fig.7-(a) より，ノルムは $\theta_{sample} = 292$ [degree] ，$\theta_{input} = 322$ [degree] において最小値を取ることがわかる．したがって，$\alpha = 30$ [degree] ならびに $\beta = 52$ [degree] である．さらに $d_0 = 40$ [pixel] が得られる．Fig. 7-(b) より，ノルムは $\theta_{sample} = 113$ [degree]，$\theta_{input} = 143$ [degree] にお



(a) sample image (b) input image

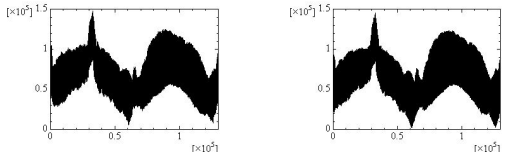


(c) transforms by original algorithm



(d) transforms by parallel algorithm

Fig.8: Computed transforms for board image



(a) original algorithm (b) parallel algorithm

Fig.9: Matching between power spectrums corresponding to board images

いて最小値を取ることがわかる．したがって，$\alpha = 30$ [degree] ならびに $\beta = 233$ [degree] である．さらに $d_0 = -39$ [pixel] が得られる．結局，並列アルゴリズムにより計算された物体の位置と姿勢は，オリジナルアルゴリズムによる計算結果と一致していることがわかる．

Fig.8-(a)，(b) に別の例を示す．画像は 8 [bit] グレースケール画像であり，サイズは 256 [pixel] × 256 [pixel] である．オリジナルアルゴリズムで計算した片側ラドン変換を Fig.8-(c) に，並列アルゴリズムで計算した片側ラドン変換を Fig.8-(d) に示す．Fig.8-(c) と (d) を比較すると，並列アルゴリズムは片側ラドン変換を正確に計算していることがわかる．Fig.8-(c) に示す片側ラドン変換に対して計算されたパワースペクトルのノルムを，Fig.9-(a) に示す．図より，$\theta_{sample} = 258$ [degree]，$\theta_{input} = 349$ [degree] においてノルムが最小になることがわかる．したがって，$\alpha = 91$ [degree] ならびに $\beta = 79$ [degree] である．さらに，$d_0 = 49$ [pixel] が得られる．Fig.8-(d) に示す片側ラドン変換に対して計算されたパワースペクトルのノルムを，Fig.9-(b) に示す．図より，$\theta_{sample} = 259$ [degree]，$\theta_{input} = 349$ [degree] においてノルムが最小になることがわかる．したがって，$\alpha = 90$ [degree] ならびに $\beta = 79$ [degree] である．さらに，$d_0 = 49$ [pixel] が得られる．この例においても，並列アルゴリズムにより計算された物体の位置と姿勢は，オリジナルアルゴリズムによる計算結果と一致していることがわかる．

次に，実環境においてオリジナルアルゴリズムと並列アルゴリズムを評価する．Fig.10にエアーテーブル上の物体の例を示す．画像の幅は 320 [pixel]，高さは 240 [pixel] である．画像の中央を座標原点とする．パラメータ θ の間隔を $\Delta\theta = 1$ [degree] とする．パラメータ ρ の範囲を $[-256, 256]$ [pixel] とし，その間隔を $\Delta\rho = 2$ [pixel] とする．CCD カメラにより得られる 8 [bit] グレースケール画像を g_{camera}，あらかじめ撮影された背景画像を g_{back} とする．入力画像 g_{input} を，以下に示す手続きで求める．まず，カメラ画像と背景画像の絶対差分を計算する．すなわち，

$$g_{ad}(x, y) = \mid g_{camera}(x, y) - g_{back}(x, y) \mid .$$

入力画像 g_{input} は，絶対差分画像 g_{ad} より次式にした

Fig.10: Sample image of object on air floating table

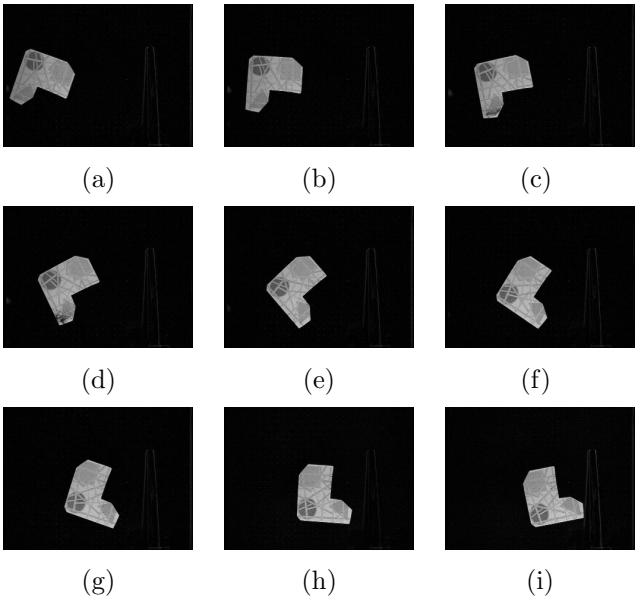
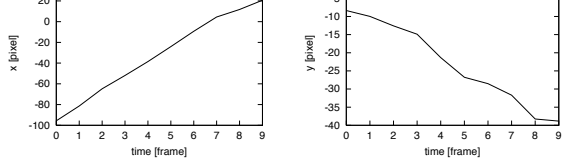
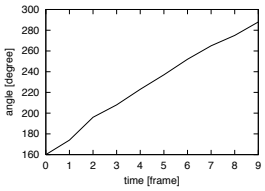


Fig.11: Successive images of object moving on air floating table

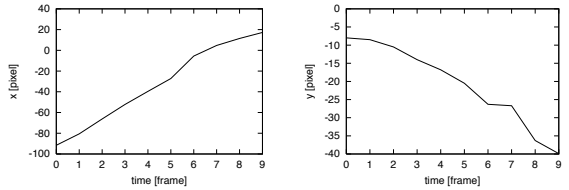


(a) position

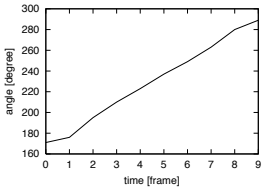


(b) orientation

Fig.12: Position and orientation computed by original algorithm



(a) position



(b) orientation

Fig.13: Position and orientation computed by parallel algorithm

がって計算される．

$$g_{input}(x,y) = \begin{cases} g_{ad}(x,y) & g_{af}(x,y) \geq 90 \\ 0 & otherwise \end{cases} .$$

姿勢の計算結果の精度を評価する．Fig.10に示す物体を，テーブル上に正確な姿勢で置く．実際の姿勢角度と二つのアルゴリズムにより計算される姿勢角度を，Table 2に示す．Table 2より，並列アルゴリズムにより得られる姿勢角度は，オリジナルアルゴリズムにより得られる角度とほぼ一致していることがわかる．ただし，絶対誤差は 10 [degree] 近くに達する．

Fig.11-(a)〜(h) に，エアーテーブル上の物体の運動を示す．これらの画像はビデオフレームレートで取得している．物体とテーブル間の摩擦は無視できるので，物体はテーブル上を等速度・等角速度で運動する．これらの画像から，オリジナルアルゴリズムで計算した物体の位置と姿勢を Fig.12に，並列アルゴリズムで計算した位置と姿勢を Fig.13に示す．並列アルゴリズムによる計算結果は，オリジナルアルゴリズムによる計算結果とほぼ一致している．さらに，物体の速度と角速度は，ほぼ一定であることがわかる．したがって，並列アルゴリズムは，オリジナルアルゴリズムと同程度

6. おわりに

本報告では，平面運動物体の位置と姿勢を検出するために，片側ラドン変換の並列アルゴリズムを構成した．まず，片側ラドン変換を用いて物体の位置と姿勢を検出するアルゴリズムを述べ，アルゴリズムの並列性を指摘した．次に，並列投票の概念を導入し，片側ラドン変換の並列アルゴリズムを構築した．さらに，並列アルゴリズムをオリジナルアルゴリズムと比較した．その結果，並列アルゴリズムは，オリジナルアルゴリズムと同程度に平面運動物体の位置と姿勢を検出できることがわかった．ただし，絶対誤差が 10 [degree] 近くに達しており，アルゴリズムの改良が必要である．

現在，ビデオフレームレートで実行することを目標として，提案した並列アルゴリズムを FPGA 上に実装している．さらに，平面運動物体の位置と姿勢を検出するビジョンシステムを構築し，位置と姿勢がランダムな物体のハンドリングに適用する予定である．

謝辞

本研究は，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．

【参考文献】

1) 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

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

3) 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

4) 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

5) 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

6) 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

7) 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

8) 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