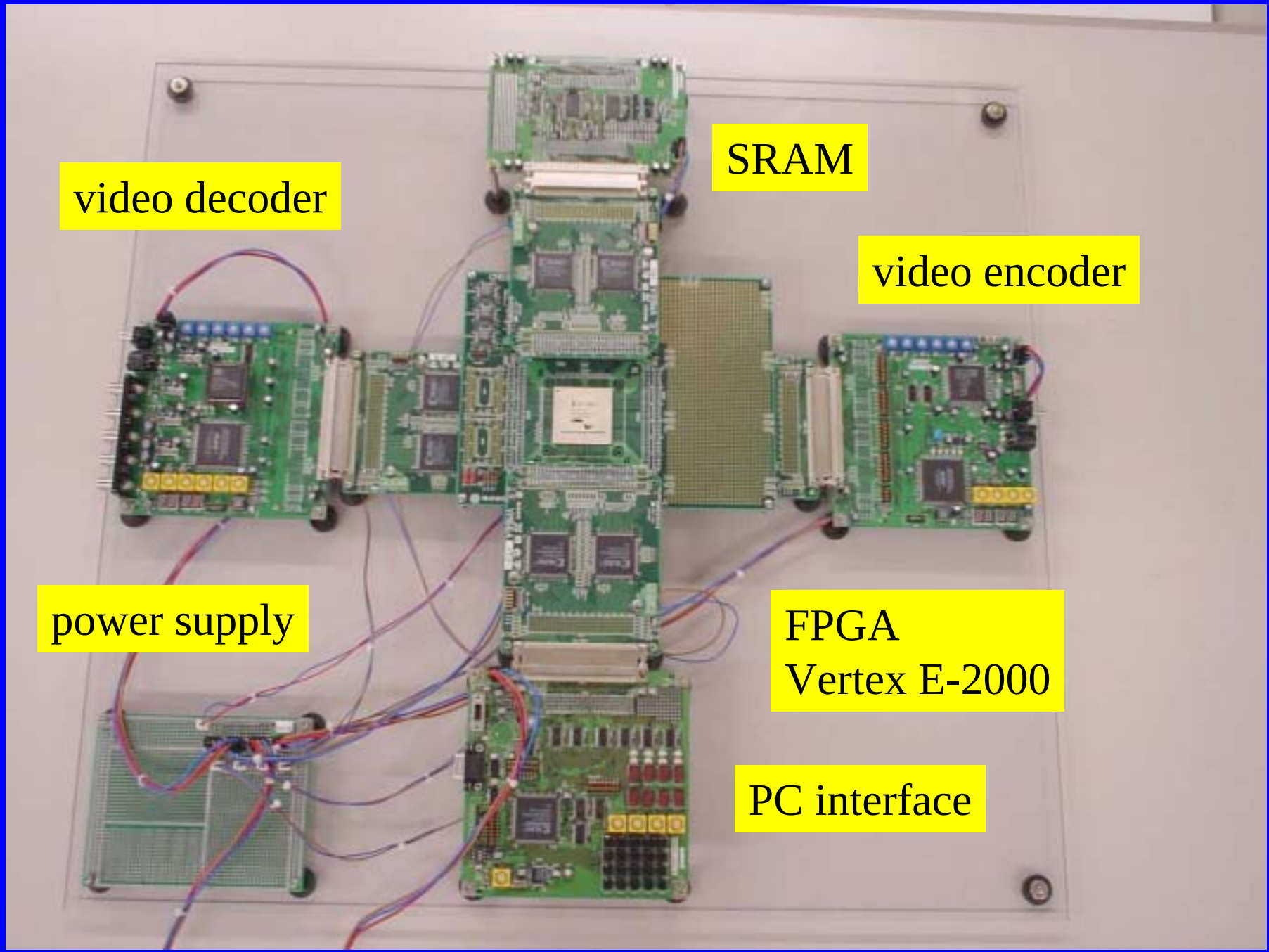
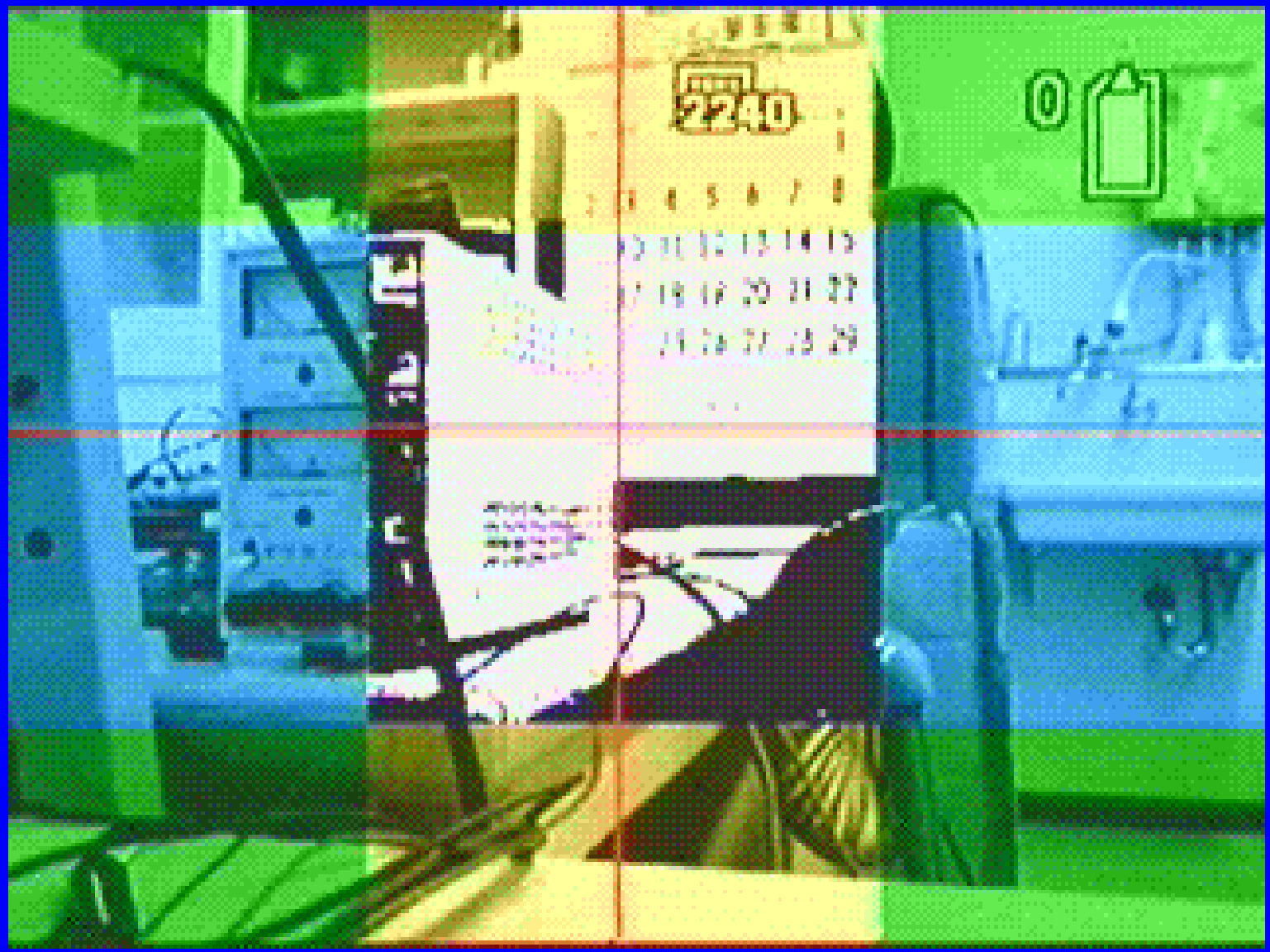
1/28, 2002

第5回　ヒト･機械コミュニケーションチップ開発研究会

ＦＰＧＡによるビジョンアルゴリズムのリアルタイム実装

平井慎一
立命館大学ロボティクス学科
ＶＬＳＩセンター

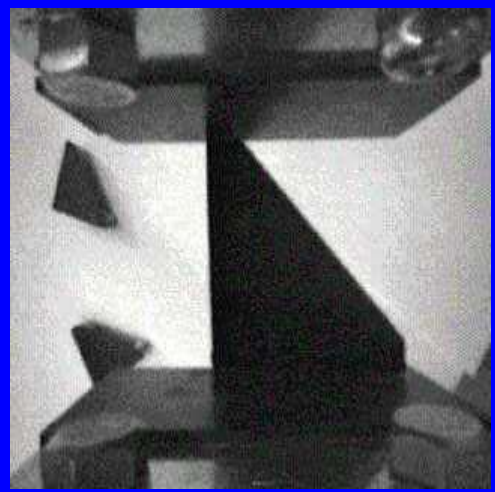

video decoder
SRAM
video encoder
power supply
FPGA
Vertex E-2000
PC interface



ロボティクス　Robotics

感覚と運動の結合

Intelligent Connection
from Perception to Action



ロボティクスにおけるＶＬＳＩ技術

運動　　△

逆動力学計算　　制御には不要

逆運動学，ヤコビ行列の計算

ＰＣで1msec（十分）

感覚　　◎

二次元時系列信号処理

視覚（リアルタイムビジョン）

触覚（分布圧センサ）

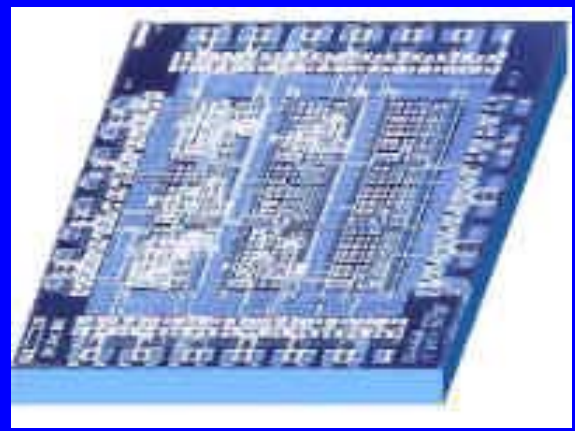
聴覚（マイクロホンアレイ）

感覚情報処理のチップ化

ビジョンアルゴリズム
- 運動検出
- 変形認識
- ステレオビジョン

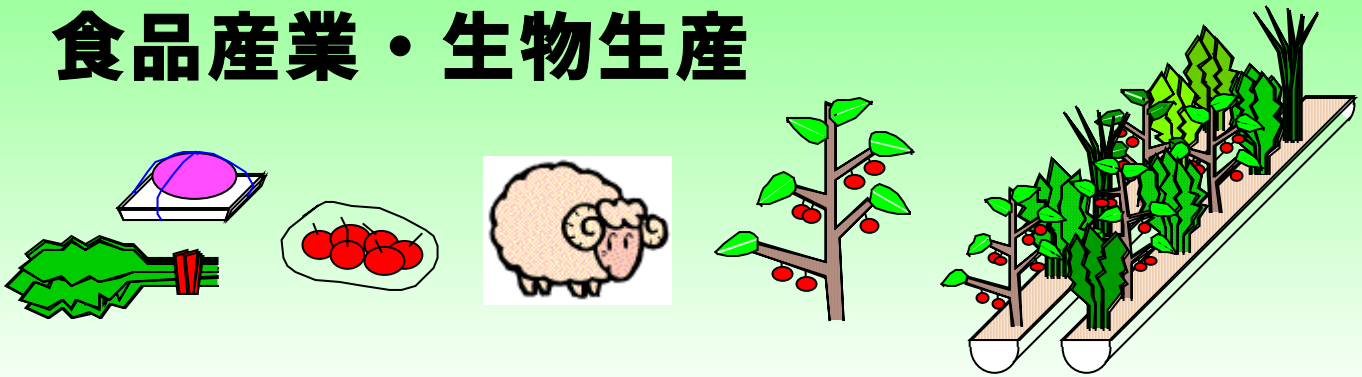
多点オーディオ信号処理
- 音源定位
- 話者認識

多点触覚信号処理
- 手触り認識



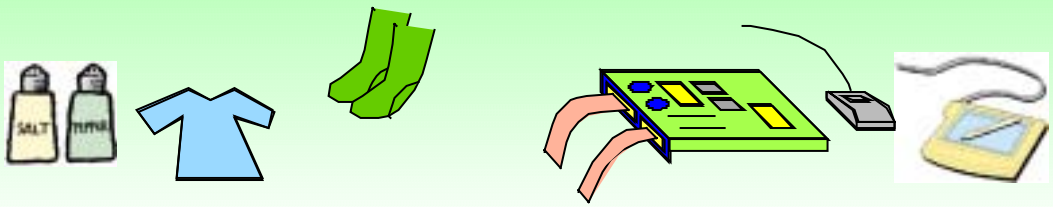
研究開発の背景

食品産業・生物生産

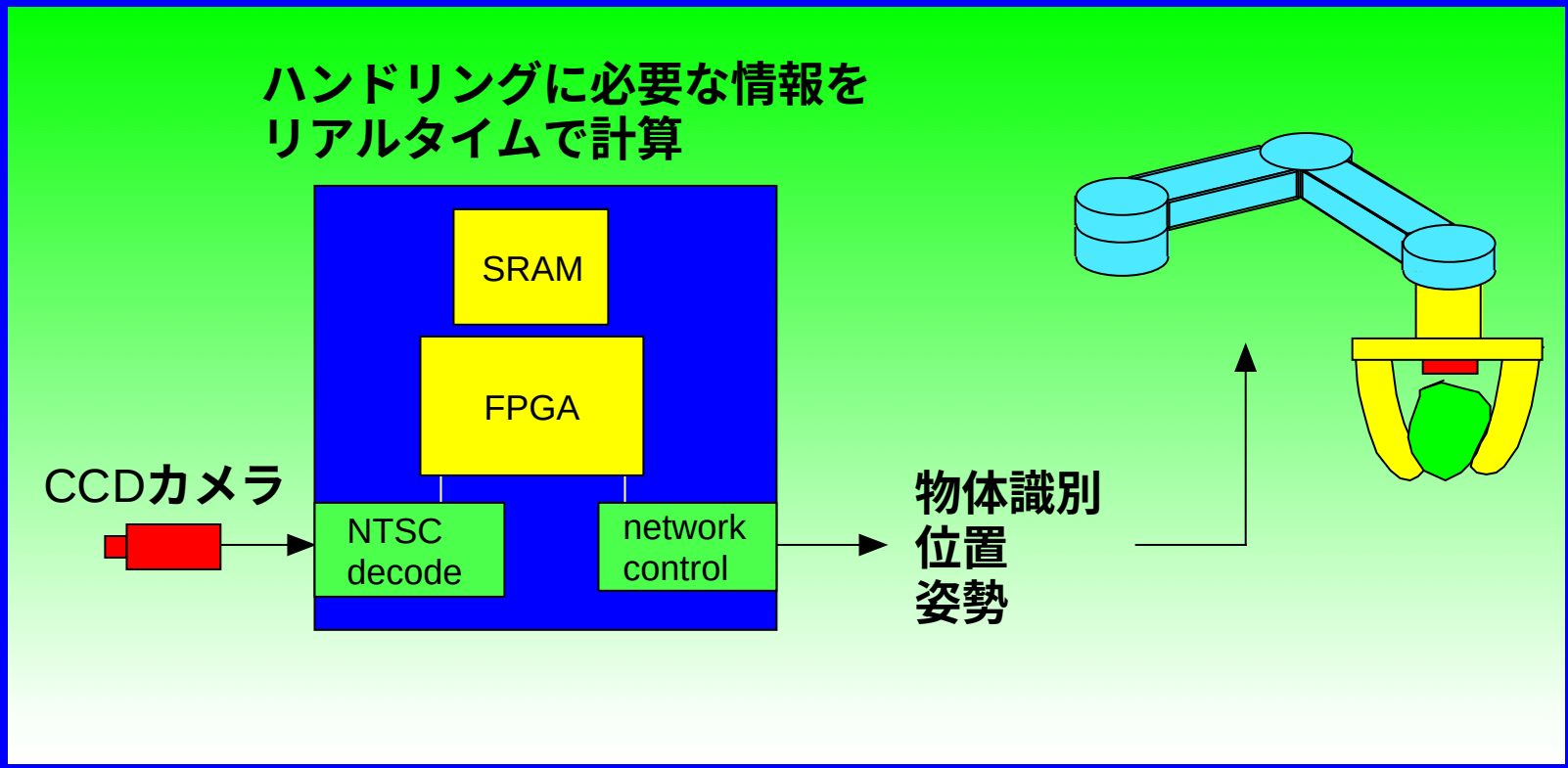


柔軟変形物ハンドリング

リサイクル産業・インバース生産



研究開発の目標



柔軟変形物ハンドリング用ビジョンシステム

目標達成の障害

- 平面運動検出と物体識別
  (トラッキングでは不十分)
- 重なり，変形に対応
- リアルタイム
  (ビデオフレームレート)

解決法

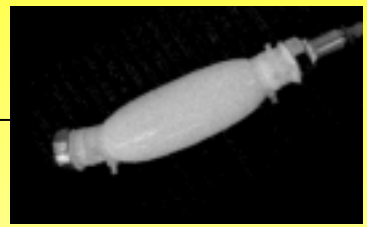
- ラドン変換法
- 並列化とＦＰＧＡへの実装

研究開発の概要

新しいビジョンチップ

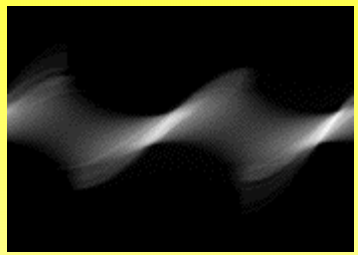
片側ラドン変換法
ハフ・フーリエ変換法

CCDカメラ

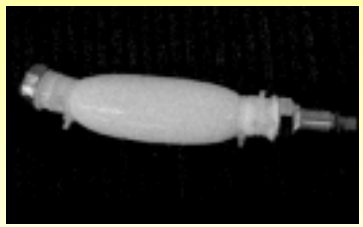


入力画像

変換

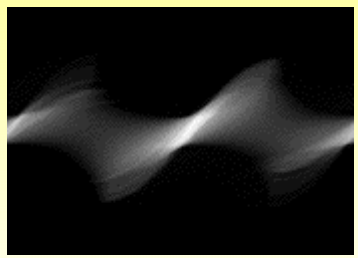


比較



参照画像

変換



位置	○
姿勢	○
変形	○

平行移動

+

回転移動
拡大／縮小
剪断変形・膨張変形

内容

- アルゴリズムの紹介
- 技術的な背景
- 並列アルゴリズム
- 試作ＦＰＧＡボード

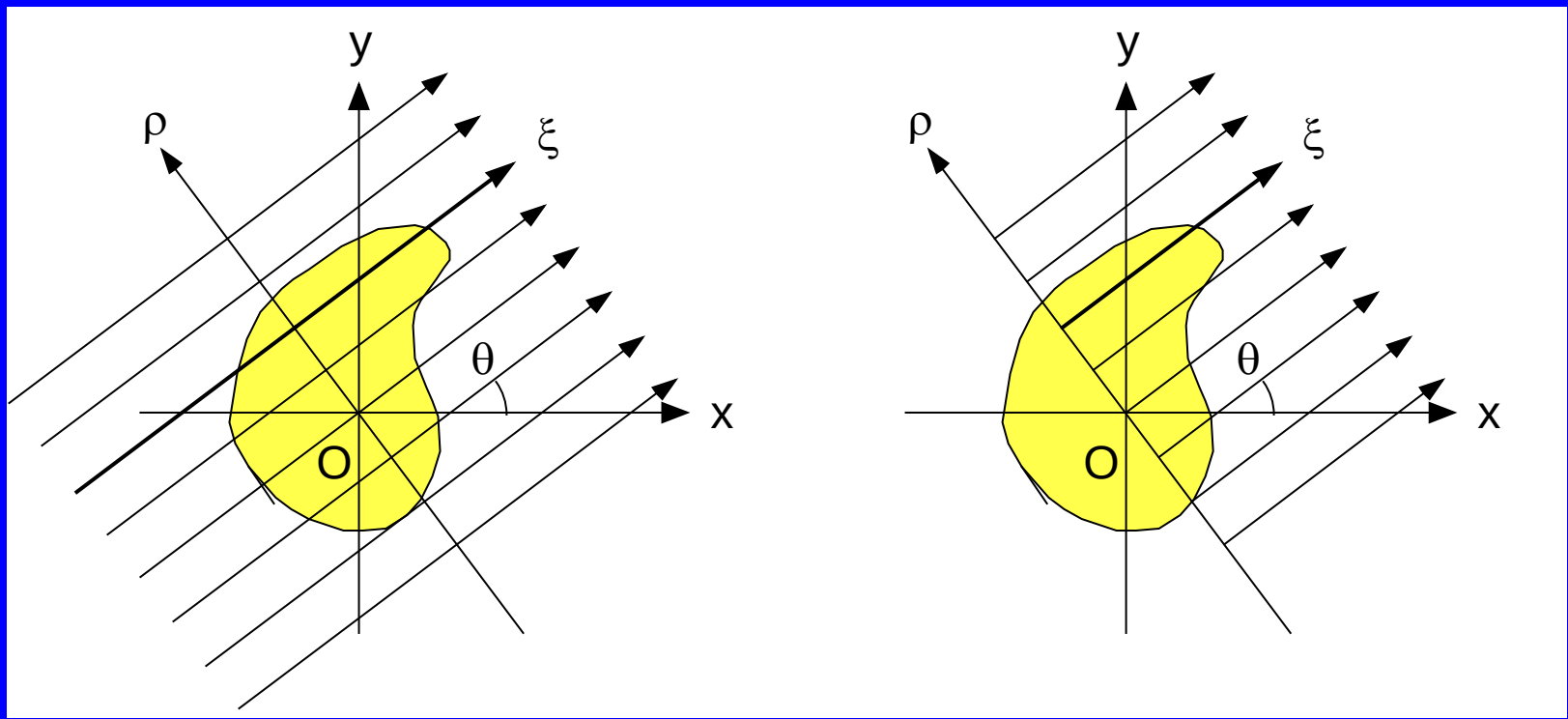
ロボット搭載用ビジョンチップ

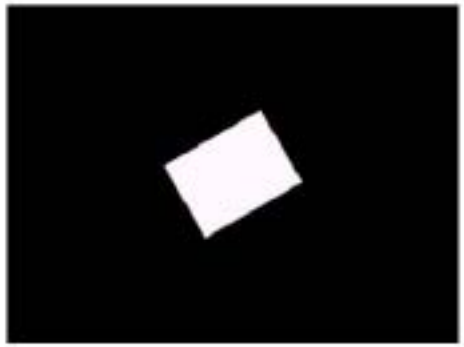
片側ラドン変換法
ハフ･フーリエ変換法

平面運動物体の識別
並進･回転運動計測

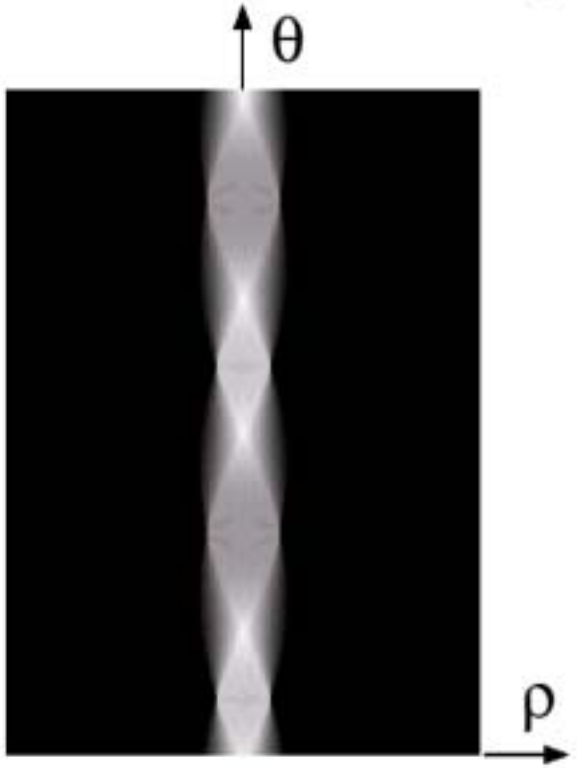
ビデオフレームレート
30 frame/sec

ラドン変換と片側ラドン変換





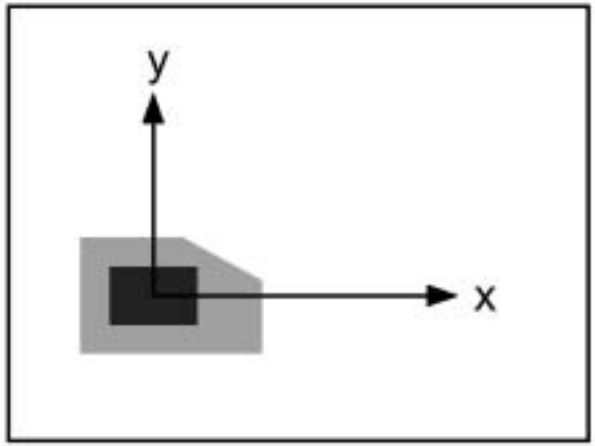
(a) image



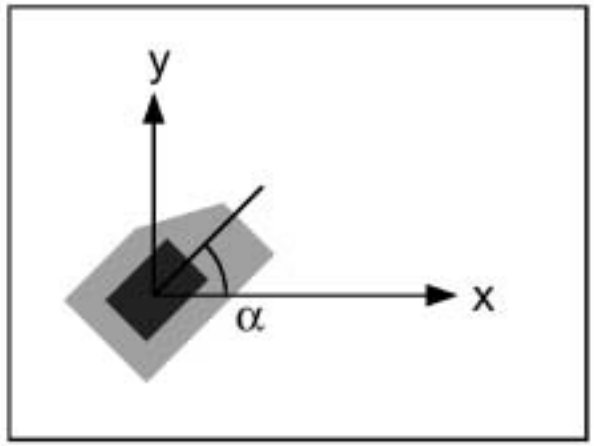
(b) Radon

(c) one-sided Radon

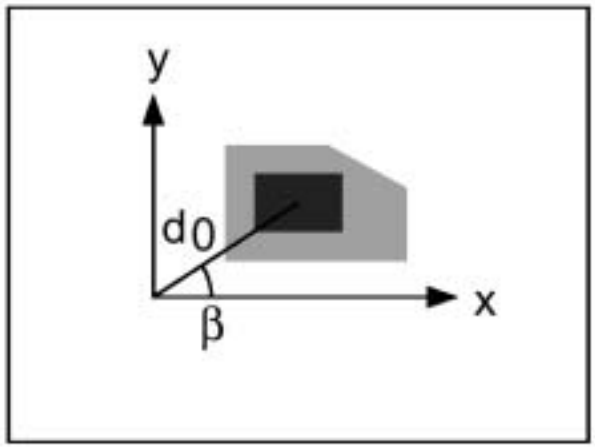
平面運動



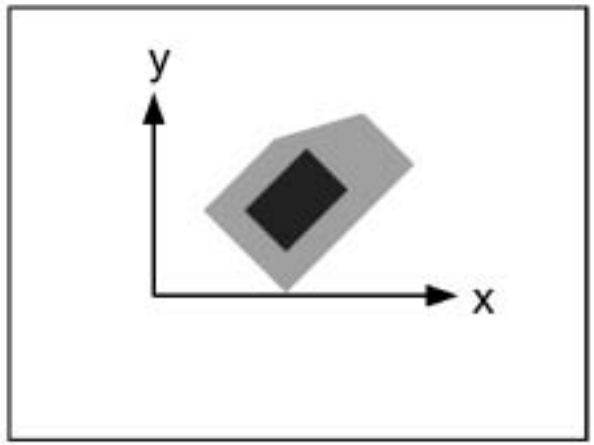
(a) original image g_0



(b) rotation g_1



(c) translation g_2



(d) planar motion g_3

ラドン変換の性質

$$R[g](\rho,\theta)=\int_{-\infty}^{\infty} g(\xi\cos\theta-\rho\sin\theta,\xi\sin\theta+\rho\cos\theta)d\xi.$$

$$R_0(\rho,\theta)\equiv R_3(\rho-d_0\sin(\theta-\beta),\theta+\alpha),\quad \forall\rho,\theta.$$

translation rotation

$$F(f,\theta)=\left|\int_{-\infty}^{\infty} R(\rho,\theta)e^{i\rho g}d\rho\right|^2.$$

$$F_0(f,\theta)\equiv F_3(f,\theta+\alpha),\quad \forall f,\theta.$$

rotation

片側ラドン変換の性質

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

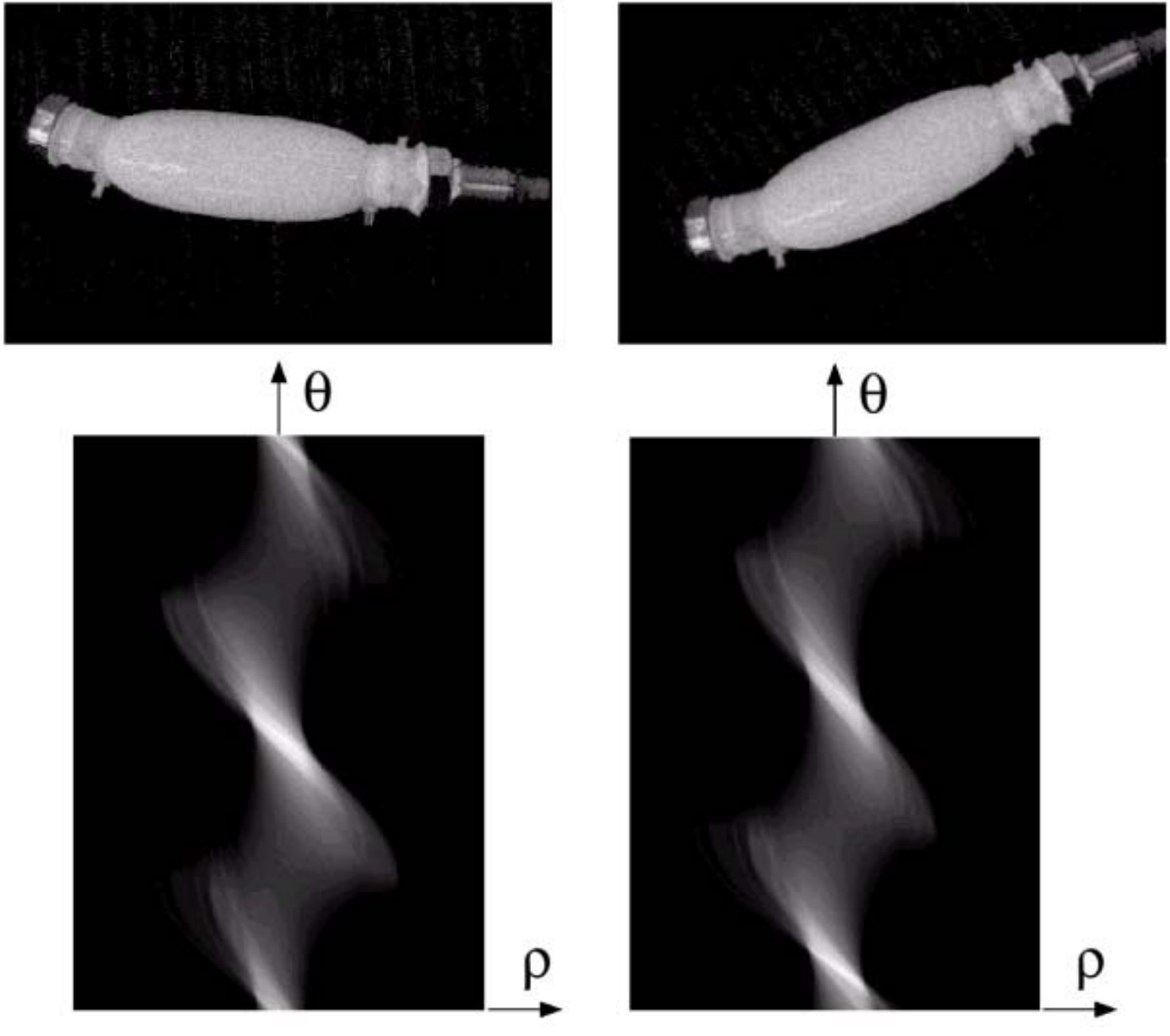
$$U_0(\rho,\theta-\alpha)\equiv U_3(\rho-d_0\sin(\theta-\beta),\theta),$$

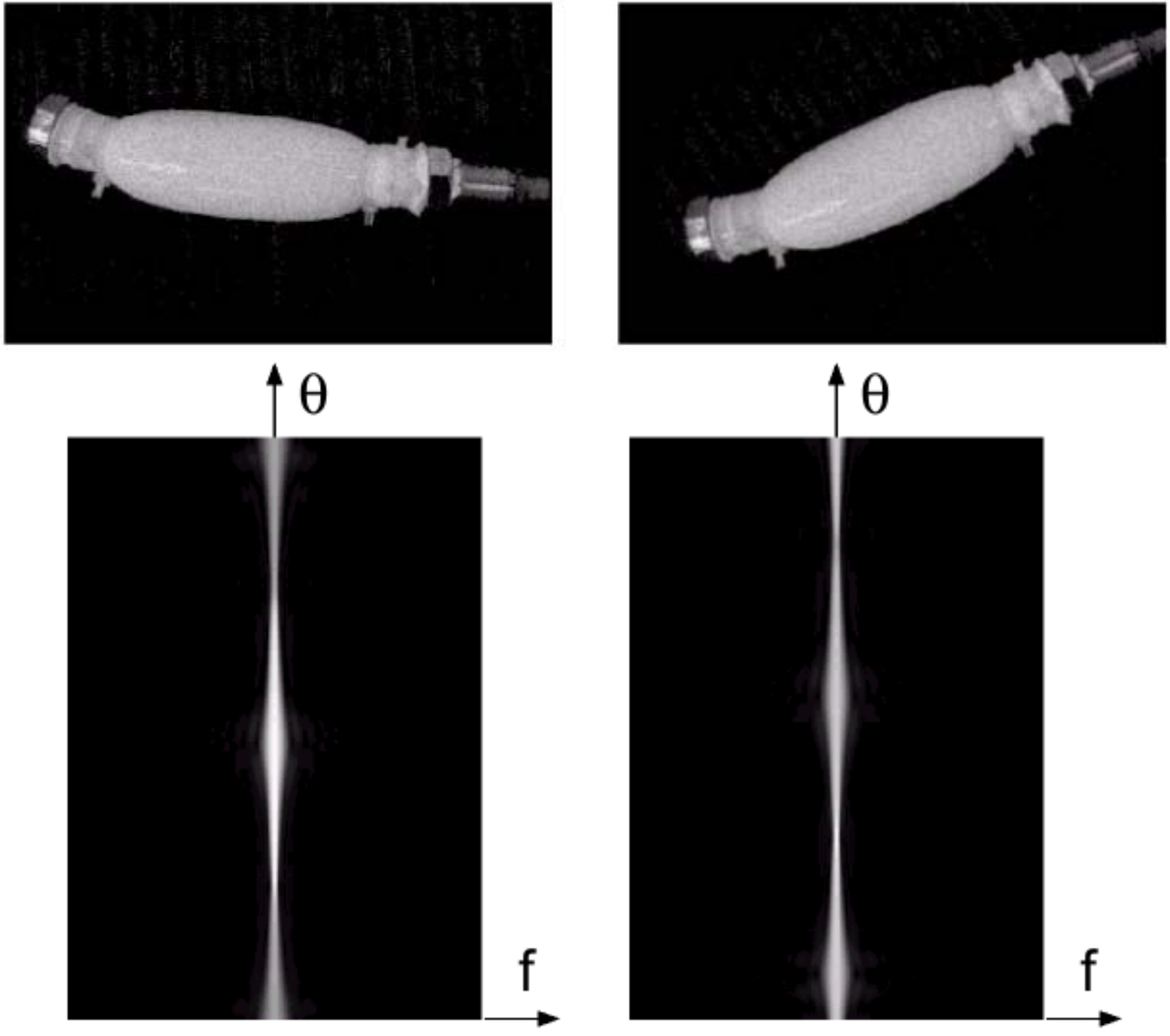
rotation　　translation

$$\forall\rho,\quad \theta+\frac{\pi}{2}=\beta.$$

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

rotation


θ
ρ
θ
ρ


θ
f
θ
f

Algorithm 1

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
$$\| \mathcal{F}_{sample}(f, \theta_{sample}) - \mathcal{F}_{input}(f, \theta_{input}) \|.$$

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

Step 5 Find d_0 that satisfies
$$U_{sample}(\rho, \theta_{sample}) \equiv U_{input}(\rho + d_0, \theta_{input}) \quad \forall \rho.$$

Algorithm 2

Step 1 Compute gravity centers G_{sample} and G_{input}.

Step 2 Compute $U_{sample}(\rho, \theta)$ and $U_{input}(\rho, \theta)$ around individual gravity centers.

Step 3 Compute $\mathcal{F}_{sample}(f, \theta)$ and $\mathcal{F}_{input}(f, \theta)$.

Step 4 Find α that satisfies
$\mathcal{F}_{sample}(f, \theta - \alpha) \equiv \mathcal{F}_{input}(f, \theta) \quad \forall f, \theta.$

Algorithm 3

Step 1 Compute gravity centers G_{sample} and G_{input}.

Step 2 Compute $U_{sample}(0, \theta)$ and $U_{input}(0, \theta)$ around individual gravity centers.

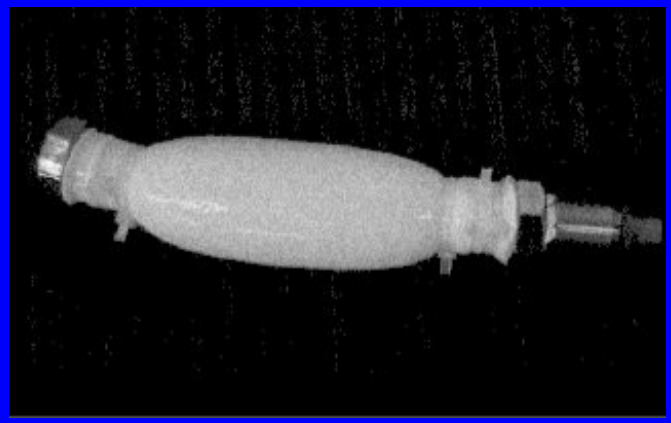
Step 3 Find α that satisfies
$U_{sample}(0, \theta - \alpha) \equiv U_{input}(0, \theta) \quad \forall \theta.$

アルゴリズムの比較

- Algorithm1
  重心計算が不要　　任意形状
  計算時間 6.24sec
- Algorithm2
  重心計算が必要　　任意形状
  計算時間 5.92sec
- Algorithm 3
  重心計算が必要　　形状に制約
  計算時間 34.2msec

計算例

30度回転
X方向 20[pixel]
Y方向 10[pixel]



参照画像

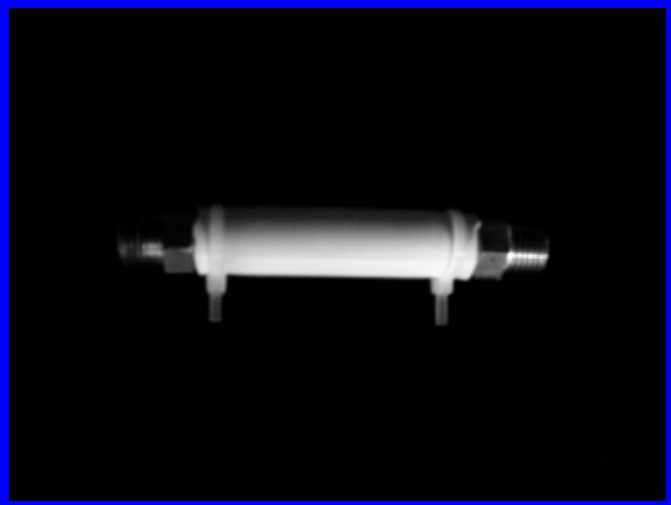
入力画像

検出結果

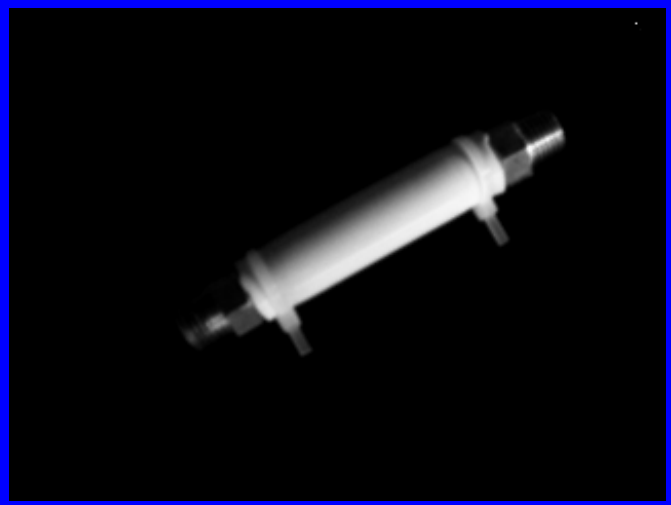
	並進(X)	並進(Y)	回転(deg)
Algorithm 1	19.6	9.99	30
Algorithm 2	13.6	18.9	30
Algorithm 3	13.6	18.9	30

計算例

30度回転
X方向 20[pixel]
Y方向 10[pixel]



参照画像

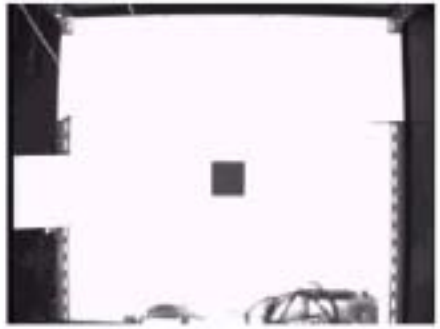


入力画像

検出結果

	並進(X)	並進(Y)	回転(deg)
Algorithm 1	19.6	9.99	30
Algorithm 2	20.1	10.1	30
Algorithm 3	20.1	10.1	30

(a) triangle



(b) square

(c) curved

act.	meas.	error
0°	0°	0°
30°	30°	0°
60°	61°	1°
90°	89°	−1°

(a) triangle

act.	meas.	error
0°	−1°	−1°
30°	30°	0°
60°	60°	0°

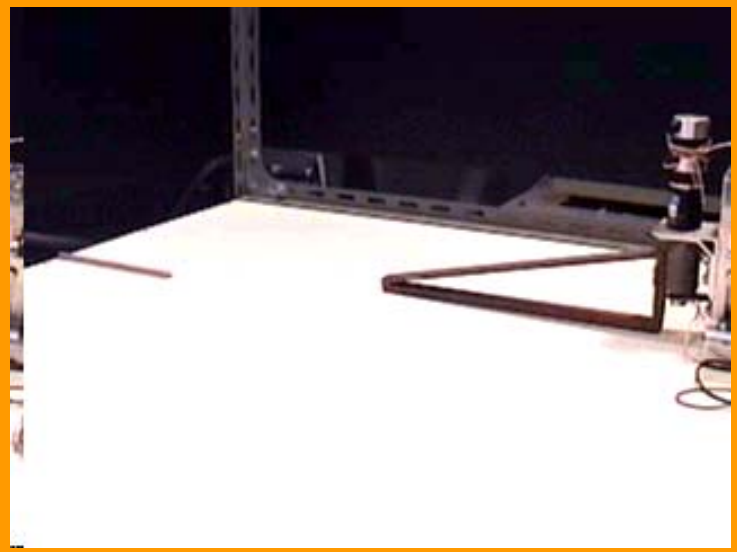
(b) square

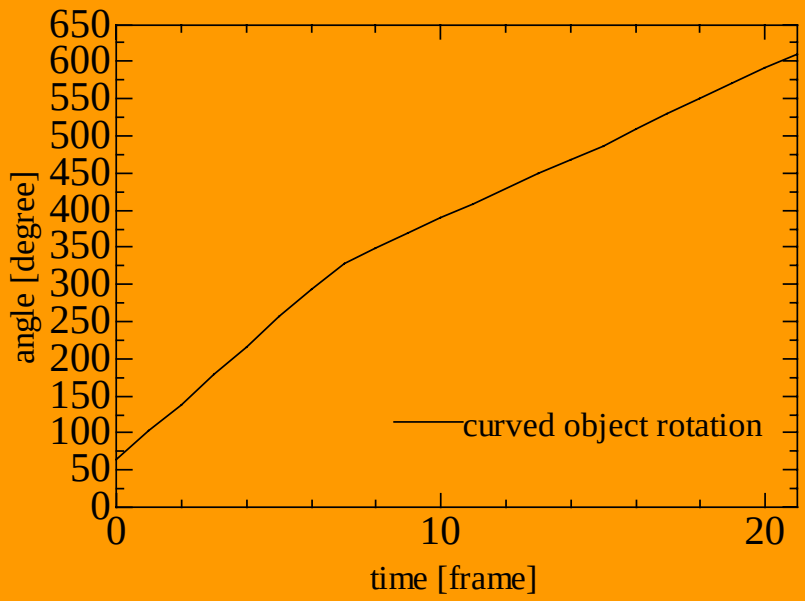
act.	meas.	error
0°	−1°	−1°
30°	29°	−1°
60°	58°	−2°
90°	85°	−5°
120°	114°	−6°
150°	140°	−10°

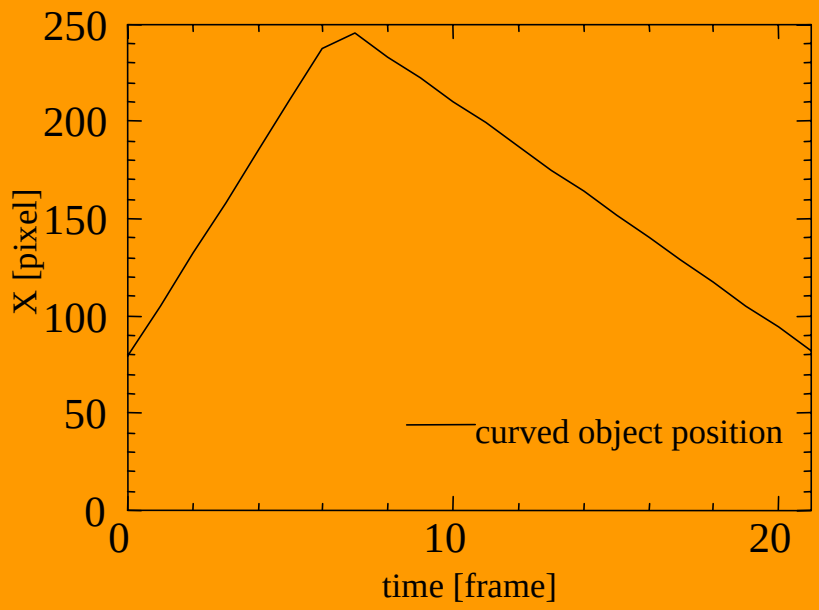
act.	meas.	error
180°	171°	−9°
210°	206°	−4°
240°	239°	−1°
270°	266°	−4°
300°	297°	−3°
330°	326°	−4°

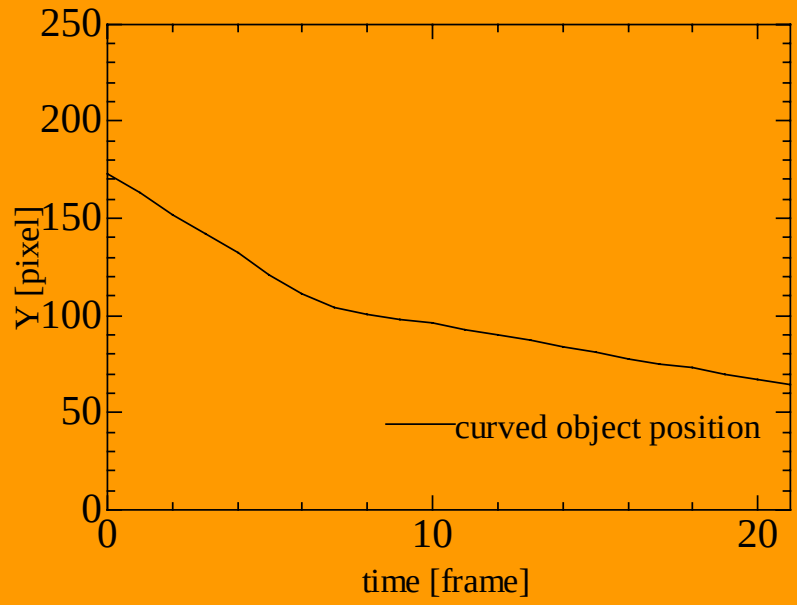
(c) curved

エアテーブル上の物体の計測





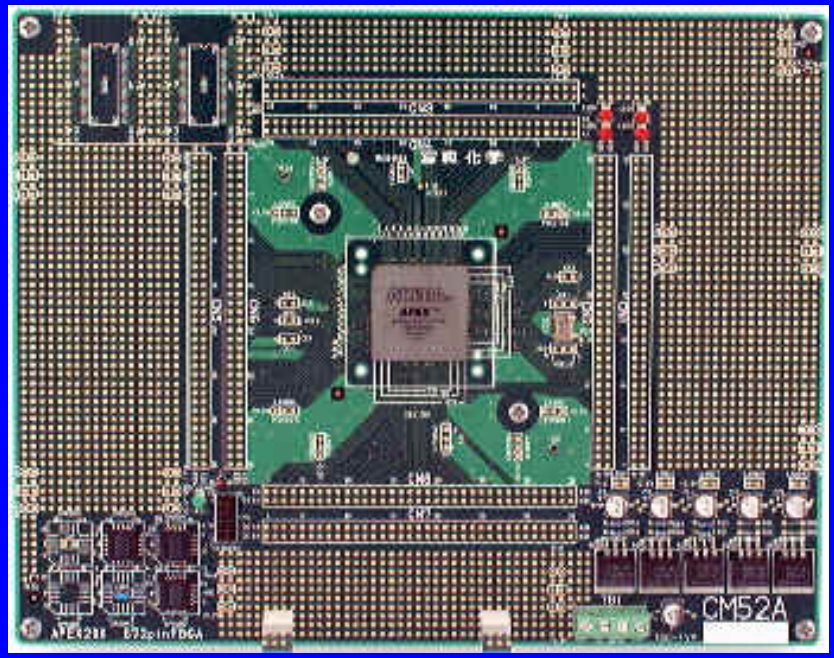




ソフトウェア vs ハードウェア

	ソフトウェア （プログラム）	ハードウェア （専用VLSI）
実行速度	△	○　（並列化）
経済性	○	×　（３００万個／月）
開発効率	△	×

FPGA
Field Programmable Gate Array



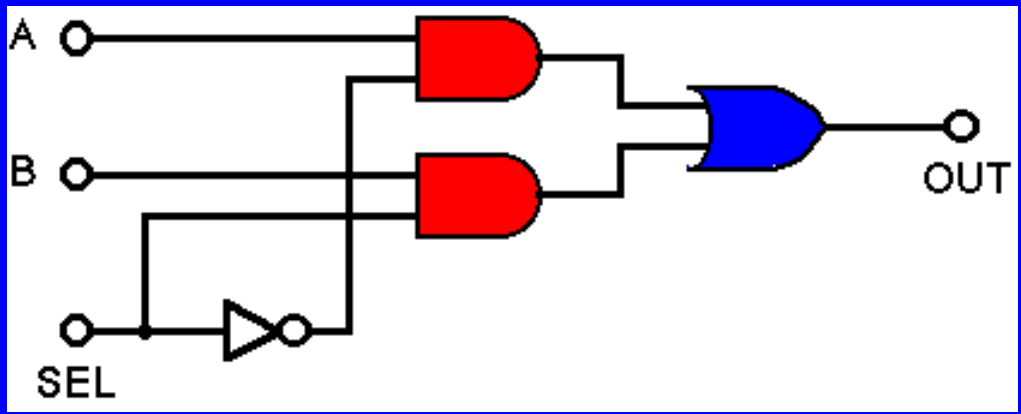
論理回路を書き換え可能
（プログラム可能）

Xilinx FPGA　APEX 20K

20K**ゲート**　200**万ゲート**

一年前　50**万ゲート**
数年前　**数万ゲート**

HDL
Hardware Description Language

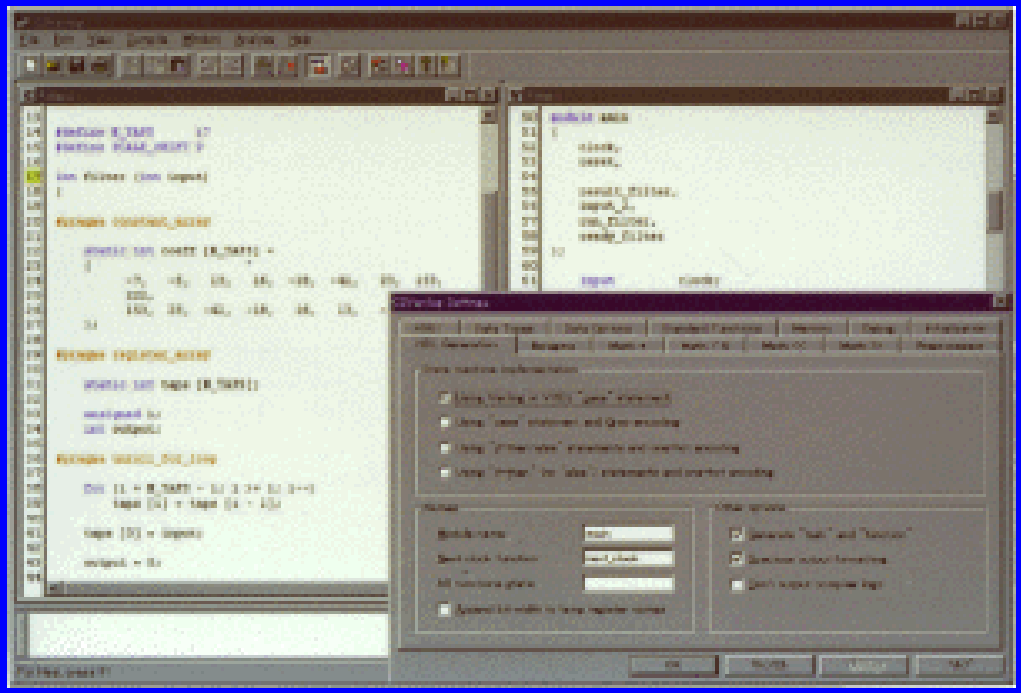


回路図
による記述

```
module SELECTER (A, B, SEL, OUT)
input    A, B, SEL;
output  OUT;
          assign  OUT = ~SEL & A | SEL & B;
endmodule
```

Verilog-HDL
による記述

Cレベル設計



C Level Design Co.Ltd
システムコンパイラ

C/C++

↓

HDL

↓

RTL

論理回路
FPGA配線

Cレベル設計

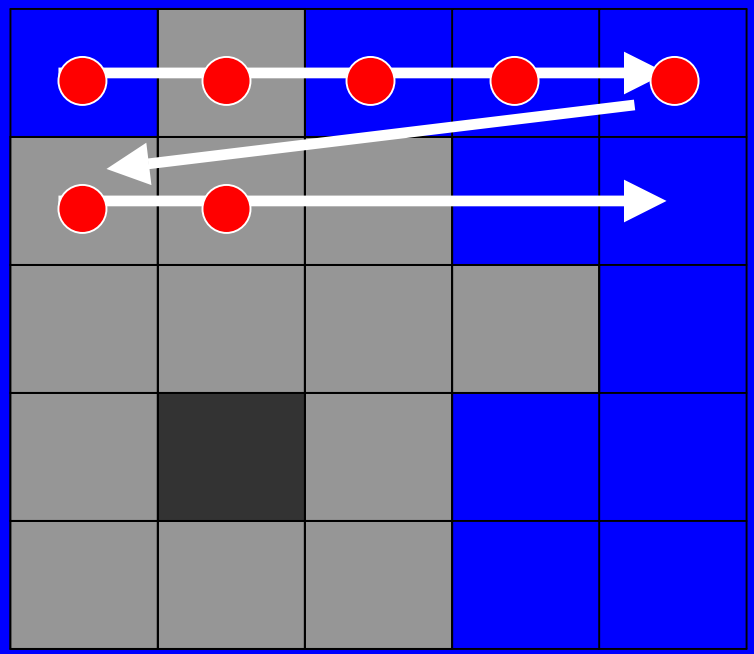
- Cycle C
  - Visual C/C++ compatible
  - PC上でコンパイル実行
  - トランスレータでHDLに変換
- Handel-C
  - C + Occum (par, alt, !)
  - CSP理論に基づく並列演算記述
  - 直接RTLに変換 (FPGA用のデータ)
- System C
- Spec C

ソフトウェア vs ハードウェア

	ソフトウェア (プログラム)	ハードウェア (専用VLSI)
実行速度	△	○　(並列化)
経済性	○	△　(一個でも可)
開発効率	△	△　(プログラミング)

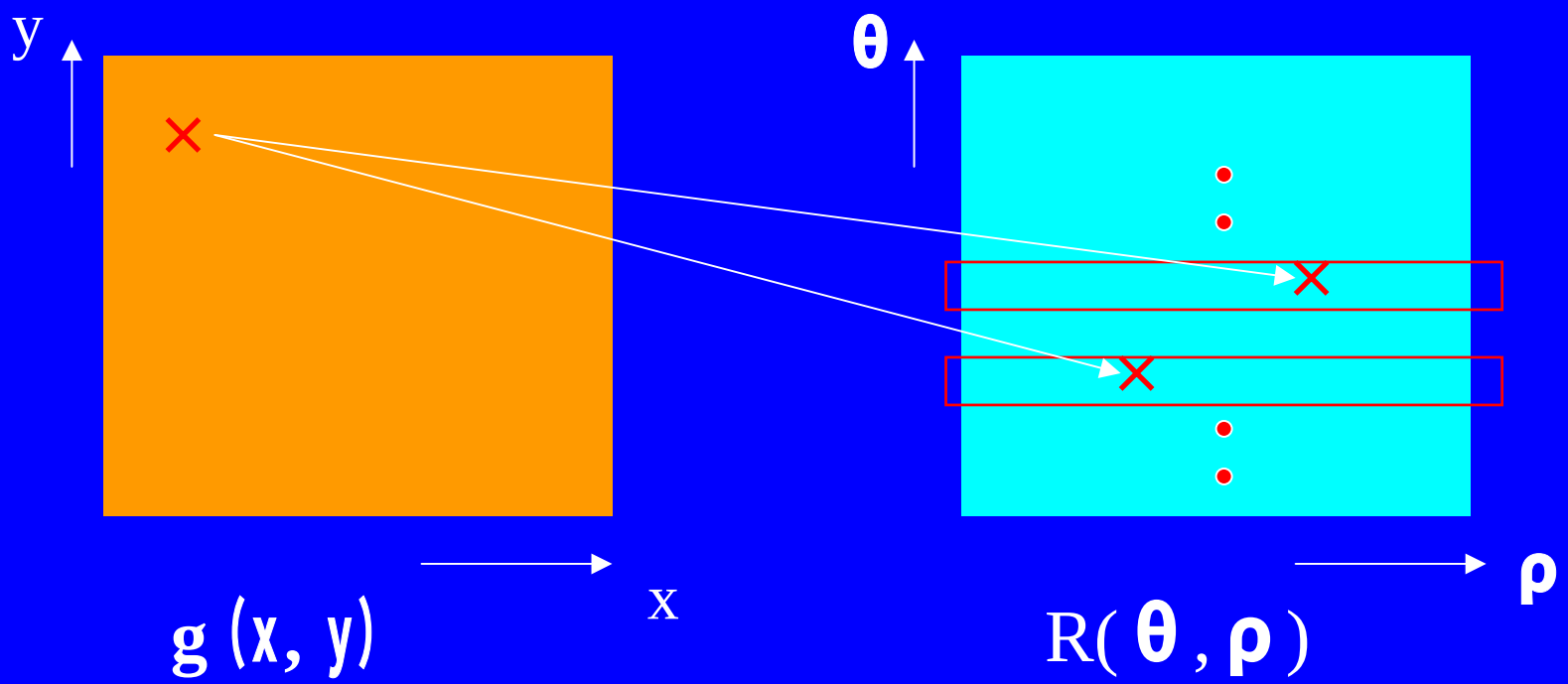
並列アルゴリズムの構築

画像メモリに順次アクセスしながら
変換を実現



順次アクセス

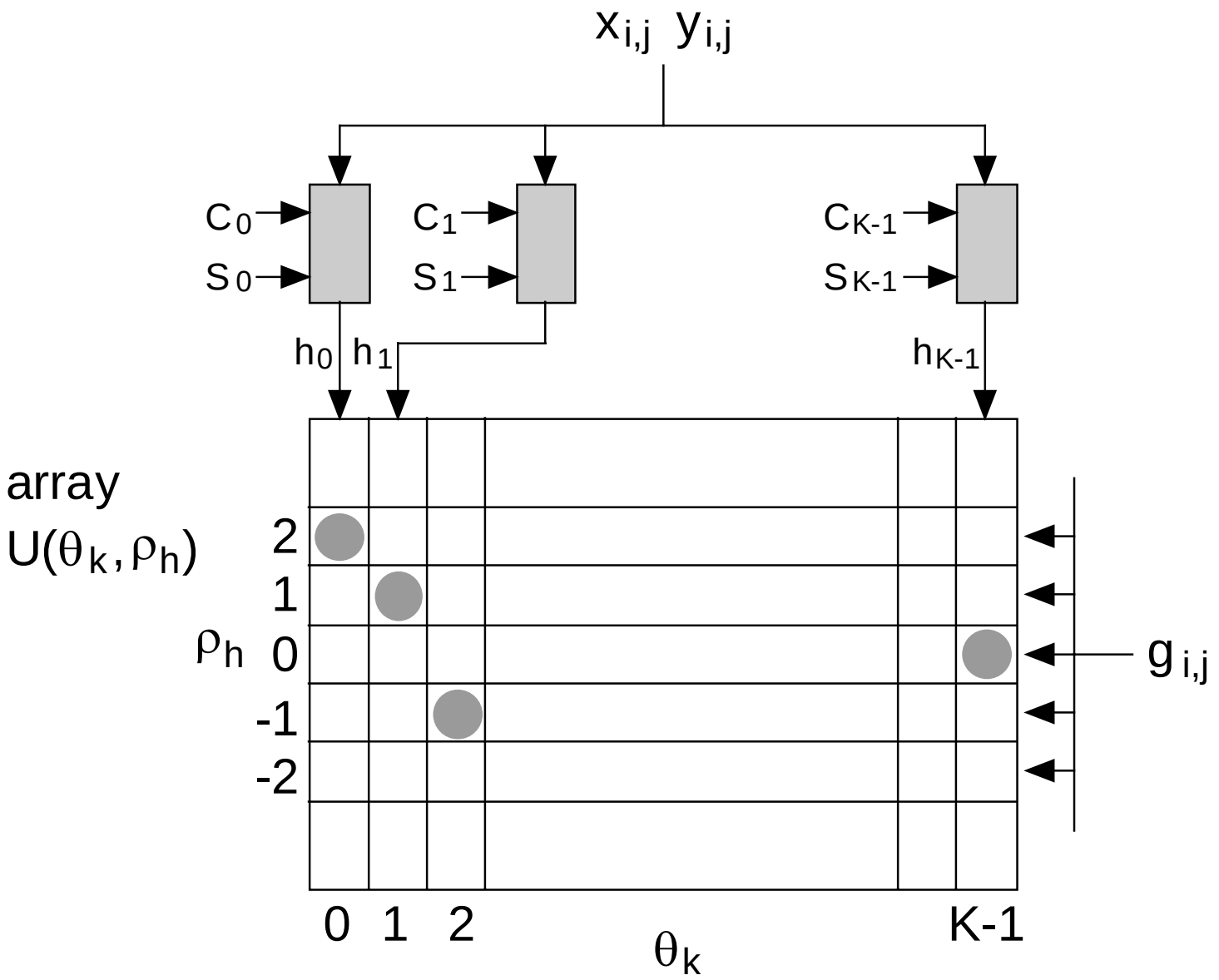
並列アルゴリズムの構築



投票による片側ラドン変換

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

並列片側ラドン変換

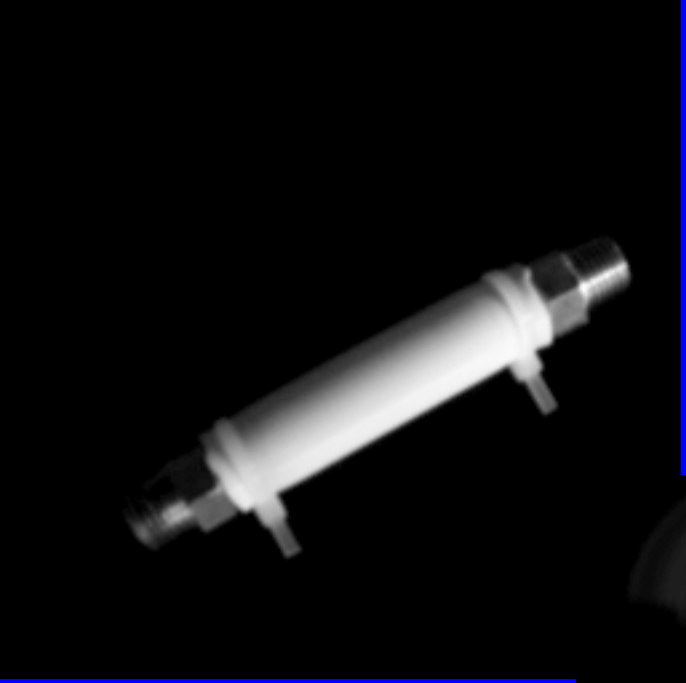

$x_{i,j}$ $y_{i,j}$
C_0
S_0
C_1
S_1
C_{K-1}
S_{K-1}
h_0 h_1
h_{K-1}
array
$U(\theta_k, \rho_h)$
ρ_h
2
1
0
-1
-2
$g_{i,j}$
0 1 2
θ_k
K-1

変換例

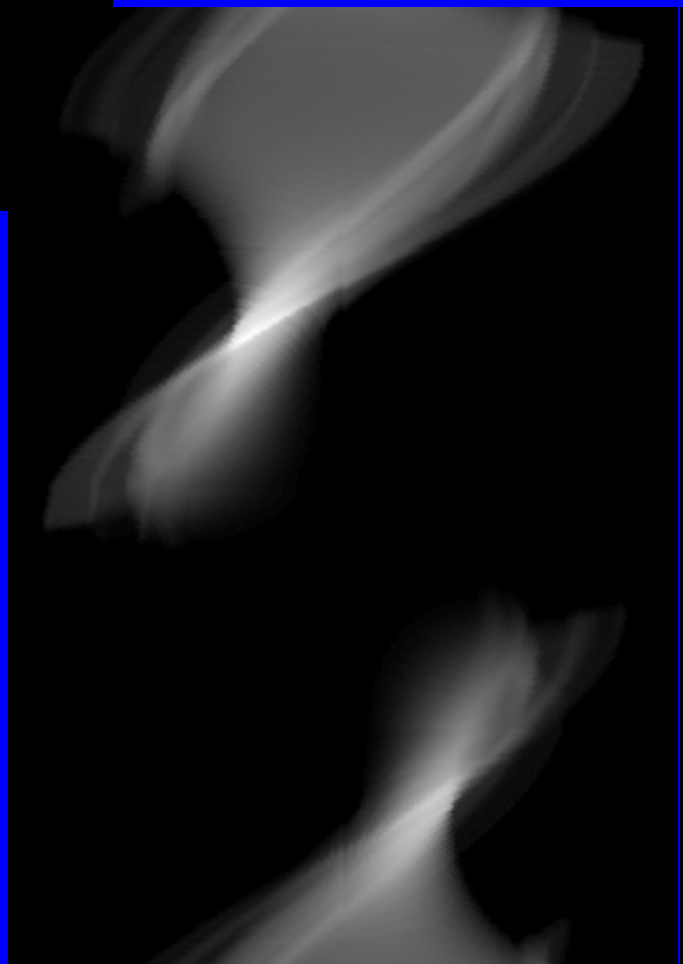
オリジナル

並列

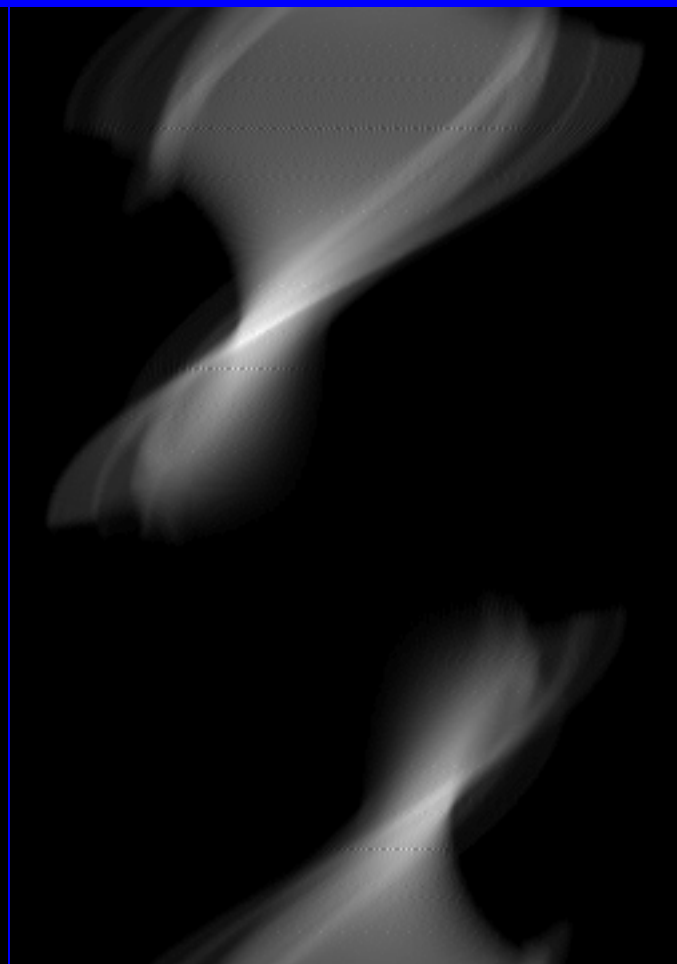
変換例



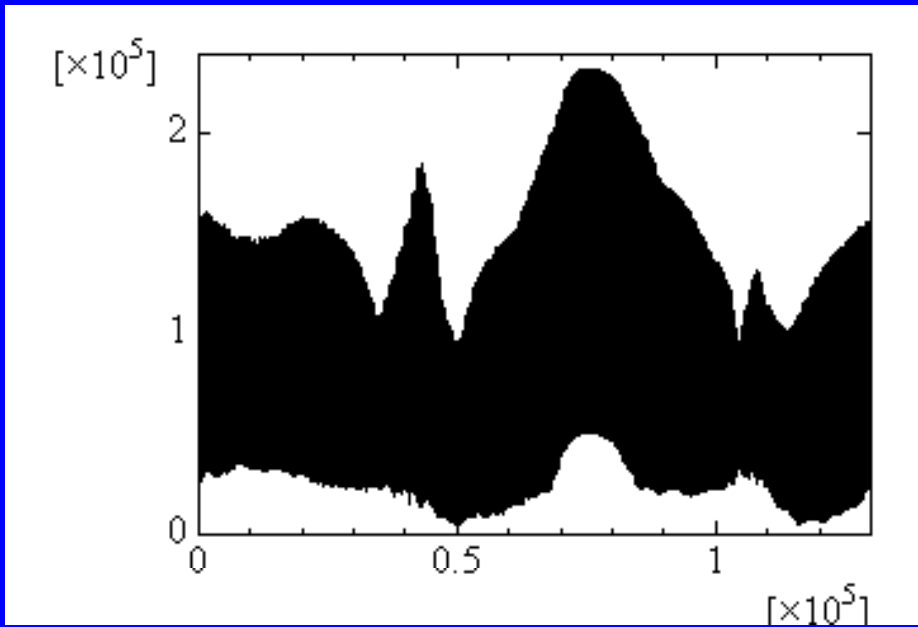
オリジナル



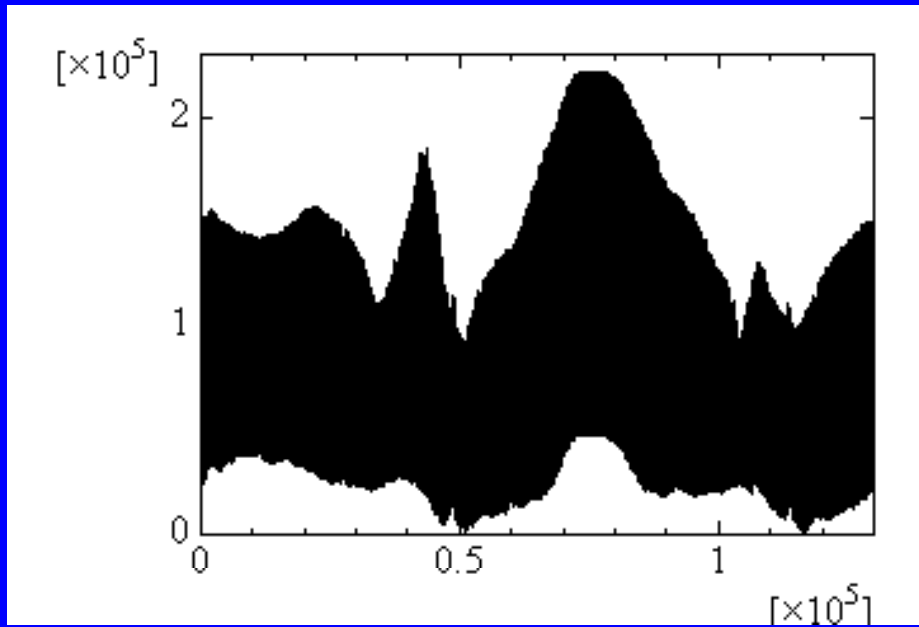
並列



パワースペクトルの比較



オリジナルアルゴリズム

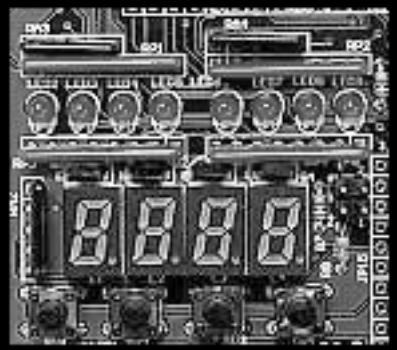


並列アルゴリズム

$$\| \mathcal{F}_{sample}(f_h, \theta_{sample}) - \mathcal{F}_{input}(f_h, \theta_{input}) \|$$

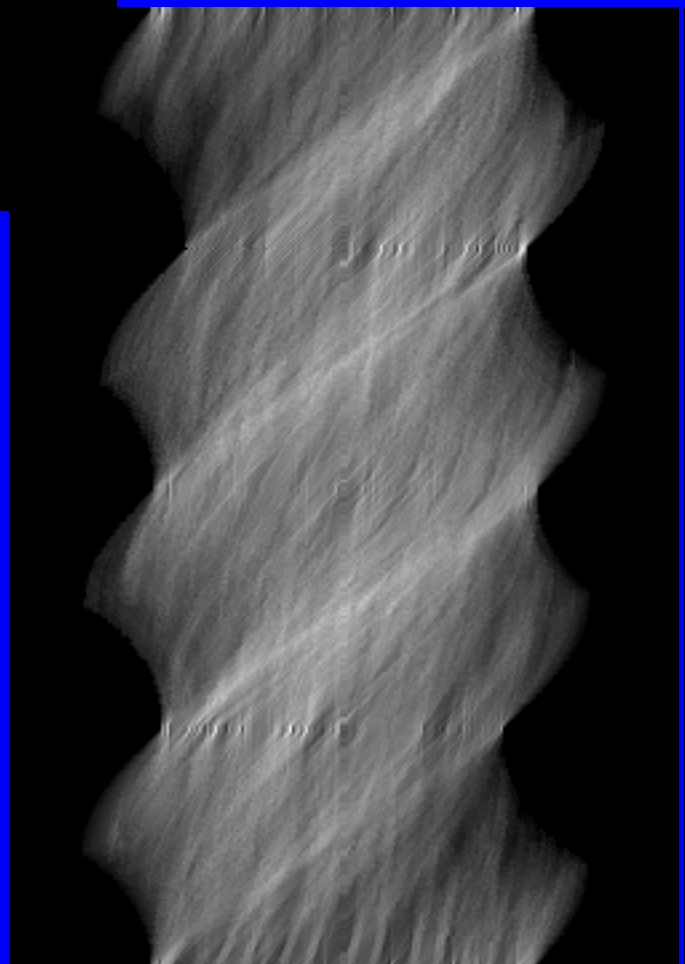
$$= \sum_{h=0}^{40} |\mathcal{F}_{sample}(f_h, \theta_{sample}) - \mathcal{F}_{input}(f_h, \theta_{sample})|.$$

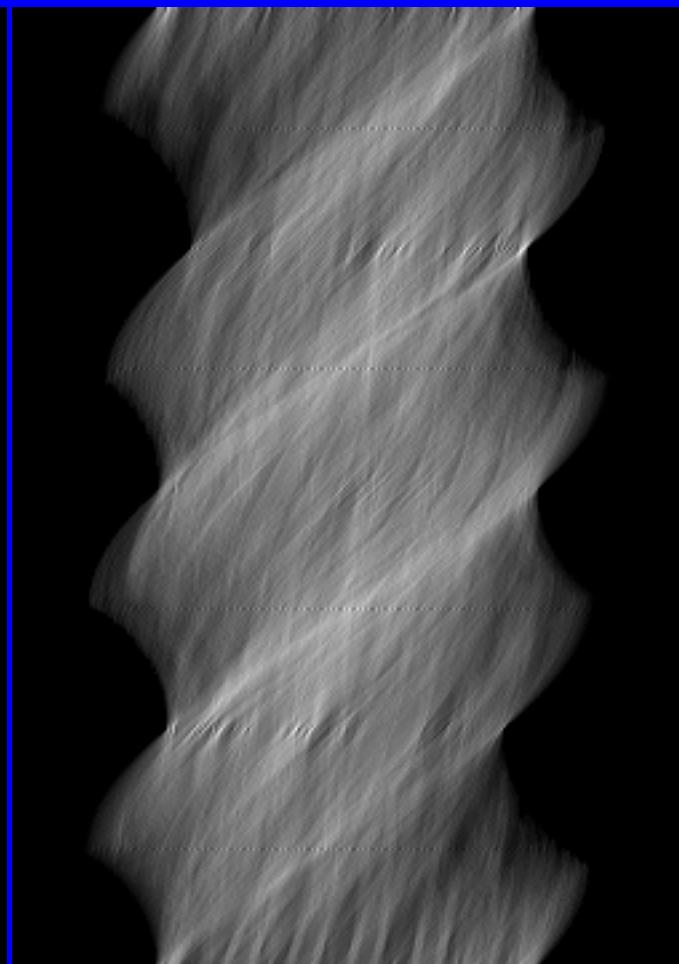
変換例



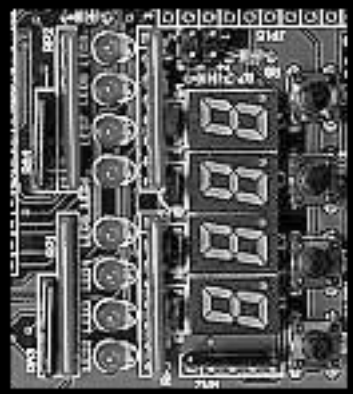
オリジナル

並列



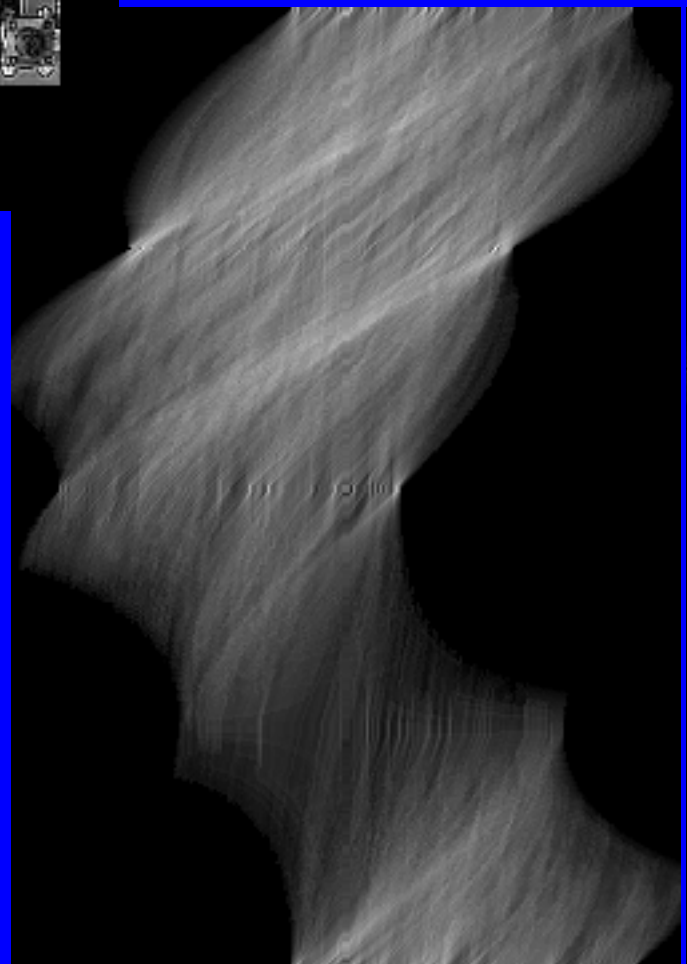


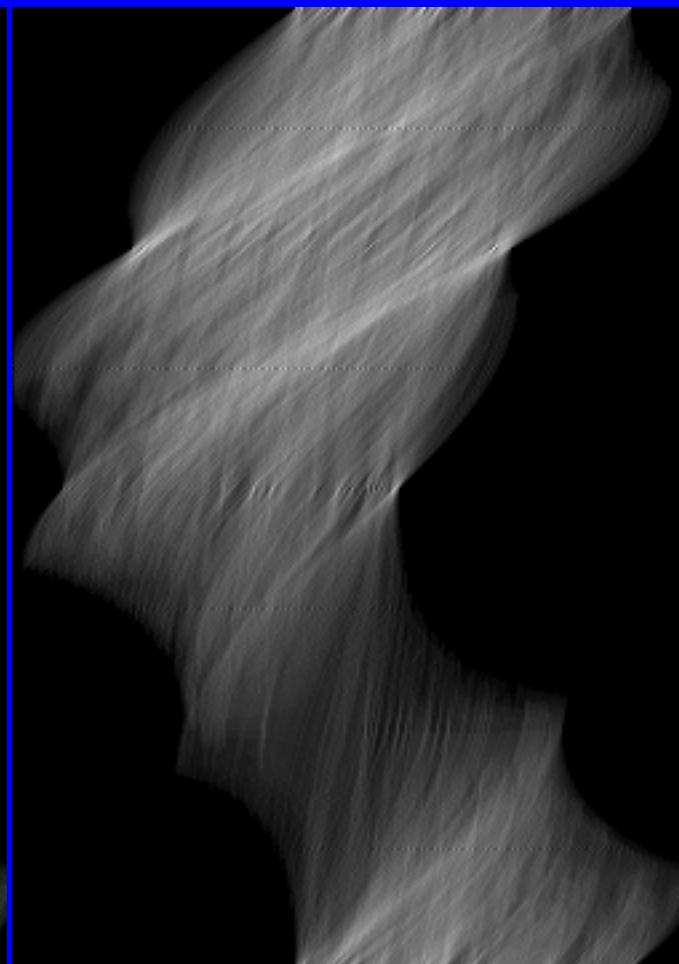
変換例



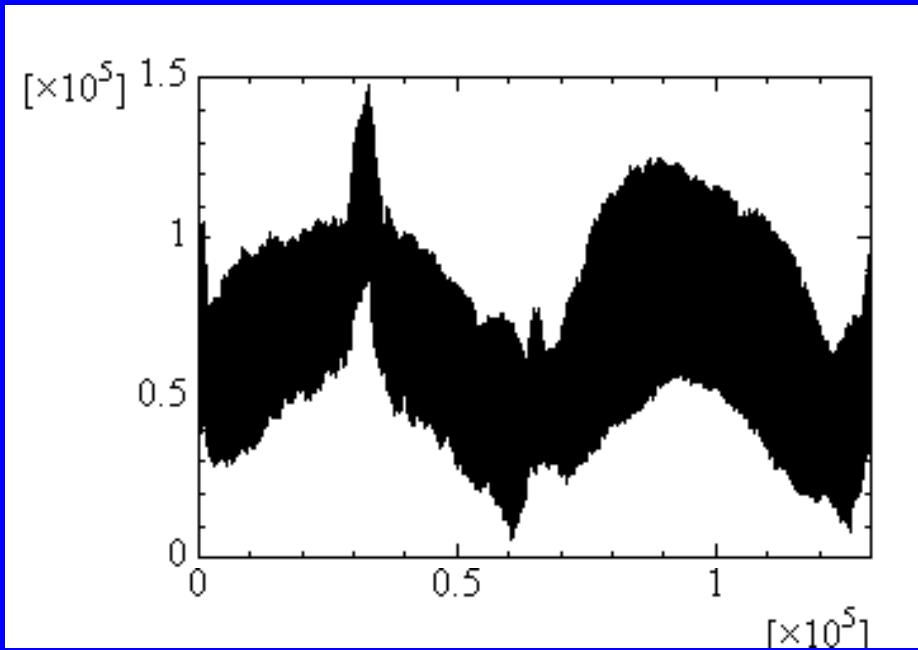
オリジナル

並列

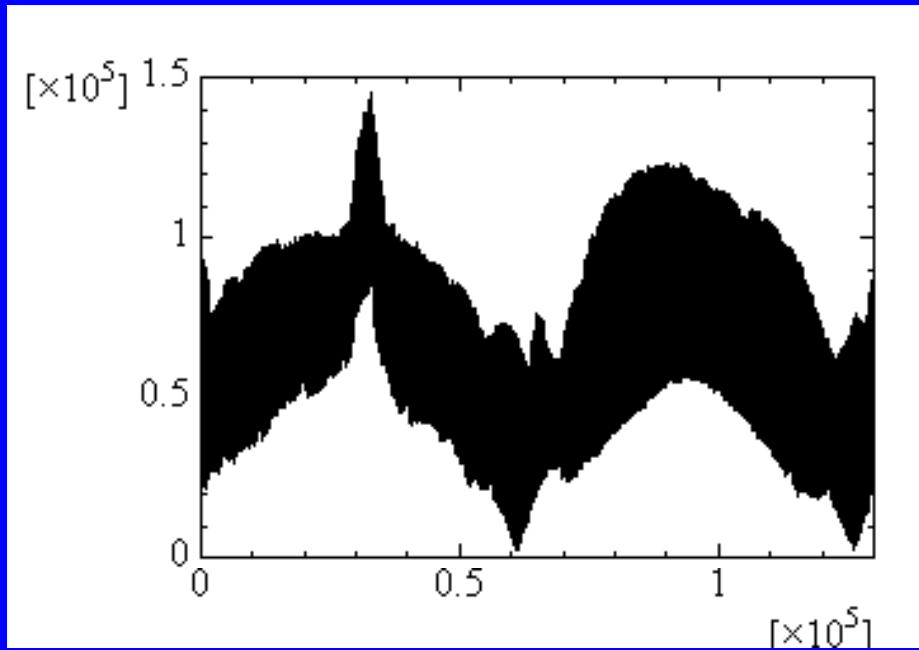




パワースペクトルの比較

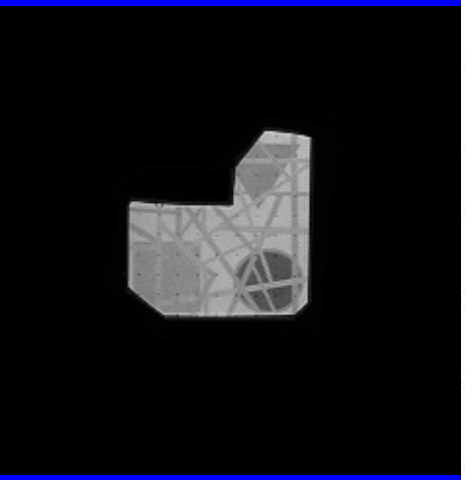


オリジナルアルゴリズム



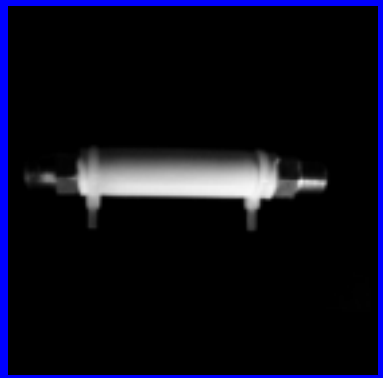
並列アルゴリズム

姿勢の計測

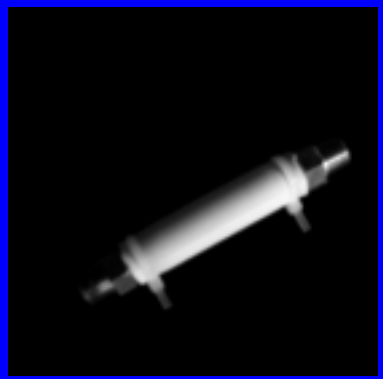


	original		parallel	
actual	computed	error	computed	error
0°	−1°	−1°	−2°	−2°
30°	30°	0°	30°	0°
60°	64°	4°	64°	4°
90°	93°	3°	93°	3°
120°	122°	2°	119°	−1°
150°	148°	−2°	149°	−1°
180°	181°	1°	180°	0°
210°	219°	9°	218°	8°
240°	238°	−2°	245°	5°
270°	266°	−4°	277°	7°
300°	294°	−6°	299°	−1°
330°	324°	−6°	326°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

位置･姿勢の計測結果

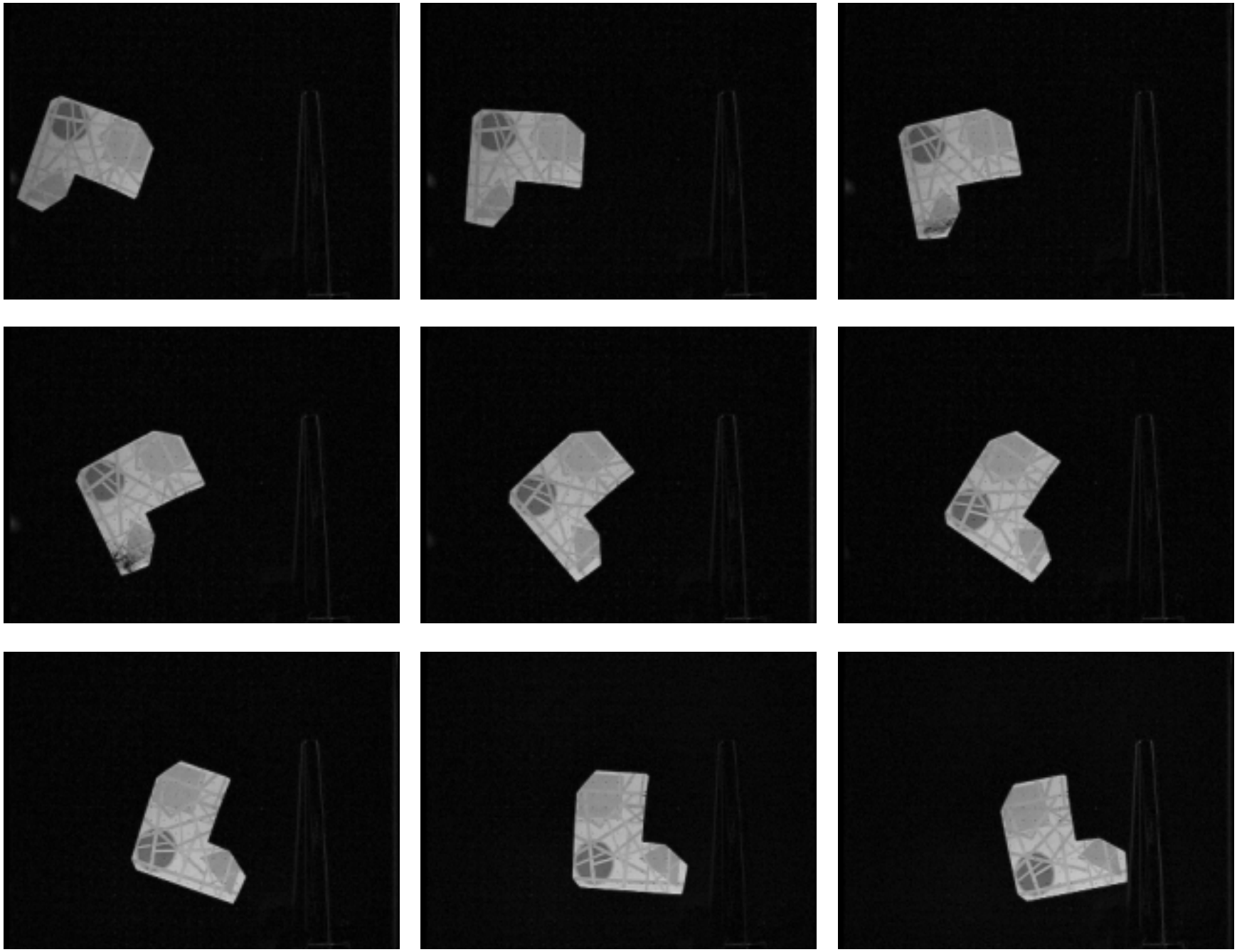


参照画像

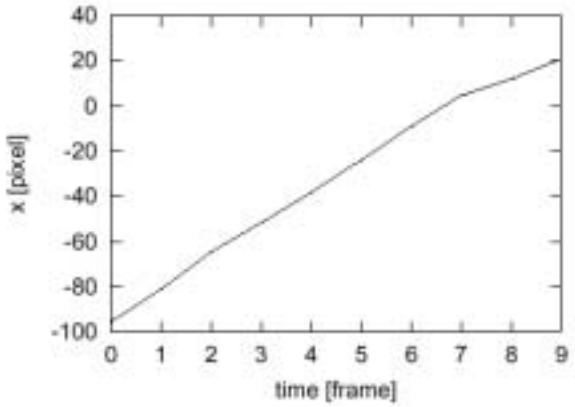


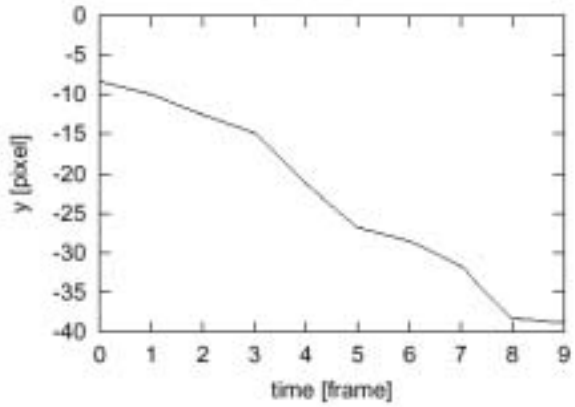
入力画像

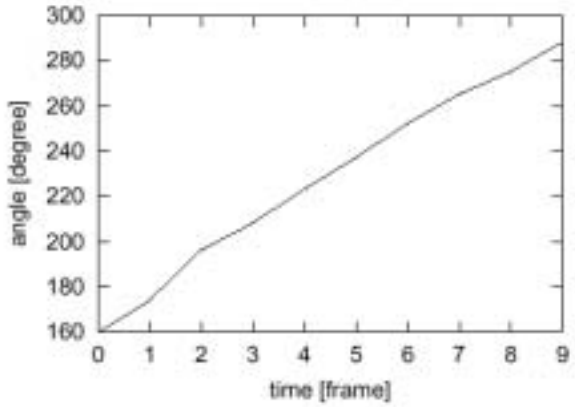
	オリジナル アルゴリズム	並列 アルゴリズム
回転	30°	30°
移動方向	322°	323°
移動距離	40	39



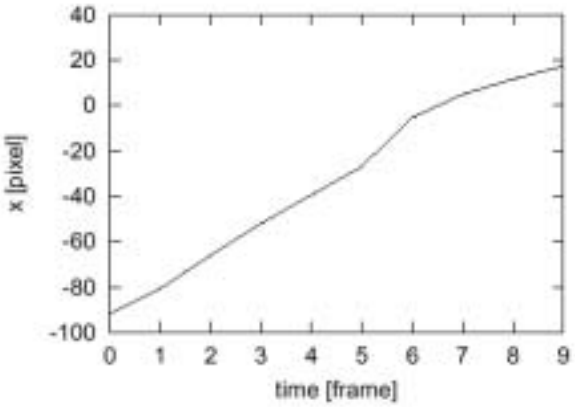
運動計測結果

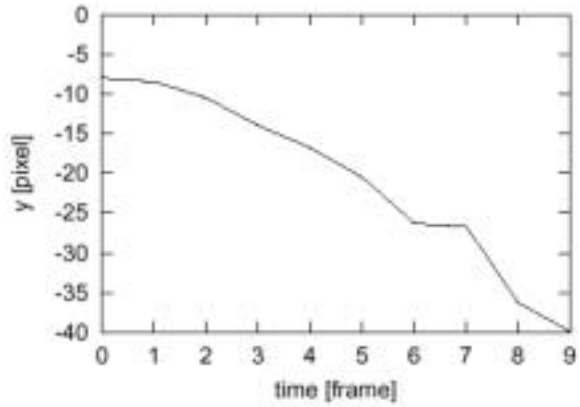


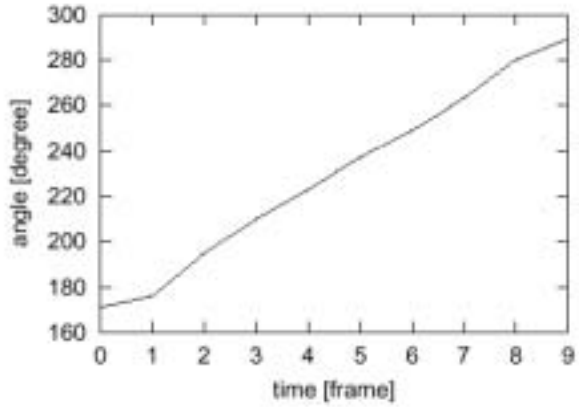




オリジナルアルゴリズム







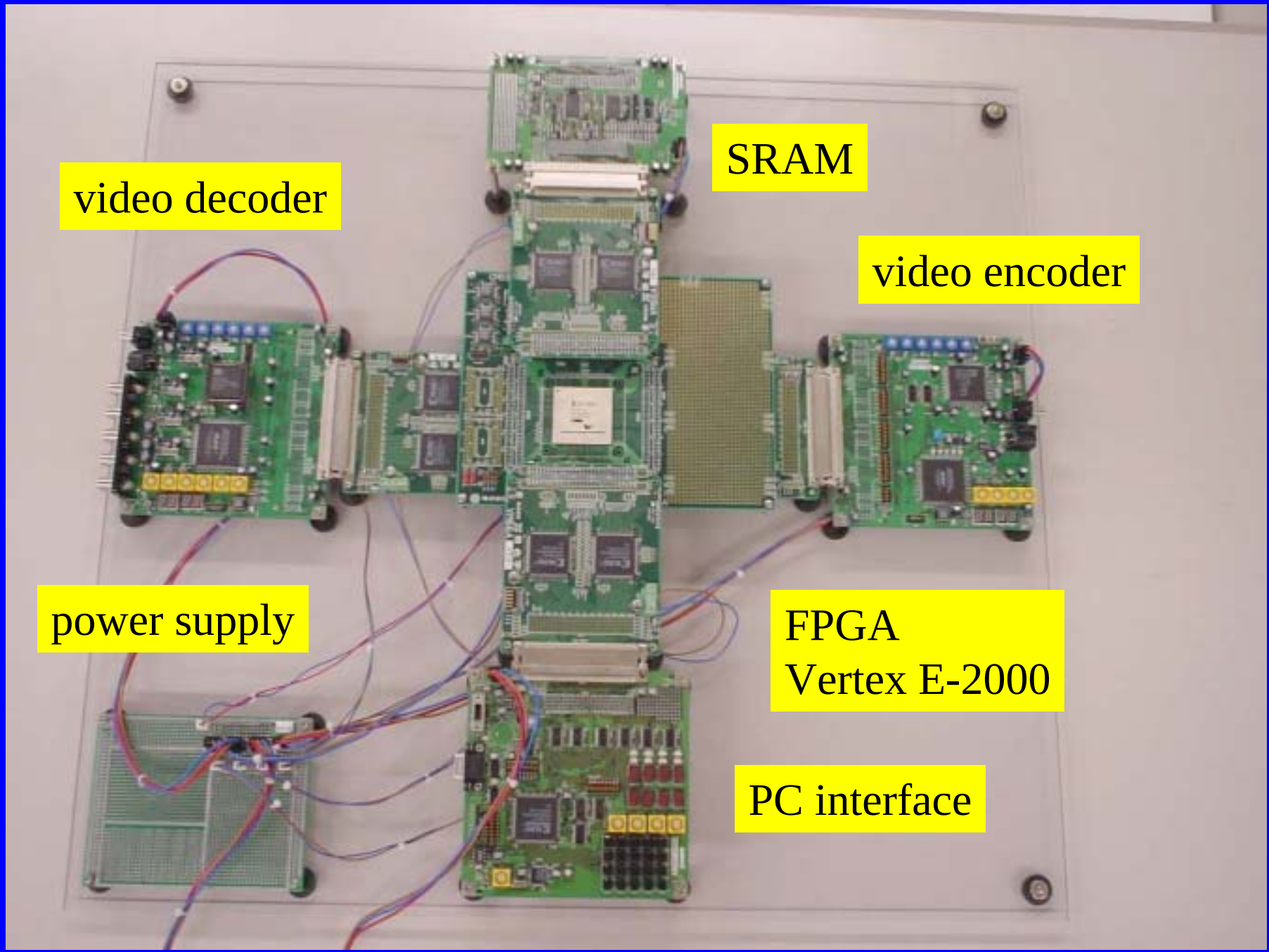
並列アルゴリズム

方針

- センサと信号処理部の分離
  - CCDカメラ
  - NTSC方式
    - 論理演算に焦点
      - FPGA + C レベル設計
- 可能な限り汎用品を使用
  - ビデオエンコーダ／デコーダ

ＦＰＧＡボードの試作

- FPGA Xilinx Vertex-E 2000
- Video encoder/decoder　Bt Series
- SRAM
- JTAG PC interface


video decoder
SRAM
video encoder
power supply
FPGA
Vertex E-2000
PC interface

Cycle C による論理記述

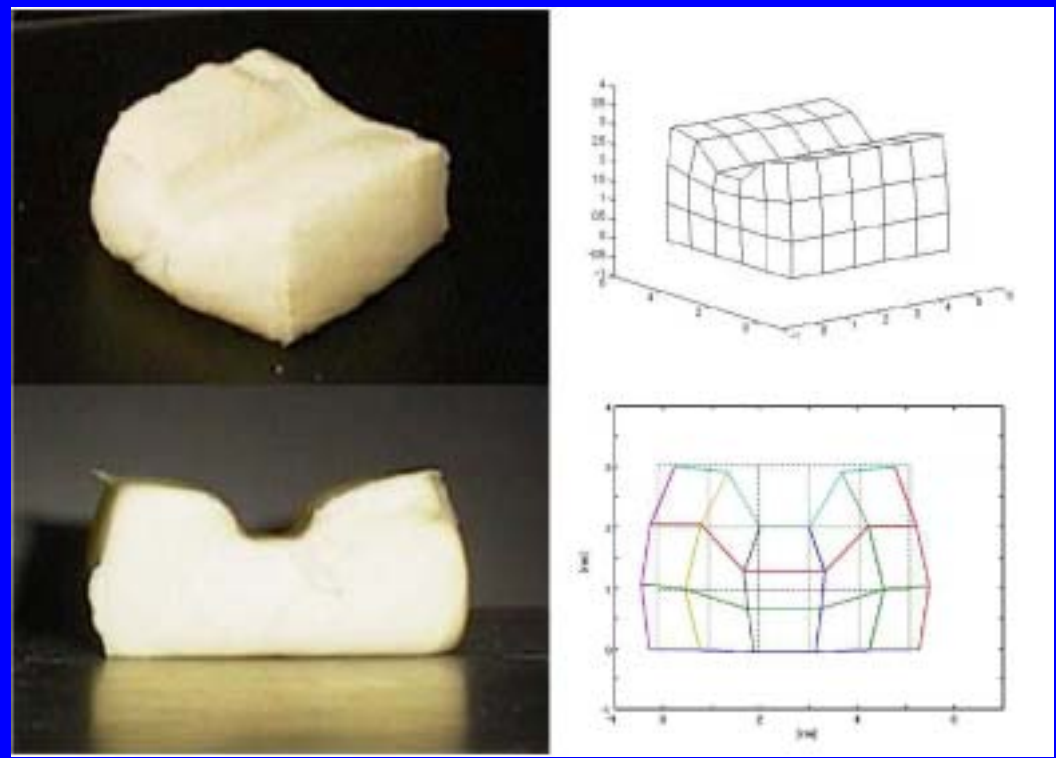
```
class Selecter {
public:
        uint1    reg;
        Selecter ( void ){ reg = 0;}
        void run ( uint1, uint1, uint1, uint1, uint1, uint1& );
};

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;
        }
}
```

進行中

- CycleCによる並列アルゴリズムの記述
- CycleCによるシミュレーション記述
  SRAM, Video Encoder/Decoder
- 試作ＦＰＧＡボードとＰＣ間の通信
  マニピュレータ，ハンドの制御用ＰＣ
- ハンドリングシステムの構築
  RH-5AH55, PA-10

リアルタイム変形シミュレーション

バーチャルリアリティ



まとめ

- ビジョンアルゴリズム
  片側ラドン変換
- アルゴリズムの並列化
  投票
- ＦＰＧＡボードの試作
- ＣｙｃｌｅＣによる論理記述