片側ラドン変換アルゴリズムの FPGA 実装

○増渕 章洋 平井慎一 （立命館大学 理工学部）

FPGA Implementation of One-sided Radon Transform

*Akihiro MASUBUCHI, Shin-ichi HIRAI (Ritsumeikan University)

Abstract : We will implement image processing algorithms based on one-sided Radon transform on FPGA's. Algorithms based on the transform can detect the position and the posture of a planar motion object but require much computation time. Here, we will develop a parallel computation of the transform and will implement the algorithms on FPGA's using C++ - based design.

Key Words : one-sided Radon transform, real-time vision, robot vision

1. はじめに

ビジョンシステムからの情報を元にロボットを制御する場合には、ビジョンシステムにはリアルタイム性が求められる。一般に画像認識処理の計算スピードと性能はトレードオフの関係にあり、リアルタイム性を追求した画像認識処理では、性能が犠牲となっている事が多い。画像処理において計算量が多くなってしまうのは、大量にある各画素に対して、計算を繰り返し行う事が主な要因である事が多い。一方、一つ一つの計算自体は簡単であり、かつ並列に演算する事が可能なケースが多い。この特徴を有した画像認識アルゴリズムは、CPU とソフトウェアの組合せよりも並列処理に強いハードウェアとして実装する事により、性能はそのままに計算スピードが向上する事が期待できる。本研究では、片側ラドン変換を使った画像認識アルゴリズム [1]に工夫を加え、アルゴリズムをハードウェアとして FPGA に実装し、ロバストなリアルタイムビジョンシステムの実現を目指す。

2. 画像処理アルゴリズム

片側ラドン変換を用いた画像認識アルゴリズムは、以下に示す処理となる。

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

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

Step 4

$$\alpha = \theta_{input} - \theta_{sample}$$

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

Step 5 Find d_0 that satisfies

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

Step 1 は、片側ラドン変換の計算である。片側ラドン変換は、2次元画像から1次元射影を求める変換であるラドン変換の積分路を、画像認識に向くように改良した変換である。上記の画像認識アルゴリズムでは、片側ラドン変換後の空間上でテンプレート画像と入力画像のマッチングを行い、対象物体の位置と姿勢を認識する。片側ラドン変換 $U(\rho,\theta)$ は、次式で表せる。

$$U(\rho,\theta) = \int_0^\infty g(\rho\cos\theta + \xi\sin\theta, \rho\sin\theta - \xi\cos\theta)\, d\xi$$

ここで、$g(x,y)$ は片側ラドン変換を適用する対象画像をあらわす。パラメータ ρ , θ は、それぞれ原点からの距離と x 軸からの角度を表す。このアルゴリズムをそのまま適用する場合は、まず対象画像 $g(x,y)$ が全画素とも揃っている必要がある。これは、入力された映像をフリーズするフレームメモリが必要であり、なおかつフリーズが完了するまで変換の演算を始められないという事を意味している。また、フレームメモリを設けたとしても、フレームメモリへのアクセスは順次アクセスではなく、ランダムアクセスとなり、メモリの性能を生かしきれない可能性がある。本研究で最終的に FPGA に実装するアルゴリズムでは、この点を改良し、ρ , θ を変化させるのではなく、入力された画素の x , y から ρ , θ を算出し、$U(\rho,\theta)$ を構築するようにしている。この工夫により理論的には、フレームメモリを必要とせず画素の入力と同時に片側ラドン変換を行う事が出来るようになった [2]。

Step 2 の処理は、Step 1 の処理の完了を待ってから開始される。Step2 では、片側ラドン変換の結果に対してフーリエ変換を行い、パワースペクトルを求める。フーリエ変換も複雑な演算ではあるがメジャーな演算でもあるため、フーリエ変換に関しては専用のライブラリが IP コアという形で FPGA デバイスのメーカーから提供されている。フーリエ変換の FPGA 実装は本研究の焦点ではないので、回路規模等の問題が出ない限りは、本研究ではこれを利用して実装を進める。また、繰り返しの演算回数を Step 1 と比較してみても、PC 上での実験によっても、このアルゴリズムで最も演算に時間がかかっているのは Step 1 である事が判っている。PC 上でのソフトウェアによる実験では、順次処理での片側ラドン変換にかかった処理時間は、FFT を使用したフーリエ変換処理の約 175 倍となった。

Step 3 では、テンプレート画像と入力画像のパワースペクトルを比較し、最も相関の高い角度を求める。Step 3 で行う演算量は、Step 2 と比較しても少ない。

Step 4 では、Step 3 の結果を元に、回転角度 α と並進移動方向 β を求める。繰り返しの演算処理が無いため、演算量は極めて少ない。

Step 5 は、Step 1 の片側ラドン変換の結果と、Step 3 の結果から、並進移動距離 d_0 を算出する。ここでは繰り返し演算が行われるが、Step 1,2 と比べると演算量は少ない。

3. 試作 FPGA ボード

前述の片側ラドン変換を使用した画像認識アルゴリズムを FPGA に実装する事を目的として FPGA ボードを試作した。その外見を Fig.1 に示す。

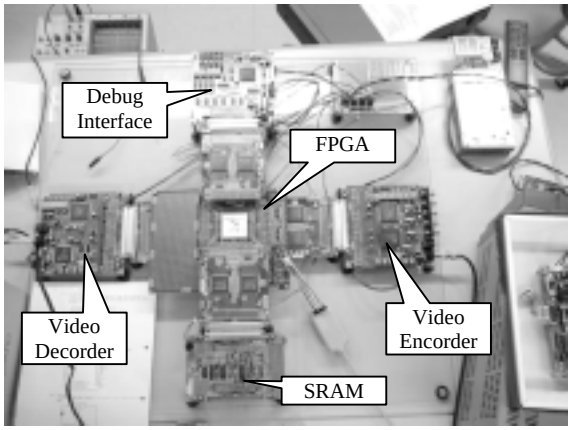

Fig. 1 : Prototype FPGA board

この試作 FPGA ボードは、写真化学製の CM59 (Xilinx 製 VertexE2000 搭載) 大規模 FPGA ボードを中心に、三菱電機マイコン機器ソフトウェア製の MU200-VDEC (ビデオデコーダ)、MU200-VENC (ビデオエンコーダ)、MU200-SRAM,MU200-EX40 (FPGA 学習用ボード) を接続した物である。これらのうち、画像認識処理に直接関係するのは、中心の大規模 FPGA とビデオデコーダと SRAM のみであり、他の部品は開発支援用に接続している物である。

このビジョンシステムは、映像入力側は NTSC または PAL のビデオ信号である。認識結果は、PC/104 ベースの小型コンピュータへと直接送信され、そこから産業用リアルタイム LAN の ARCNET または Ethernet を使ってロボットシステム等のホストシステムに送信される。想定しているロボットシステムとの接続例は、以下のようなシステムである。

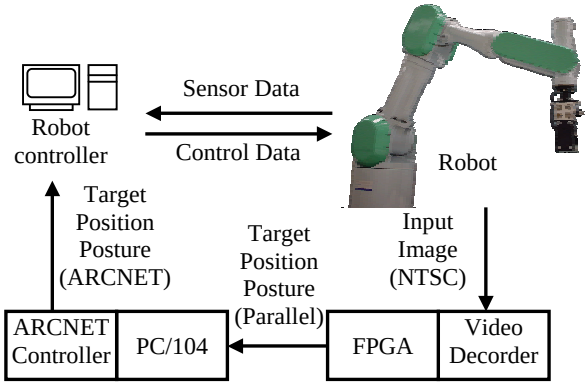

Fig. 2 : Target Robot System

4. FPGA 内部回路設計

片側ラドン変換と一連のマッチング処理は複雑な回路になるため、まずは以前にソフトウェアで PC 上に実装した、片側ラドン変換を限定的に使用する簡易実装[3]を FPGA でも行う事とする。この簡易実装は、最初に対象の重心を求め、その重心を原点として片側ラドン変換を行う。また、片側ラドン変換も ρ=0 の場合のみを計算し、マッチングを行う。片側ラドン変換はオリジナルではなく工夫された順次処理を使う。ただし、重心を求めるために、SRAM にフレームメモリを設けて映像をフリーズする事とする。以上の構成を Fig. 3 に示す。

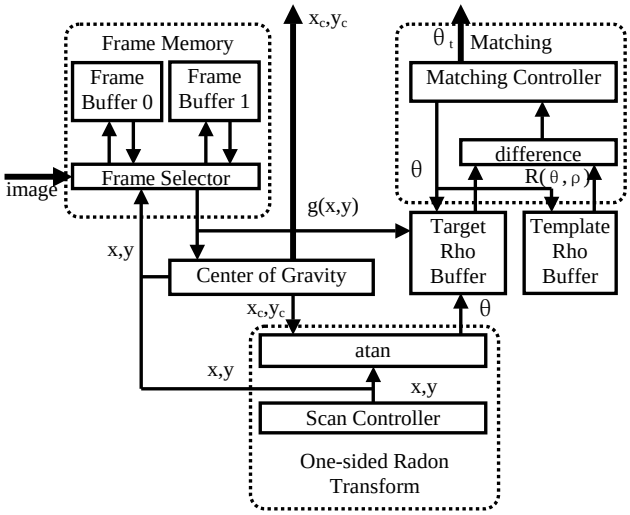

Fig.3 : One-sided Radon Transform LSI

Fig.3 の中で、点線で囲ってある部分は主要な回路を表している。主要な回路では、片側ラドン変換部分とマッチング部分は実装しており、画像入力とフレームメモリ関連の回路をこれから製作する。これらの回路は、開発を容易にするために、全回路が入力画像のクロックに同期して動作するよう設計する。

回路の設計には、通常使用される HDL は使わず、C++ 言語ベースの C Level Design 社の System Compiler を用いる。C++ベースであるため、シミュレーションを通常の C++ プログラムと同様の手法で行える。これは、画像処理のように演算量が多くても高速にシミュレーションが可能で、テストパターンの生成にも既存の資産を生かせるという事を示している。

5. おわりに

片側ラドン変換アルゴリズムの FPGA 化のために、試作 FPGA ボードを試作した。今後、この試作ボードにまず簡易版の片側ラドン変換を使用した画像処理アルゴリズムを実装し、次に完全版のアルゴリズムを実装し、ロボットシステムと接続し実証実験を行う。

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) 坪井, 平井:"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",電子情報通信学会 情報・システムソサエティ大会 講演論文集,pp.225, 2001

2) 座光寺,平井:"リアルタイムビジョンのための片側ラドン変換法の並列化",第 19 回ロボット学会学術講演会予稿集,pp.301-302, 2001

3) 増渕,平井:"ロボット搭載用高速高機能ビジョンモジュールの研究",第 18 回ロボット学会学術講演会予稿集,pp.649-650, 2000