市街地地図認識におけるパイプライン処理*

3C－7

押谷 徹　渡邉 豊英†

名古屋大学大学院 工学研究科 情報工学専攻‡

1 はじめに

画像処理、パターン認識において、処理の効率化の手法としてパイプライン処理がある。パイプライン処理では、処理をいくつかのステップへと分割し、各ステップを異なるプロセッサが行う。そして、処理データがプロセッサの間を処理手順に従って流れることによって、複数のプロセッサが並列に動作可能となる。すなわち、画像をいくつかの部分画像へ分割し、各部分画像をパイプラインへ流すことによって、処理の高速化を実現できる。また、1台のプロセッサで処理を行うとき、処理の効率化のために、従来の処理アルゴリズムを一部変更したパイプライン処理の手法がある[1]。これは、画像全体を行単位で分割し、行ごとにメモリへ送り、プロセッサが処理し、その結果を行ごとに出力する手法である。従って、この手法では記憶領域の効率化を実現することができる。

前者の方法では、各部分画像が独立に処理されるので、部分画像の分割境界での後処理が必要となる。また、後者の方法では、局所的処理に必要な行のメモリを確保すれば、分割境界を意識する必要はない。しかし、大局的な処理においては不向きである。本稿では、分割されたデータ間での協調処理のためのパイプライン処理の手法について述べる。

パイプライン処理の対象として、分割型多層黒板モデルを用いた市街地地図認識の並列処理[2, 3]を取り上げる。分割型多層黒板モデルでは、市街地地図認識の各処理データを保持する黒板を多層構造とし、各層は部分域へと分割される。そして、処理可能となった部分域をタスクとしてプロセッサへ割り当てることによって、データ並列と機能並列の複合的な実行を実現している。この分割型多層黒板モデルの下で、パイプライン処理の手法について述べる。

2 枠組み

黒板の各層を部分域へ分割することによって、データ並列を実現できる。しかし、各部分域が独立に処理されることによって、部分域間の分割境界や大域的処理が問題となる。これらの問題に対して、部分域間での協調処理が重要となる。よって、部分域間で協調処理が必要となった場合、パイプライン処理を行う。

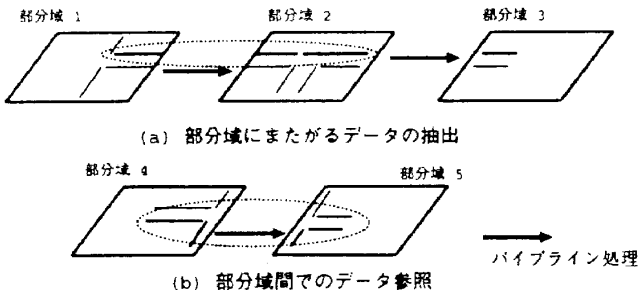

図 1: パイプライン処理

パイプライン処理の概要を図1に示す。パイプライン処理が必要となる状況は、以下の通りである。

1. 線分抽出などの複数の部分域にまたがるデータの抽出（図1 (a)）
2. 平行線分対同定、交差点同定などの道路要素同定処理において、部分域間でのデータ参照（図1 (b)）

以上のように、パイプライン処理が必要となった場合、隣接する部分域に対して要求することによって、パイプライン処理を実現する。例えば、1.では、部分域1の処理過程で部分域の端へ到達したとき、隣接する部分域2に対して処理データを渡し、抽出処理を継続するよう要求する。また、2.では、部分域4の処理において、隣接する部分域5に関連するデータが存在する可能性がある場合、処理データを渡し、同定処理を継続するよう要求する。

要求を受けた部分域では、受け取った処理データと処理内容に基づいて、該当する部分を中心にデータの抽出や同定処理を行う。また必要に応じて、隣接する部分域に対してさらにパイプライン処理を要求する。

*Pipeline Processing for Urban Map Recognition
†Toru OSHITANI and Toyohide WATANABE
‡Department of Information Engineering, Graduate School of Engineering, Nagoya University

3 実現方法

パイプライン処理は、隣接する部分域間での処理の要求によって行われる。各部分域はタスクとしてプロセッサで処理される。よって、パイプライン処理を隣接する部分域へ要求するために、リクエストを用いる。プロセッサは、パイプライン処理が必要となった場合、リクエストを生成し、リクエストの実行制御を行うリクエスト・マネージャへ送信する。

3.1 リクエスト

リクエストはパイプライン処理で実行される処理内容を保持しており、リクエストの送信先、処理データ、処理手続きからなる。

- リクエストの送信先
  パイプライン処理が要求される部分域を表す。
- 処理データ
  リクエストが生成される前までにプロセッサの処理によって得られた中間データ。リクエストの実行時には、この処理データと関連する部分を中心にデータの抽出や同定処理を行う。
- 処理手続き
  プロセッサが実行する処理手続きを表す。

このリクエストがプロセッサでの処理単位となる。

3.2 リクエスト・マネージャ

リクエスト・マネージャは、プロセッサによって生成されたリクエストの実行制御を行う。リクエスト・マネージャは、プロセッサが実行している処理手続きや部分域の情報を保持しており、この情報に基づいてリクエストをプロセッサへ割り当てる。すなわち、リクエストの送信先である部分域と処理手続きを持つタスクを実行しているプロセッサが存在するかどうかを調べる。プロセッサが存在する場合、そのプロセッサにリクエストを割り当てる。リクエストを受け取ったプロセッサは、実行中のタスクが終了した後、リクエストを実行する。一方、プロセッサが存在しない場合、リクエストを実行するタスクを生成し、スケジューリング・キューへ登録する。このタスクはスケジューラによってプロセッサへ割り当てられ、実行される。このスケジューリングにおいて、リクエストのタスクは通常のタスクより優先的に選択される。

この様子を図2に示す。部分域1と処理手続きaからなるタスクを処理するプロセッサAが、部分域2に対してパイプライン処理が必要となった場合、リクエストを生成し、リクエスト・マネージャへ送信する。リクエスト・マネージャは、リクエストの送信先である部分域2と処理手続きaを持つタスクを実行しているプロセッサが存在するかどうかを調べる。プロセッサBがそのタスクを実行している場合、プロセッサBへリクエストを割り当てる。一方、プロセッサが存在しない場合はタスクを生成する。

リクエストを送信したプロセッサは引き続きタスクを実行する。黒板への書込みなどリクエストの実行後の処理は、リクエストを処理するプロセッサが行う。

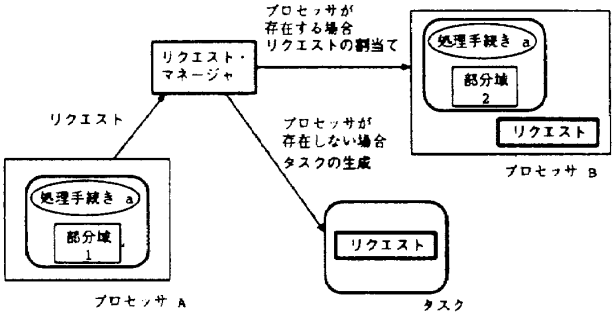

図 2: リクエスト・マネージャ

4 おわりに

本稿では、分割型多層黒板モデルの下で分割されたデータ間での協調処理のために、リクエストを用いたパイプライン処理について述べた。今後の課題として、ボトムアップ処理、同一層での部分域間の協調処理に認識率の向上を目的とした道路ネットワークの修正や推定を行うトップダウン処理や層間での協調処理を組み合わせることが挙げられる。

参考文献

[1] 中山晶, 木村文隆, 吉田雄二, 福村晃夫: “大規模画像に対する局所並列型アルゴリズムのパイプライン方式 による効率化”, 電子情報通信学会論文誌 D, Vol. J69-D, No. 2, pp. 259–260 (1986).

[2] 押谷徹, 渡邉豊英: “市街地地図認識の並列処理”, 電子情報通信学会技術研究報告, Vol. PRMU 98, No. 70, pp. 93–100 (1998).

[3] 押谷徹, 渡邉豊英: “分割型多層黒板モデルを用いた市街地地図認識の並列処理”, 並列処理シンポジウム JSPP'98, pp. 271–278 (1998).