データベースシステム 99−11
（1994. 7. 20）

分散処理環境における汎用画像オブジェクトの構成

脇山 俊一郎 † 川島 享 †† 金森 吉成 †† 増永 良文 †††

† 仙台電波高専 †† 群馬大学 ††† 図書館情報大学

本稿では、画像オブジェクトをアプリケーションには依存させず、汎用的でかつ拡張可能なものにするために、画像処理を専門に行なう画像処理サーバを設け、それを画像データベースのクライアントとするアーキテクチャを提案する。画像処理サーバには、一般的な画像処理手続きをオブジェクト化した画像処理オブジェクトがある。アプリケーションでは、このサーバから画像処理オブジェクトを検索し、画像オブジェクトのメソッドとして仮想的に振舞うようにしている。画像処理サーバでは、画像処理機能に加え、試行錯誤的な画像処理の適用を支援する処理履歴管理機能や、各画像処理オブジェクトの処理機能を例示するヘルプ機能も提供する。

An Organization of Generic Image Objects in Distributed Processing Environment

Shunichiro Wakiyama † Susumu Kawashima ††
Yoshinari Kanamori †† Yoshifumi Masunaga †††

† Sendai National College of Technology
†† Gunma University
††† University of Library and Information Science

In this paper, we describe an organization of image objects, which are generic and extensible, and are indepent on any applications. We propose an architecture which has image processing server as clients in image database systems. There are image processing objects in the server. These objects are made by organizing general image processing procedures. When applications use image processing objects retrieved from the server, these objects virtually seem to be methods of image objects. There are two other functions: version management which supports how to appy image processing objects by trial and error, and help to illustrate examples before and after processing images.

1 はじめに

画像データをオブジェクト指向データベースシステムで管理する場合、画像オブジェクトの実装法によっては構築されるデータベースが特定用途向けになってしまい、データベースが本来持つべき共有資源としての汎用性を失ってしまうという問題がある [1]。これは画像オブジェクトに埋め込まれたメソッドが、特定の動作環境やアプリケーションに依存したものであるために生じる問題である。

画像オブジェクトの汎用性を保持しつつ、アプリケーション依存の振舞いを与えるためには、アプリケーション依存の処理を画像オブジェクトのメソッドから外しておき、それを必要に応じてその都度追加できるような拡張可能性を画像オブジェクトに持たせる必要がある。

本稿では、画像処理に関する振舞いをすべてデータベース上の画像オブジェクトから取り除き、それらを別途設けた画像処理サーバで分散処理させることで画像オブジェクトの汎用性と拡張可能性を実現する方法を提案する。

2 画像を取り扱うアプリケーションの構造

オブジェクト指向で画像を取り扱うプログラムを作成する場合、プログラムの構造は図 1のようになる。

アプリケーションユーザはユーザインタフェースを介して画像オブジェクトにメッセージを送り、所望の画像処理を施しその結果をユーザインタフェースに反映させる。

画像オブジェクトに対する直接操作 [2] を重んずるばかり、画像オブジェクトにユーザインタフェースの部分も含んでしまう場合があるが、そのような実装では現代の異機種分散環境に対応することはできない。動作環境に依存するユーザインタフェース部は、画像オブジェクトとは切り分けておく必要がある [1]。

アプリケーションレベルの画像オブジェクトには、そのアプリケーションで必要となる画像

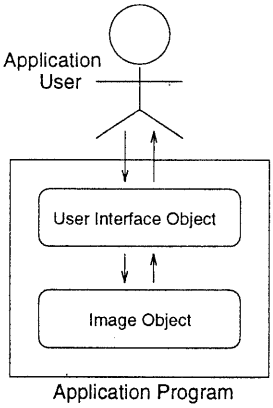

図 1: 画像を取り扱うアプリケーションの構造

オブジェクトの振舞いがメソッドとして格納されていることが望ましい。一方、オブジェクト指向データベースで管理される画像オブジェクトには、特定のアプリケーションに依存しない汎用性が要求されるため、メソッドの格納は避けなければならない。

我々はアプリケーションレベルの画像オブジェクトを仮想的な画像オブジェクトとして捉え、その仮想画像オブジェクトをアプリケーションに提供し、そこで直接メソッドを選択する機構を設けることにした。

3 仮想画像オブジェクトを提供するアーキテクチャ

アプリケーションに特化した仮想画像オブジェクトを提供するためのアーキテクチャを図 2に示す。

システムは

(1) 画像オブジェクトと画像処理ライブラリ (メソッド群) を管理するオブジェクト指向データベース (サーバ)

(2) (1) のオブジェクト指向データベースのクライアントとして
- 画像オブジェクトの検索処理を行なう

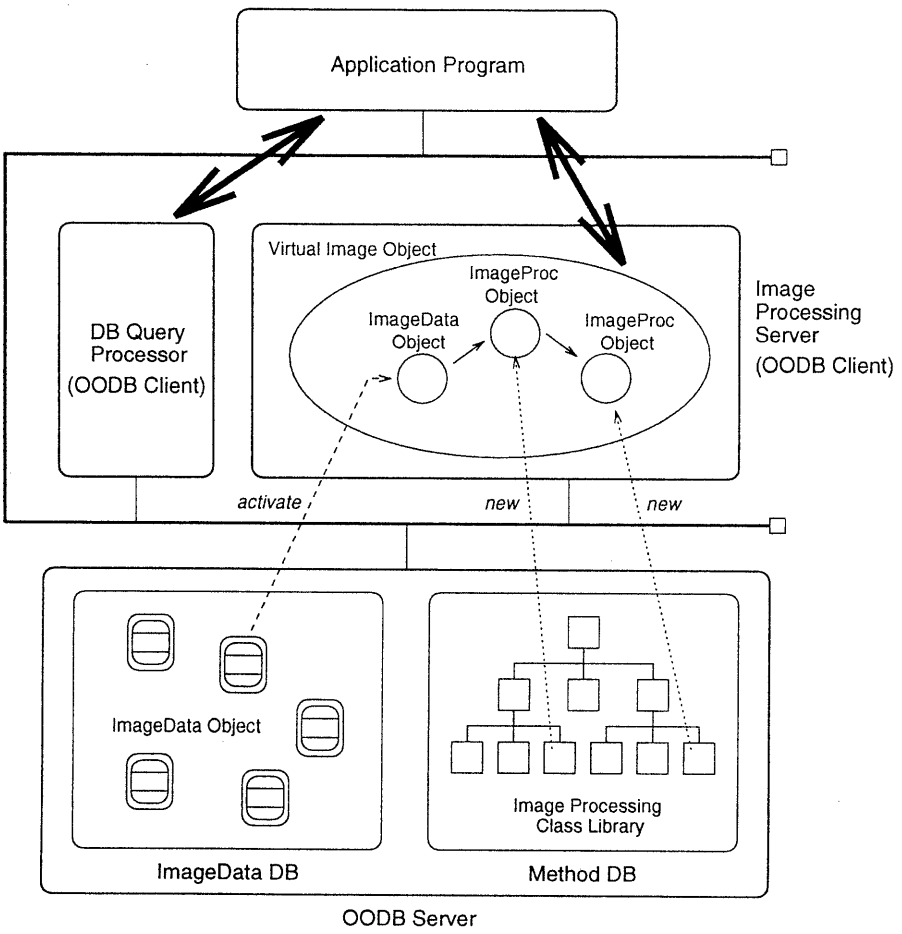

図 2: 仮想画像オブジェクトを提供するアーキテクチャ

Query プロセッサ

- 画像処理を行ない、アプリケーションに仮想画像オブジェクトを提供する画像処理サーバ

(3) アプリケーションプログラム

の3層型クライアント・サーバモデルを構成している。

3.1 画像データベース

データベースに格納する画像オブジェクトの汎用性を確保するためには、アプリケーションに依存するメソッドを画像オブジェクトのメソッドから排除しなければならない。

論文 [1] では、データベース上の画像オブジェクトのメソッドとして、画像の拡大·縮小、色の量子化などの基本的な機能を残していたが、外部の処理メソッドとの間での処理制御が複雑になってしまうため、画像処理メソッドは一切削除することにした。従ってデータベース上の画像オブジェクトが持つメソッドは、外部からの要求に応じて画像データを提供するなどの極限られたものだけになる。

3.2 画像処理サーバ

画像処理には、画像の回転、移動、拡大・縮小などの幾何学変換や、ノイズ除去、平滑化、エッジ強調、エッジ検出、輝度・色相・彩度調整などのフィルタ処理、さらにヒストグラム、分散、標準偏差などを求める統計処理など、400 個を越える非常に多くの手続きが用意されている [3]。画像処理サーバではこれらの手続き群を分類し、クラスライブラリの形でオブジェクト指向データベースで管理する。

画像処理サーバはデータベースで管理しているクラスライブラリから要求に応じて必要な画像処理オブジェクトを生成し、画像データベースに格納された画像オブジェクトと協調して、アプリケーション独自の画像オブジェクトの振舞いを提供する。一般にアプリケーション独自の振舞いを実現するためには、複数の画像処理オブジェクトの組合せが必要となる。このような機構により、アプリケーションプログラムからは、画像処理サーバ上にはアプリケーションに特化した画像オブジェクトが存在しているように見える。

画像処理サーバの働きについては次章で詳しく説明する。

3.3 アプリケーションプログラム

画像データベースと画像処理サーバを利用することで、データベースに格納されている汎用的な画像オブジェクトを様々な目的に利用することができるようになる。

アプリケーションは必要な画像オブジェクトを得るために、検索条件を添えて Query プロセッサに検索を依頼する。Query プロセッサは与えられた検索条件を元に画像データベースを検索し、該当する画像オブジェクトの OID(Object ID) をアプリケーションに返す。

アプリケーションは検索された画像オブジェクトに対して、希望する振舞いを実行するようにメッセージを送付する。このメッセージは画像オブジェクトには伝えられずに、画像処理サーバに送られる。画像処理サーバは振舞いを実現する画像処理オブジェクトをデータベース上のクラスライブラリから生成し、画像オブジェクトから画像データをコピーし、それに対して処理を適用する。

アプリケーション独自の画像オブジェクトの振舞いは、その振舞いを実現するために必要な画像処理オブジェクトの種類と、処理の適用順序の指示をスクリプトの形で用意しておく。この種の振舞いを実行する場合は、スクリプトが画像処理サーバに送られ、画像処理サーバはスクリプトに従って画像処理オブジェクトを生成し、処理を適用していく。

4 画像処理サーバの構造と機能

4.1 画像処理サーバの構造

図 3に画像処理サーバの構造を示す。画像処理サーバは、オブジェクト指向データベース管理システム (OODBMS) のクライアント・サーバモデルの構造を利用したものである。

4.1.1 OODBMS による画像処理サーバの実装

OODBMS ではオブジェクトはサーバ側に不活性な状態として格納されている。データベース上のオブジェクトに対して何らかの操作を行なう場合は、一旦クライアント側のメモリ上にオブジェクトを活性化 (activate) してから操作を行なう。新たに生成したオブジェクトもクライアント側のメモリ上に置かれ、トランザクションをコミットした時点でデータベースに書き込まれる [4]。

画像処理サーバは OODBMS のクライアントで、画像処理手続きを画像処理オブジェクトとして取り扱う。画像処理オブジェクトはデータベース上にはインスタンスを持たずスキーマだけが格納されている。このスキーマは画像処理のためのクラスライブラリそのものである。図 4はスキーマの一例である。

アプリケーションからメッセージが送られてくると、画像処理サーバはトランザクションを開始し、メッセージで指定された画像処理オブ

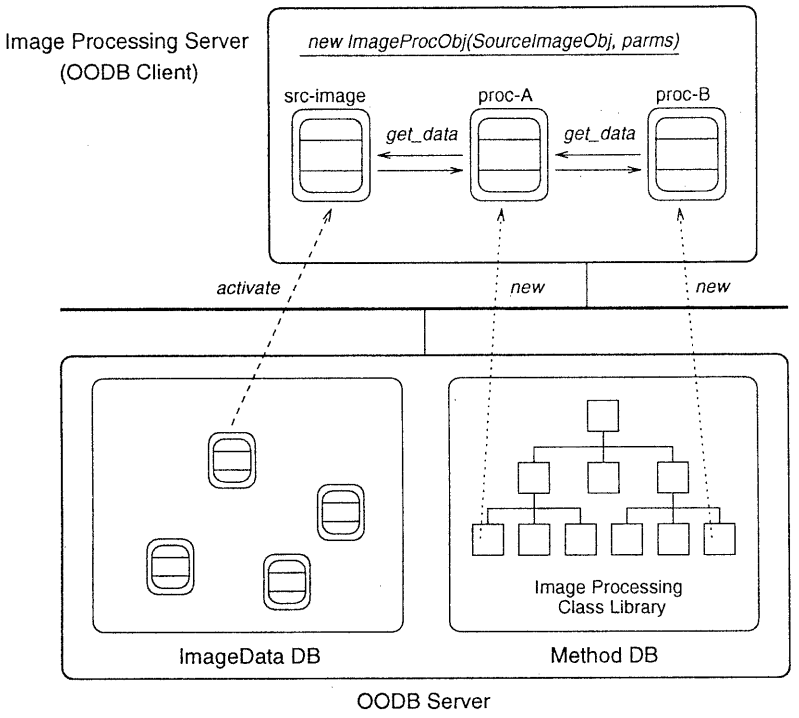

図 3: 画像処理サーバの構造

ジェクトを、OODBMS で管理しているスキーマから OODBMS のクライアントである自身のメモリ上に生成する。

画像処理サーバではアプリケーション依存の仮想的な画像オブジェクトを提供するが、この仮想画像オブジェクトの元になる画像データベースで管理されている画像オブジェクトも、画像処理サーバ上に活性化される。

4.1.2 画像処理オブジェクトの内部構造

画像処理の手続きは 3.2 節で述べたように非常に多くの種類が存在している。画像処理サーバではこれらの画像処理手続きをそれぞれ一つの画像処理オブジェクトとして実装している。

図 4では画像処理オブジェクトの典型的なものとして、Filter クラスと Statistic クラスを取り上げた。これらのクラスのオブジェクトの内部構造を示したものが図 5である。

画像処理オブジェクトは、

new ImageProcObj(SourceImageObj,parms);

というメッセージを受けとると、*SourceImageObj* で指定された原画像データを保持するオブジェクトにデータの譲渡を要求するメッセージ *get_data* を発し、得られた画像データを画像処理オブジェクト内に用意した原画像データ格納用のワークエリア (図 5中の Source Image) に格納する。続いて、*parms* で与えられた値を処理パラメータとして設定し、ワークエリアの原画像に対して画像処理を適用する。処理結果は、Filter クラスのオブジェクトでは Cooked Image にまた Statistic クラスのオブジェクトでは Statistic Data にそれぞれの画像処理オブジェクトの属性として保持される。

図 4に示すように、画像データベースに格納される画像オブジェクト (ImageData クラス) と、画像処理オブジェクト (ImageProc クラス) のう

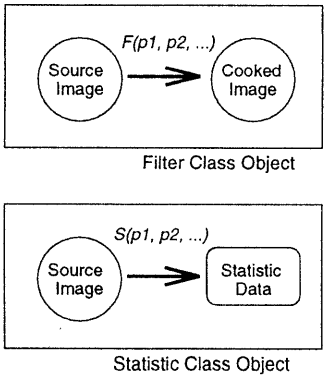

図 5: 画像処理オブジェクトの内部構造

ち処理結果を画像として保存するような Filter クラスに属する画像処理オブジェクトは、どちらも抽象クラス Image のサブクラスとして定義されている。従っていずれのオブジェクトも、ある処理を行なうために生成される画像処理オブジェクトの原画像を保持するオブジェクトとして指定することが可能である。

4.2 画像処理サーバの機能

画像処理サーバは、画像オブジェクトの汎用性と拡張可能性を実現する機構を提供している。画像データベースが様々なアプリケーションに有効利用されるためには、画像処理特有の複数の手続きによる複合処理や会話的処理を支援する機能や、画像処理の専門家でなくとも必要な画像処理手続きを選択できるような機能などが必要になってくる。

4.2.1 画像処理機能

前述したように、個々の画像処理手続きは画像処理オブジェクトの形で提供される。画像データベースに格納されている画像データをそれぞれの目的で利用するためには、複数の画像処理オブジェクトを利用した複合処理が必要になる。

例えば、画像データベースに格納されているフルカラーの画像を、256 色同時発色可能なディスプレイ環境に指定した大きさで表示することを考える。この場合の画像処理サーバでの処理手順は、

(1) 画像データベース上の画像オブジェクトを画像処理サーバ上に活性化

(2) (1) のオブジェクトを原画像としてリサイズ処理オブジェクトを生成

(3) (2) で生成したリサイズ処理オブジェクトを原画像として減色処理オブジェクトを生成

となる。この例ではフィルタ処理を直列に適用させるだけなので、処理の制御は容易である。

一方、アプリケーション固有のパターン認識を含む応用処理を行なう場合は、その処理手順はより複雑なものになる。我々が研究している歯科矯正学向けの医療画像データベースでの例で説明する。

患者のプライバシー保護のため、第三者に正面顔写真を見せる時は目線を塗りつぶした形で表示したい。この場合は、

(1) 表示サイズに画像の大きさをリサイズする

(2) 目の位置を求める

  (a) カラー画像をグレースケール画像に変換する

  (b) エッジ検出をする

  (c) 目の位置を同定する

(3) 減色のためにディザ処理する

(4) (2) で得られた目の位置を使って目線の部分を矩形に塗りつぶす

という手順をふむ。この一連の処理は、画像処理サーバ上では図 6 に示すような形態で実現される。

ある振舞いを実現するために必要な画像処理オブジェクトの種類と数、それらの適用順序は、その振舞いごとに変化する。すなわち応用的な画像処理は、一般的に図 6のようなネットワーク構造として与えられることになる。

4.2.2 処理履歴管理機能

画像を取り扱うアプリケーションでは、会話的に処理を選択・実行するような不定型処理が必要な場合が多い。前節で所望の画像を得るための手順はネットワーク構造で与えられることを示したが、そのネットワーク構造を獲得するまでには試行錯誤的な処理の適用が繰り返されることになる。この試行錯誤的な処理の適用を支援するための処理履歴の管理機能が画像処理サーバには不可欠である。

試行錯誤的な処理適用は、これまでの処理履歴の任意の時点に戻っての処理のやり直しが可能でなければならない。処理履歴の簡単な管理法としては、適用処理の種類、処理パラメータなどの情報を保存していくことが考えられる。このような管理法の場合、途中経過からのやり直しを行なうためには、最初の処理からやり直しを始める時点までの一連の処理を再適用しなければならず、非常に効率の悪いものになってしまう。効率の良い試行錯誤的な処理適用を支援するためには、各時点での処理結果も保存しておくことが必要である。

履歴管理では、ある画像オブジェクトとそれから導出された画像オブジェクトとの関係を保存することになる。これはバージョン管理と同様な問題であるが、ここではアプリケーションのレベルで履歴を管理する点が大きく異なる。

画像処理サーバでは個々の画像処理をオブジェクトの形で提供しており、各々の画像処理オブジェクトはトランザクション途中であれば処理結果を保持したまま画像処理サーバのメモリ上に存在している。このような画像処理サーバの特徴は、処理履歴管理にも有効に機能する。任意の時点からの処理のやり直しは、その時点の1つ手前で適用した画像処理オブジェクトを引数にして、やり直す処理を行なう画像処理オブジェクトを新たに生成すればよい。

4.2.3 ヘルプ機能

400 を越える画像処理手続きの中から適切な処理を選択するためには、画像処理に関するかなりの知識が要求される。たとえば微分フィルタだけでも数種類あり、さらに与えるパラメータによっても処理結果は大きく変化する。

画像データベースを有効利用するためには、画像処理の専門家でなくとも所望の画像を得るために必要な画像処理オブジェクトを選択できるようでなければならない。オンラインのヘルプ機能はその一助になる。

画像処理の詳細に不案内であっても、実際の処理適用例をいくつか見ることができれば必要な処理の選択には大いに役立つ。従ってヘルプ機能は、各々の画像処理オブジェクトに対して、原画像、処理適用後の画像、パラメータを変化させたときの処理結果の差異などが一覧できることが望ましい。

5 おわりに

オブジェクト指向データベース管理システムを用いた3層型クライアント・サーバモデルによる分散処理で、画像データベースに格納された画像オブジェクトを汎用的でかつ拡張可能なものにするアーキテクチャを提案した。

本アーキテクチャは基本的にすべてをオブジェクト指向でとらえており、異機種分散環境でのオブジェクトの相互運用性を実現する CORBA 環境 [5] にも適用できるものと考えられる。

参考文献

[1] 脇山俊一郎, 大津浩二, 福田紀彦, 金森吉成, 増永良文:オブジェクト指向データベースシステムにおける画像オブジェクトの構成と実装, 情報処理学会データベースシステム研究会,94-22(1993).

[2] Ben Shneiderman : Designing the User Interface, Addison-Wesley,Menlo Park (1987).

[3] 画像処理サブルーチン・パッケージ SPIDER USER'S MANUAL, 電子技術総合研究所 (1980)

[4] ONTOS Developer's Guide (Relese 2.1), ONTOS(1991).

[5] Object Management Group: The Common Object Request Broker: Architecture and Specification, OMG Document Number 91.12.1 Revision 1.1, Draft 10 December 1991

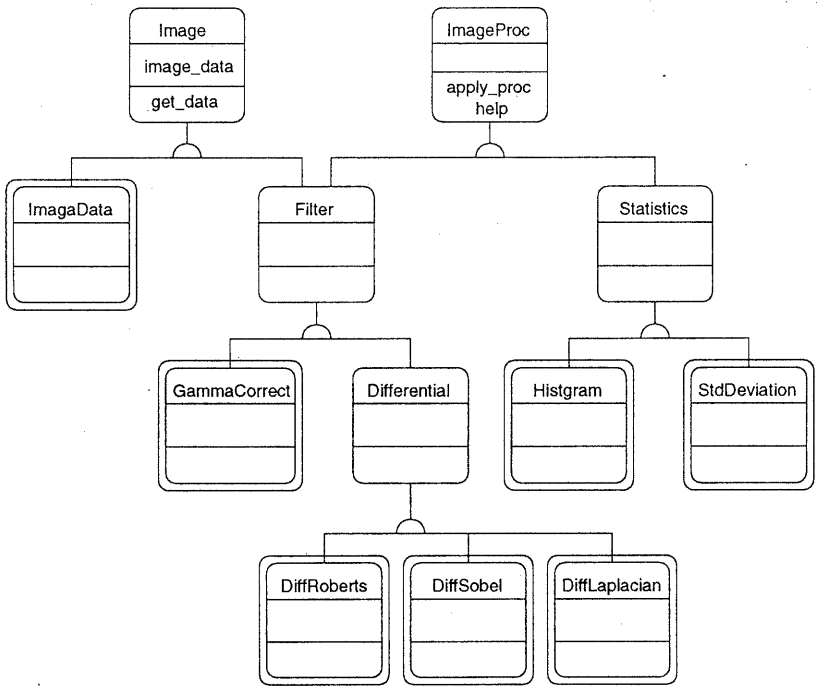

図 4: 画像処理オブジェクト (メソッド) データベースのスキーマの一例 (Coad-Yourdon 法による表現)

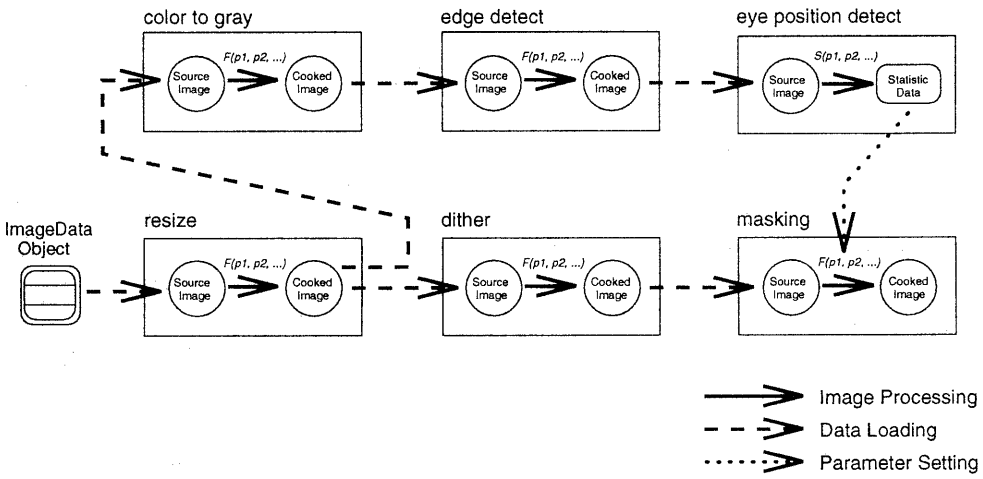

図 6: 画像オブジェクトの複合処理例