データベースシステム 101−17
（1995. 1. 27）

画像オブジェクトサーバにおける版管理モデル

川島 享[†] 金森吉成[†] 増永良文[††] 脇山俊一郎[†††]

[†] 群馬大学工学部情報工学科 [††] 図書館情報大学 [†††] 仙台電波高専

本論文では、画像処理の適用により、新たに生成される画像オブジェクトを版管理問題として捉え、その画像オブジェクトを提供するアーキテクチャ、および版管理モデルを提案する。アプリケーションには独立で、効率的かつ構造的に柔軟な画像オブジェクトを生成するため、画像オブジェクトサーバを設けた。そのサーバ上で画像処理をメソッドとして動的に結合させてオブジェクトを生成する。画像オブジェクトサーバ内には、4つの意味的相違からなる画像オブジェクトの版を管理するための Generic Image Object がある。このようにして、一連の複雑な画像処理過程の履歴をシステム側で管理するから、アプリケーションの開発は容易となる。

Version Management Model in an Image Object Server

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

[†] Department of Computer Science, Gunma University
[††] University of Library and Information Science
[†††] Sendai National College of Technology

This paper proposes an architecture to produce image objects and a version management model for objects created by applying image processing functions. Here, we look that such productions of image objects are version problem.

To make image objects that are application independent, efficient, and flexible structure, we introduce image object server into the architecture. The server creates image objects by dynamic binding among image data and processing functions. In the server, there are Generic Image Objects that manage versions of objects consisted of four different meanings. User can easily develop applications, because the system controls complex histories of a series of image processings.

1 はじめに

オブジェクト指向の画像データベースはアプリケーションに依存して構造的に柔軟性が欠ける欠点を持っている。この視点から著者等は、アプリケーションに独立した画像オブジェクトの構成とそれを実現するための画像データベースシステムアーキテクチャについて研究してきた [1, 2]。

一般に、目標とする画像オブジェクトを得るまでには一連の種々の処理を加える場合が多い。その際に、既処理のある画像オブジェクトを再利用する、または、パラメタを変更して再度処理をやり直すなどの操作が多々ある。それ故、これらの操作を効率的かつ円滑にするためには、画像処理過程の履歴を管理し、保存することが必要になる。換言すれば、個々の画像処理適用により発生する画像オブジェクトを版管理問題と捉えることができることを意味している。これらの版管理は、一般的にかなり複雑になることが予想される。そこで、アプリケーション側で版を管理することはユーザにかなりの負担を課すことになり、アプリケーション開発を困難にする。従って、システム側で版を管理、運用するデータベースシステムアーキテクチャが必要となる。

本研究では、この観点から画像オブジェクトサーバで版を管理、運用するアーキテクチャと版管理モデルを提案し、それらについて詳細に検討し考察を加えた。

2 画像処理過程の履歴

一連の画像処理の例を取り上げて説明する。例えば、"顔画像の目を塗りつぶして表示する" 場合の処理手順は、

(1) 原画像を指定した大きさにリサイズする.
(2) 目の位置を求める.
  (a) カラー画像を白黒濃淡画像に変換する.
  (b) エッジを検出する.
  (c) 目の位置を同定する.
(3) 減色処理をする.
(4) (2) で得られた目の位置を塗りつぶす.

であり、これを図 1に示す。

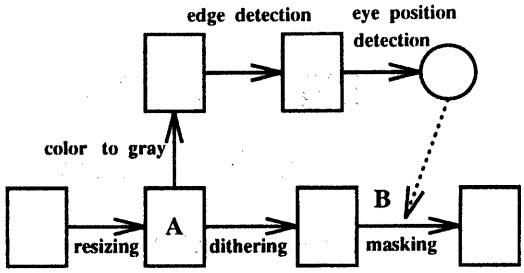

図 1: ネットワーク構造を成す画像処理の例

図 1の **A** で示す画像は、2 回使用されている。このように 1 枚の画像に対して、別の画像処理を施したり、簡単に以前の画像から処理をやり直したり、一連の処理途中のパラメタを変更するなど、処理結果の相違を確認しながら処理を進める場合が多くある。従って、処理過程のすべての画像データを保存しておかなければならない。

また、図 1の **B** で示されたように分岐処理が統合される場合 (例では、masking 処理に対してパラメタとなる)、目の位置を同定する処理を時間的に優先して行なっていなければならないという条件が加わる。それ故、処理手順間の時間的順序関係も考慮しなければならない。

上述から画像処理の履歴を管理することの必要性は明らかである。しかしながら、ユーザが履歴を管理することは、一般的に困難である。従って、システム側でこれらの履歴を管理するモデルが必要となる。

3 画像オブジェクト

3.1 画像オブジェクトのアプリケーションサイトモデルと内部モデル

ユーザ側から見た画像オブジェクトをアプリケーションサイトモデルと定義する。これは属性値に画像データ、メソッドに多種多様な画像処理を保持するオブジェクトである。これを図 2の左側に示す。しかし、実際画像オブジェクトを構成する時には、このモデルをそのまま実装することはできない。その理由として、

- 画像処理は 400 種類以上 [3] あり、アプリケーション固有の処理もある.
- 画像処理にはソフトウェアによる処理とハードウェアによる処理が存在する.

という特徴がある。これらの画像処理を一括して画像オブジェクトのメソッドとするとメソッドの更新のたびにオブジェクトを再構築しなければならない。また、アプリケーションがそれらのメソッドをすべて必要とするわけではなく、メソッドが多数ある大きなオブジェクトは無駄が多い。そのため、画像データと画像処理を分離して画像処理メソッドの階層構造を定義し、必要に応じて動的に結合することを著者等は提案した [1]。これらを考慮したモデルを図 2の右側の内部モデルとして示す。

内部モデルの画像オブジェクトでは、画像処理をその使用頻度で次の 2 種類に分類した。

- 基本画像処理
- 応用依存画像処理

基本画像処理は、画像オブジェクト自身のメソッドであり、どのような画像にも使用される使用頻度の高い処理を含む。一方、応用依存処理は、上記の基本画像処理以外の画像処理で、アプリケーション固有の処理を含む。これは、個々の画像処理を 1 つのオブジェクト (画像処理オブジェクトと定義する) とし、画像オブジェクトはその画像処理オブジェクトに処理を依頼することにより、画像処理を行なう。これを実現するため、画像オブジェクトサーバを設け、このサーバ内で画像オブジェクトと画像処理オブジェクトを動的に結合させる。上記のように画像オブジェクトは、内部モデルを用いることにより、容易に操作、管理ができて、拡張も可能となる。一方、ユーザ側にはアプリケーションサイトモデルの画像オブジェクトしか見えないから、内部モデルを全く気にする必要はない。従って、アプリケーション開発は容易となる。

3.2 画像データベースシステムアーキテクチャ

画像オブジェクトを提供するアーキテクチャとして 3 階層から成るモデルを提案した [2]。これを図 3に示し、概要を以下に説明する。

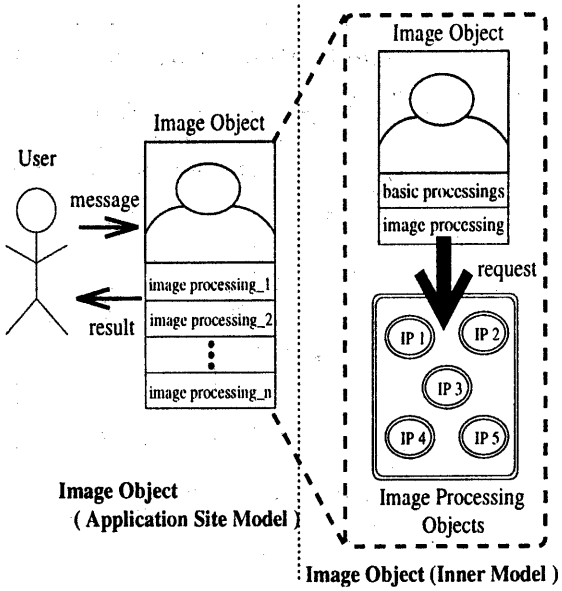

図 2: 画像オブジェクトのアプリケーションサイトモデルと内部モデル

- **アプリケーション層**
  - アプリケーションプログラム
    アプリケーションが容易に開発できるスクリプト言語の設計を行なった [4]。ここではこの言語の利用を想定している.
- **中間層**
  - **Query Processor**
    画像データオブジェクトの検索.
  - **画像オブジェクトサーバ**
    画像オブジェクトの動的生成、履歴管理、及び画像処理オブジェクトの検索、実行.
- **データベース層**
  - **画像データオブジェクト DB**
    画像データオブジェクトを管理する。ここで画像データオブジェクトとは、属性値に画像データを保持し、画像処理メソッドを全く持たないオブジェクトである。これは画像オブジェクトの属性値となる画像データとして利用される.
  - **画像処理オブジェクト DB**
    画像処理オブジェクトを管理する.

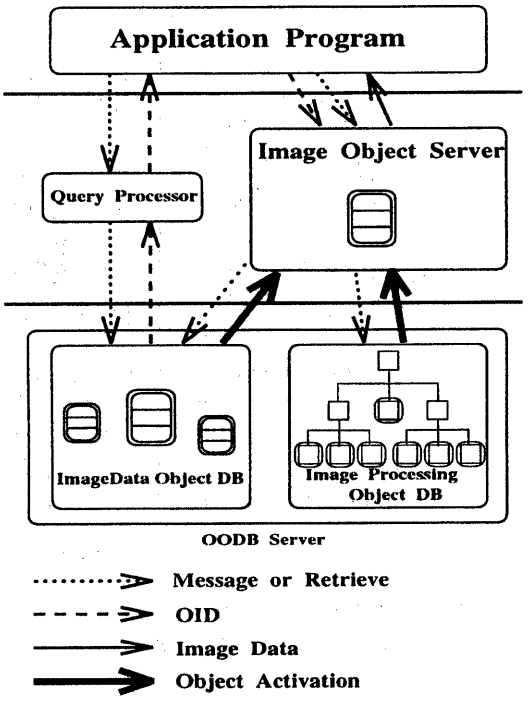

図 3: 画像オブジェクトを提供するアーキテクチャ

また、個々のモジュール間でオブジェクトやメッセージなどのやりとりを説明すると次のようになる。

- **アプリケーション ⇔ Query Processor**
  アプリケーションは OQL 文 [5] を Query Processor にメッセージとして渡し、その結果 (OID 相当) を Query Processor から受けとる.
- **Query Processor**
  ⇔ 画像データオブジェクト DB
  Query Processor はアプリケーション側からの OQL 文により画像データオブジェクトを検索し、その結果 (OID 相当) を受けとる.
- **アプリケーション**
  ⇔ 画像オブジェクトサーバ
  検索された画像データオブジェクトの OID により、画像オブジェクトの生成を要求する。また、画像処理の要求を行ない、その結果、必要に応じてサーバは属性値の画像データをアプリケーション側へ返す.
- **画像オブジェクトサーバ**
  ⇔ 画像データオブジェクト DB
  アプリケーション側より渡された OID に基づき画像データオブジェクトを検索し、それを活性化する.
- **画像オブジェクトサーバ**
  ⇔ 画像処理オブジェクト DB
  画像オブジェクトに画像処理の要求があった場合、その画像処理に相当する画像処理オブジェクトを活性化させ、画像処理を行なう.

3.3 画像オブジェクトの生成方法

画像オブジェクトを生成すると、その画像に対してアプリケーション側でユーザは画像処理を実行することができる。画像オブジェクトサーバ内での画像オブジェクトの生成には、次の 2 種類が考えられる。

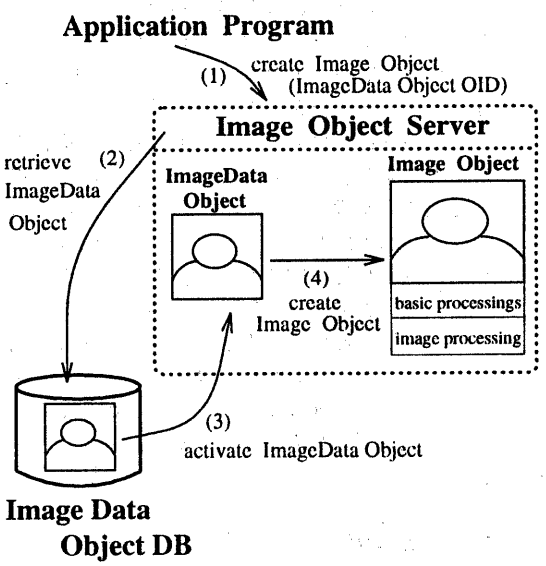

図 4: 画像オブジェクトの生成

(i) 画像データオブジェクトから最初の版の画像オブジェクトを生成する.

(ii) 既存の画像オブジェクトに、画像処理を要求する。換言すれば、画像処理をすることにより別の版の画像オブジェクトができる.

ここでは (i) の画像データオブジェクトを用いた画像オブジェクトの生成手順を以下に説明し、その様子を図 4に示す。

(1) アプリケーション側からある画像データオブジェクトを画像オブジェクトにせよというメッセージ (OID 相当を含む) を画像オブジェクトサーバに送る.

(2) 画像オブジェクトサーバは OID に基づき画像データオブジェクトを検索する.

(3) 検索された画像データオブジェクトを画像オブジェクトサーバ上に活性化する.

(4) 活性化された画像データオブジェクトから画像データを受けとることにより画像オブジェクトを生成する。生成時に基本画像処理メソッドが自動的に加わる。このオブジェクトは図 2の内部モデルの画像オブジェクトに相当する.

3.4 画像処理の方法

3.1節で述べたように画像オブジェクトにおける画像処理メソッドは 2 種類ある。

その 1 つの基本画像処理は、一般のオブジェクトがメソッドを実行するようにして画像処理を行なう。もう一方の動的に結合する応用依存画像処理の動作方法をエッジ処理を例に用いて以下に説明する。これは、3.3節の (ii) による画像オブジェクト生成である。この様子を図 5に示す。

(1) アプリケーション側から既存の画像オブジェクトにエッジ処理を施すメッセージを送る.

(2) 画像オブジェクトは、エッジ処理オブジェクトを検索する。この検索は画像オブジェクトのメソッド image processing が実行する.

(3) 画像オブジェクトサーバ上に、エッジ処理オブジェクトを活性化する.

(4) 画像オブジェクトは、画像データとパラメタをエッジ処理オブジェクトに渡す.

(5) エッジ処理オブジェクトは受けとったデータを元に画像処理を行ない、その結果を新たな版の画像オブジェクトとして生成する.

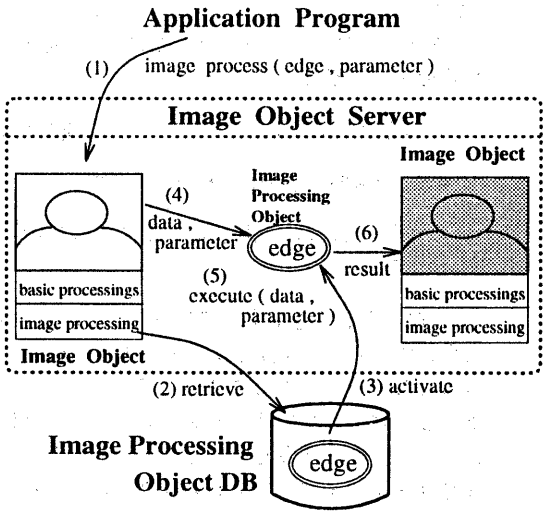

図 5: 画像オブジェクトにおける応用依存画像処理の動的結合動作

4 画像オブジェクトサーバにおける版管理

従来、主に CAD 分野においてデータベース内のインスタンスの版管理が研究されてきた。また、オブジェクト指向データベースでは、インスタンスのみならずスキーマ (クラス) の版管理についても研究がなされている [6, 7, 8]。

これらの研究における版管理では、個々の版の間の意味的関係を導出関係 (derived-from 関係) と version_of 関係 (kind_of 関係、IS_A 関係とも呼ばれる) で表現し、版階層構造 (version hierarchy) を形成する。また、抽象化した "generic object" がこれらの版の共通の属性と個々の版の履歴構造を保持することにより、階層構造全体を統合的に管理する。

4.1 画像オブジェクトの版管理モデル

2節で述べたように、画像処理を施す毎に画像データは変化していく。つまり、画像処理を施すたびに新たに画像オブジェクトが生成される。これらの個々の画像オブジェクトを1つの版と考え、アプリケーション上の画像オブジェクトを generic object(Generic Image Object と呼ぶ) と見なすことにより、版管理問題として捉えることができる。これを示したのが、図6である。

画像オブジェクトの版管理モデルの特徴として次の3つが挙げられる。

- 画像オブジェクトサーバでの版管理である.
- 画像処理の適用による関係を表す.
- 複数の generic object が1つの版を参照する.

一般的な版管理はすべてデータベース内の版管理、すなわち、永続オブジェクトの版管理を対象としている。これに対し、画像オブジェクトの版管理モデルでは、アプリケーションが動作している間、画像処理を行なうことにより生成される画像オブジェクトを対象としている。従って、アプリケーションが終了すると共に管理も終了となるのでアプリケーション側での版管理であり、永続的オブジェクトを対象としていない。これが画像オブジェクトモデルにおける版管理の第1の特徴である。

4.2 意味的相違による版関係の表現

画像オブジェクトの版管理モデルにおける第2の特徴として、画像処理により、個々の版のには意味的に異なる次の3種類の関係があることが挙げられる。

- 導出関係 (derived_from relationship)
- パラメタ関係 (parameter_from relationship)
- 融合関係 (merged_from relationship)

これらの関係を画像処理の例を用いて説明する。まず、“顔画像の大きさを変更し、減色処理をする”場合は、原画像の顔画像をリサイズ処理をして、減色処理を施すので、それらの版の間は導出関係で結ばれる (図6の1 ~ 3)。

次に“この画像の目を塗りつぶして表示する”場合は、リサイズ後の画像を利用して、目の位置を同定する。この結果、目の位置は座標値 (x,y) で与えられる。図6の6が円であるのは、画像データが数値データに変化したことを意味する。その後、その目の位置をパラメタにして目を塗りつぶす画像処理を施す (図6の7) ので、これがパラメタ関係となる。

最後に“目を塗りつぶした画像に顔の輪郭線を重ねて表示する”場合を考えると、前処理でエッジ処理を施した画像 (図6の5) を用いて顔の輪郭線を抽出して、それを先ほどの画像と重ね合わせる (図6の7 ~ 9)。この重ね合わせが融合関係となる。

このように画像オブジェクトの版管理モデルに3種類の関係を導入することにより、ある版から導出されたり、パラメタとなったり、複数の版が融合することが可能となる。この一連の流れは、順序が重要であり、これを無視すると所望の結果が得られない。

一方、個々の版と Generic Image Object の関係を version_of 関係で結ぶことにより、Generic Image Object は版階層構造を管理することができる。そこで、この Generic Image Object を操作して、個々の画像オブジェクトに画像処理を行なわせることができる。また、Generic Image Object と1つの版の間の関係を見るとそれは画像オブジェクトのある時点での状態を表している。

4.3 画像オブジェクトサーバの機能

画像オブジェクトサーバは、画像オブジェクトの版管理を行なう。このサーバの主な役割は以下の通りである。

- 個々の版や Generic Image Object に識別子を与える.
- 個々の版や Generic Image Object を管理運用し、画像処理を実行する.

後者は、これまでに述べてきた通りであり、Generic Image Object や画像オブジェクトの

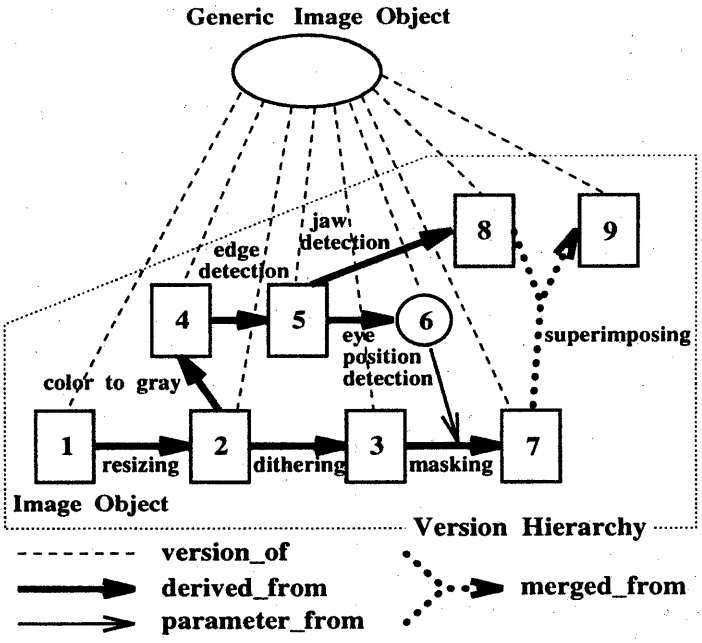

図 6: 画像オブジェクトの版管理モデル

管理運用や画像処理を実施する役割である。一方、前者は個々のオブジェクトを区別するためにオブジェクト識別子を付加する役割を持っている。この識別子を Generic Image Object では GID(Generic image object IDentifier) とし、画像オブジェクトは IID(Image object IDentifier) とする。ここで IID は、サーバ内で唯一と定義する。以下にこの理由を例を用いて示す。

例えば、前節の例で "顔画像の大きさを変更して、減色処理を施した結果を画面上に表示する" ことを考えてみる。この手順は、

(1) Generic Image Object **gen1** の生成.

(2) 顔画像を画像データオブジェクト DB から検索し、画像オブジェクト **io0** を生成.

(3) リサイズ処理 (**io1**)、ディザ処理 (**io2**) を実施して画面上に表示.

であり、図 7の点線の枠内にこれを示す。次に "画面上の別の場所に先ほどの結果に反転処理を施して表示したい" とする。この場合、画面上の画像オブジェクトが Generic Image Object **gen1** に相当するので、別の Generic Image Object を新たに生成しなければならない。手順としては、

(1) 新しく Generic Image Object **gen2** を生成.

(2) **gen1** の関連のある版と **gen2** を version_of 関係で結ぶ.

(3) 最後の画像オブジェクト (**io2**) に対して反転処理を施す (**io3**).

である。このようにすると、それまでのオブジェクトには何ら影響なく資源、時間共に節約できる。ここで **gen2** を設ける理由は、もし **io3** が **gen1** の版とすると、**gen1** が削除されたら、現在表示されている **io3** も同時に削除されてしまう。そこで上述のようにして Generic Image Object を互いに独立にし、**gen1**、**gen2** に新しく版が追加されても、あるいはどちらかの Generic Image Object が削除されても他に影響を及ぼさないようにしなければならない。また、図 7の

Generic Image Object **gen3** のように同じ原画像にリサイズ処理を施して使用する場合には、まず、サーバ上にその画像が存在するか確認し、存在すればそれを利用する。これらから複数の generic object を設けることの利点が明らかである。また、これらの Generic Image Object は画像処理により変化したオブジェクトの異なった View を表現しているとも解釈できる。

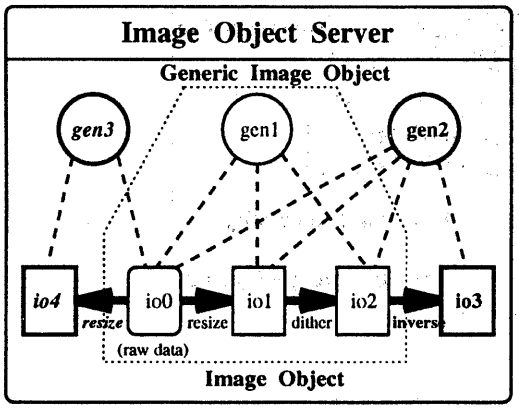

図 7: 画像オブジェクトサーバにおけるオブジェクト識別子

このように、画像オブジェクトの版管理モデルの第 3 の特徴は、資源、時間共に節約のため、複数の Generic Image Object から 1 つの画像オブジェクトを参照することであり、このため画像オブジェクトの識別子 IID は画像オブジェクトサーバ内で唯一にしなければならない。

5 おわりに

画像オブジェクトに各種画像処理を要求することにより、その画像オブジェクトの属性値である画像データが変化する。これを新たな画像オブジェクトの生成、すなわち、オブジェクトの版の生成と捉えることにより、画像処理適用を版管理問題と見ることができる。アプリケーション側で、版の発生、削除を管理することはユーザに多大な負担を課すことになる。また、資源を効率的に利用する見地からも、繰り返して同一画像処理の実行を避けることが望ましい。これらの視点から、画像処理過程を版の履歴として保存し、画像オブジェクトサーバでシステムが管理、運用するアーキテクチャを提案した。

版管理には、derived_from 関係、version_of 関係などの従来の関係に加えて、画像処理特有な関係である parameter_from 、merged_from が必要であることを示した。アプリケーション側ではこれらの関係で表現された Generic Image Object を操作することにより、個々の版の管理を気にすることなく、より容易にかつ効率的に画像処理結果を求めることが可能となる。

今後、画像オブジェクトサーバの実装についても検討する予定である。

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, Vol. 94, No. 22, 7 月 1993.

[2] 脇山俊一郎, 川島享, 金森吉成, 増永良文. 分散環境における汎用画像オブジェクトの構成. 情報処理学会データベースシステム研究会, Vol. 99, No. 11, 7 月 1994.

[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] 大津浩二, 松田宜之, 金森吉成, 増永良文, 脇山俊一郎. 画像 DB 用アプリケーション開発言語の設計. 情報処理学会データベースシステム研究会, Vol. 100, No. 7, 10 月 1994.

[5] *The Object Database Standard: ODMG-93.* Morgan Kaufmann Pablishers, 1994.

[6] Randy H. Katz. Toward a Unified Framework for Version Modeling in Engineering Databases. *ACM Computing Surveys*, Vol. 22, No. 4, December 1990.

[7] G. Talens, C. Oussalah, and M.F. Colinas. Versions of Simple and Composite Objects. *Proceedings of the 19th VLDB Conference*. pp. 375-408, 1993. Dublin,Ireland.

[8] Won Kim. *Introduction to Object-Oriented Databases.* The MIT Press, 1990.