半構造データモデルによる画像処理履歴の管理

田幡 勝[†,††] 有次正義[†] 金森吉成[†]

画像データベースを扱うアプリケーションプログラムでは，画像処理演算は必須の要素である．しかし，画像処理の詳しい知識を持たないユーザは，画像処理演算の実行順序を決定したり適切なパラメータを画像処理演算に指定することはできない．このため，我々は画像処理の専門家が作成した画像処理演算の実行手順を再利用する方法を提案してきた．我々は，この実行手順を画像処理履歴と呼んでいる．画像処理履歴はその履歴に含まれる画像処理演算の種類やその演算に用いたパラメータの型がそれぞれ異なったものとなる．このため，画像処理履歴を半構造データとみなすことができる．本論文では，画像処理履歴に半構造データモデルを適用し，その履歴を管理し問合せる方法を述べる．そして，半構造データを扱うためにワイルドカードや正規表現を持った問合せ言語が画像処理演算の実行順序を指定することに有効であることを示す．

Managing histories of image processing by a semi-structured data model

MASARU TABATA,[†,††] MASAYOSHI ARITSUGI[†]
and YOSHINARI KANAMORI[†]

Image processing operators are essential elements in applications of image databases. However, for users who do not have knowledge of image processing in detail, it is difficult to determine execution order of image processing operators and to specify appropriate parameters for them. We have proposed a way to reuse execution plans created by image processing experts. We call execution plans "histories of image processing." Histories of image processing can be considered as semi-structured data because various operators are included in an individual history and parameter types used in them are different from one another. In this paper, we apply a semi-structured data model to histories of image processing, and describe how to manage and retrieve these histories. Then, we show that the query language with wild cards and regular expressions for treating semi-structured data is effective for specifying execution order of image processing operators.

1. はじめに

リモートセンシング画像をカテゴリ分類するといった操作や医療画像内の腫瘍を鮮明にするといった操作は，複数の画像処理を組合せて実現される．このような一連の画像処理演算の組合せは複雑なものとなる．このため，画像処理の専門的な知識や，リモートセンシングや医療画像特有の知識を持たないと画像処理演算の実行順序やその画像処理演算で用いるパラメータの適切な値を指定することは困難である．

この課題を解決するため，画像処理の専門家が画像処理演算を実行したときの手順をデータベースに格納し他のユーザが再利用することによって専門的知識の不足を補うことを考える．また，専門家がどのような画像処理演算の手順を実行したかを知ることによって，画像処理演算に関する知識を学習できる．

このため，我々は画像データベースから検索した画像に対して一連の画像処理を実行したときに生じる結果を画像オブジェクトの版としてデータベースで管理し，同時に画像処理の手順に関する情報も管理するモデルを提案し，その機構を実現した[3),9),15)]．ここで，画像処理演算の順序およびそれらの画像処理演算で用いられたパラメータ値のことを画像処理履歴と呼ぶことにする．複数のユーザが画像処理履歴のデータベースを共有することで，過去に作成された一連の画像処理演算の実行手順を再利用することができる．データベースシステムがこのような機構をユーザに提供することは，画像データベースのアプリケーション開発にとって非常に有効であると考える．

† 群馬大学工学部情報工学科
Department of Computer Science, Gunma University
†† 日本学術振興会特別研究員
Research Fellow of the Japan Society for the Promotion of Science

そこで，画像処理履歴をどのようなデータモデルで表現し，問合せを実現するかが課題となる．一般に，画像処理履歴に含まれる

- 実行された画像処理演算の種類および個数．
- 実行された画像処理演算の組合せ．
- 個々の画像処理演算が必要とするパラメータの型および個数．

などは履歴毎に異なるものとなる．したがって，画像処理履歴は不規則なデータであり，あらかじめ画像処理履歴に対するスキーマを定義することは困難である．このような「構造が不規則なデータの集まりではっきりしたスキーマが定義できないようなデータ群」[16]は，半構造データと呼ばれる．半構造データに対する厳密な定義は存在しておらず数多くのデータ群が半構造データとして定義されているが[1),16)]，画像処理履歴も不規則なデータ群であるから半構造データであるとみなすことができる．

そこで，本研究では画像処理履歴の検索機構を実現するため，半構造データモデル OEM(Object Exchange Model)[2),11)] と OEM に対する半構造データベースシステム Lore を利用した画像処理履歴の表現方法を検討する．これらを利用した理由は，このデータモデルはグラフ構造で表現されるため画像処理履歴の表現に適しているからである．このモデルにおいて画像処理履歴はエッジとして画像処理演算，ノードとしてその画像処理演算の実行前後の画像を保持する有向グラフによって表現される．

画像処理履歴を検索するとき，その全ての構造を正確に指定することは一般に困難である．半構造データモデルは従来のデータモデルよりデータ型の指定が緩やかであるため，半構造データに対する問合せ言語は従来の SQL や OQL のような問合せ言語よりも柔軟なものとなる．この特徴の一つに正規表現とワイルドカードを用いたパス表現 (path expression) の記述が挙げられる．正規表現とワイルドカードを用いることによって，画像処理手順の一部分を指定するだけで画像処理履歴が検索できるようになる．一方，同様の問合せを SQL や OQL 等の問合せ言語を用いて記述することは困難である．さらに別の検索方法としてカテゴリとコンテキスト[5)] の概念を導入することによって，複雑な画像処理履歴を検索することもできる．これらの方法を組合せて使うことによって画像処理履歴に対する柔軟な問合せが記述でき，画像処理履歴の再利用が可能になる．また，検索された画像処理履歴の構造から，画像処理演算の利用方法を学習することもできる．

本論文の構成は，以下のとおりである．まず，2 章で画像処理によって生じる画像の版と画像処理履歴の管理方法の概要を述べる．次に，3 章で画像処理履歴の問合せに必要な条件について検討する．そして，4 章で OEM による画像処理履歴の表現方法を説明する．それから，5 章で Lore の問合せ言語 Lorel[2)] を用いた画像処理履歴の問合せ例を示す．6 章で関連研究について概要を述べ，最後に 7 章でまとめとする．

2. 版管理機構

2.1 版管理モデル

我々はこれまでに画像データベースから検索した画像に対し一連の画像処理を実行したときに生じる結果および画像処理履歴を画像オブジェクト (Image Object, IO) の版として管理するためのモデルを提案し，その実装を行ってきた[3),9),15)]．版管理の目的をまとめると，以下の二つが挙げられる．

(1) 一連の画像処理過程における途中結果を版として管理し，特定の時点における画素値データを再利用する．

(2) ある画像に対して一連の画像処理演算を実行したとき得られた画像処理手順の履歴を再利用する．

前者の管理は，汎画像オブジェクト (Generic Image Object, GIO) によって行われる．後者の目的は，画像処理履歴オブジェクト (History of Image Processing Object, HIPO) によって実現される．

GIO による版管理の例を図 **1** に示す．図 1 では，番号が付いた長方形 (画素値データを保持) と円 (数値データを保持) が IO，楕円が GIO，点線が version_of 関係を表している．また，矢印が画像処理を表してい

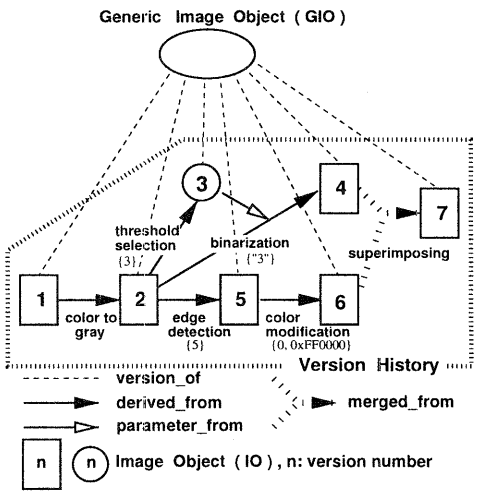

図 1 汎画像オブジェクト
Fig. 1 A generic image object

る．画像処理には，画像処理前後のIOの関係の違いによって，次の3種類の関係がある．

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

あるIOへ画像処理を実行した結果，新たなIOが生成されたとき，これらのIOの間を導出関係で関連付ける．例えば，図1ではIO2からIO5がエッジ検出処理によって導出されていることが表されている．エッジ検出処理の下の括弧内の数字5は，エッジ検出処理に用いたパラメータを表している．

また，IO2に対して2値化のしきい値検出処理を行った結果がIO3である．この処理は画素値データでなく整数型のしきい値を返すため，画素値データと区別するためにIO3は円で表現されている．さらに，2値化処理によって白黒濃淡画像のIO2から2値画像のIO4を導出している．このとき，2値化処理のしきい値としてIO3の持つ値をパラメータとして用いている．このIO3とIO4の間の関係をパラメータ関係と呼ぶ．また，IO4は2値化処理によってIO2から生成されているので，IO2とIO4は導出関係で関連付けられる．2値化処理の下の括弧内の文字列"3"はパラメータとしてIO3を用いたことを表している．最後の融合関係は，同一のGIOで管理されている複数のIOを用いて画像処理を実行した結果，新たなIOが生成されたことを意味する．IO7は，IO4とIO6を重ね合わせた結果として生成されるため融合関係で関連付けられる．

GIOに管理される一連の画像処理手順を再利用するため，GIOからHIPOを作成する．画像処理履歴を再現するとき途中結果の画像は必要ないため，HIPOは途中結果の画像データを保持する必要がない．すなわち，HIPOはGIOの持つ画像処理の実行順序とパラメータに関する属性から構成される．

ユーザがHIPOに対し画像データベースから検索した画像データを処理することを依頼したとき，HIPOが保持する画像処理履歴に基づいて一連の画像処理演算がその画像データに対して実行される．

2.2 システム構成の概要

我々は，画像の版および画像処理履歴を管理する機構を実現した[15]．この管理機構を異種分散環境で提供するため，CORBA[13]を用いて実装している．図2はこの機構の概要を示し，この図中の矢印はCORBAオブジェクトへのリモートメソッド呼出しを表す．画像処理演算を実行するため，ユーザは画像オブジェクトサーバ（Image Object Server, IOS）で管理されるGIOに画像処理演算の実行を依頼する．GIOおよびHIPOは，各ユーザ毎に管理される版データベース（Version DB）に保存される．版データベースは，ObjectStore[12]を用いて実現されている[15]．

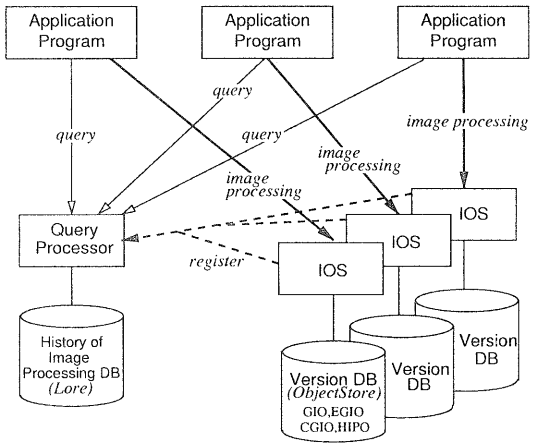

図2 版管理機構の概要
Fig. 2 An overview of the versioning mechanism

画像処理履歴データベース（History of Image Processing DB）は，各版データベースに対するHIPOに関する問合せを実現するためのものである．各版データベースにはHIPOの実体が格納されているが，画像処理履歴データベースにはそれらのHIPOを検索するためのOEMによる画像処理履歴表現が格納される．この表現については，4章で詳細に述べる．

ユーザは，query processorを介して画像処理履歴データベースに対して画像処理履歴の検索を行うことによって，各ユーザに管理されている版データベース内のHIPOへの参照を得る．IOS上のHIPOはCORBAオブジェクトとして実装されているため，ネットワーク上の各種のアプリケーションプログラムからHIPOを再利用することができる．

3. 画像処理履歴の問合せ方法

各ユーザが画像処理履歴データベースを用いて画像処理履歴を共有するためには，その履歴を検索する手段が必要となる．ここでは，以下に述べる3種類の検索方法を対象とする．一般的には画像処理履歴の作成者や作成日を指定するような検索方法も考えられるが，画像処理履歴特有の問題ではないため本論文では扱わない．

3.1 カテゴリによる検索

最も単純な検索方法は，アプリケーション毎に画像処理履歴を分類しておき，その分類に基づいて検索することである．例えば，リモートセンシングを対象とした画像処理履歴では，“土地利用区分画像の作成”，“指定領域の面積計算”などが考えられる．そこで，画像処理履

歴を分類するカテゴリを予め作成し，そのカテゴリをキーとして検索する．

3.2 コンテキストによる検索

画像処理履歴のカテゴリが同じであったとしても，各画像処理履歴に含まれる画像処理演算やそのパラメータが違う場合がある．例えば，“土地利用区分画像の作成”というカテゴリを持った画像処理履歴を考える．このとき，市街地の画像データに対する画像処理履歴と山間部の画像データに対する画像処理履歴では，それぞれが対象とする画像の種類が異なるため，各履歴に含まれる画像処理演算の組合せは異なったものとなる．このように，入力とする画像データが異なる場合，目標は同じであっても画像処理履歴の内容が異なるものとなる．

このため，画像処理履歴が対象とする画像の種類を特定した検索を行う必要がある．入力画像に関する情報をBorg[5]ではコンテキストと呼んでおり，我々もこれをコンテキストと呼ぶことにする．

例えば，リモートセンシングを対象とするアプリケーションでは入力画像のコンテキストとして

- 撮影した衛星
- 撮影したセンサのバンド番号
- 撮影した画像の緯度，経度
- 撮影した画像の地域
- 画像の解像度
- 撮影日時

等の属性を挙げることができる．

3.3 パス表現による検索

ある画像処理演算を実行することは分かっているが，その後にどの画像処理演算が必要であるか分からない場合が考えられる．同様に，途中に実行すべき画像処理演算を知っているが，その前後に必要な画像処理演算を知らないという場合もある．この場合，ユーザが持つ一部分だけの情報に基づいて画像処理履歴を検索する手法が必要である．

図1に示されるGIOは，エッジとして画像処理演算，ノードとしてその画像処理演算の実行前後の画像を持つ有向グラフによって画像処理履歴を表現している．このグラフに対し，ある画像処理演算を含んだパスを指定することによって，履歴を検索することができる．これは，半構造データベースで提案されているパス表現による検索手法と同じであると考えることができる．このパス表現では正規表現やワイルドカードを用いることができるため，画像処理履歴の一部分だけを指定した問合せを記述することが可能である．

半構造データモデルは，上記の二つの検索の要求事項も表現できることから，画像処理履歴を半構造データモデルで表現する．次に，画像処理履歴データベースを実現するための半構造データモデルによる画像処理履歴の表現方法を説明する．

4. OEMによる画像処理履歴の表現

前述したように画像処理履歴を半構造データとみなすことができる．そこで，上記の3種類の検索方法を実現するため，画像処理履歴をOEM(Object Exchange Model)[2),11)]に従って表現する．OEMは，オブジェクトとオブジェクト識別子(oid)の概念を基本とし，各データはint，string等のatomic型またはオブジェクトリファレンスの集合であるcomplex型のいずれかとなる．complex型のオブジェクトをノード，atomic型のオブジェクトをリーフ，オブジェクトリファレンスをエッジとするグラフが構成される．オブジェクトリファレンスは，(ラベル，参照先のオブジェクトのoid)の対として記述される．スキーマ情報は予め定義されるのではなく，各オブジェクトの型を表すラベルとそのオブジェクトが保持する属性のラベルによって記述される．したがって，画像処理履歴を表すラベルと各オブジェクトの属性を定義することが必要となる．

まず，ラベルの定義から説明する．各ラベルを説明するにあたって，図1に示した画像処理履歴をOEMで表現した図**3**を用いる．図3では，oid &19のオブジェクトは，二つのオブジェクトリファレンス(parameters, &20)と(ip_ColorModification, &22)を持つcomplex型のオブジェクトであり，&19から画像処理演算ColorModificationによって&22を作成したことが表されている．また，oid &21のオブジェクトは，値5を持つint型のatomicオブジェクトである．画像処理履歴は，以下に示すラベルによって構成される．

HDB 画像処理履歴データベースのエントリポイントを表すラベル．図3では&1がエントリポイントである．

*input_*⟨入力画像数⟩ 画像処理履歴を作成するのに必要な入力画像の枚数を表す．図3では，ラベル*input_1*が入力画像が1枚の画像処理履歴の集合を表している．入力画像が複数枚の場合は後述する．

root 各HIPOの画像処理履歴の出発点を表す．図3では，HIPO-Aが保持する画像処理履歴の先頭となる&3を参照している．

*ip_*⟨画像処理オブジェクト名⟩ 画像処理演算はオブジェクトとして実現されている．このオブジェクトを画像処理オブジェクト(Image Processing Object, IPO)[3),9),15)]と呼び，個々のIPOは単一の画像処理演算に対応する．そこで，ある画像処

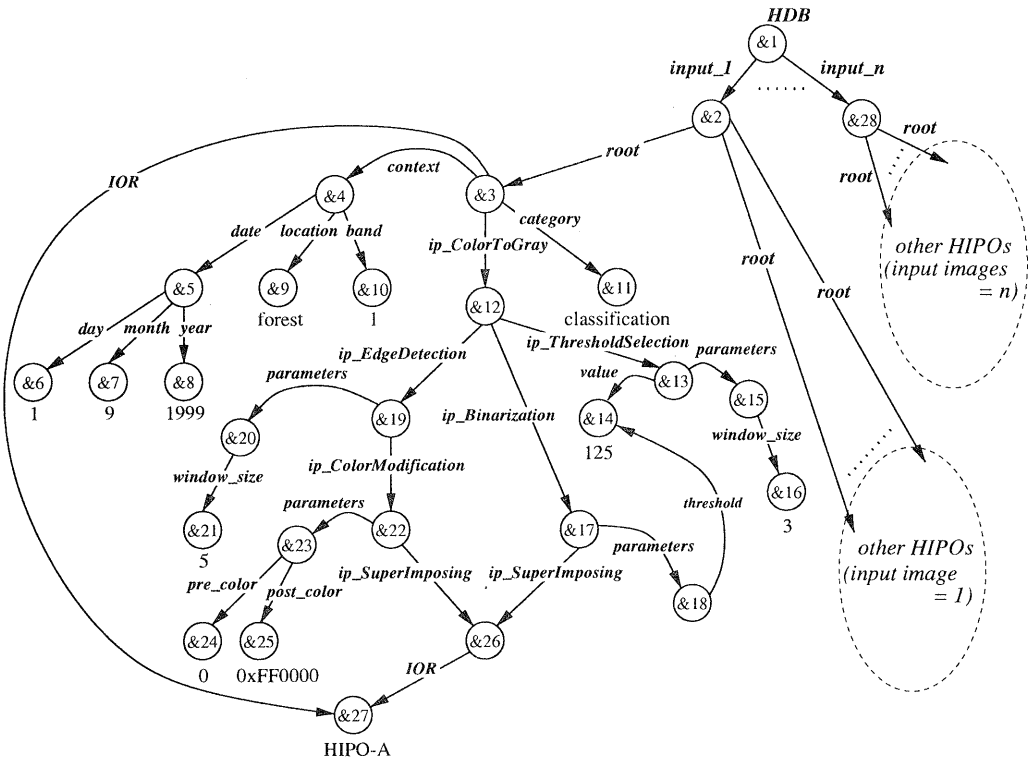

図 3 画像処理履歴の OEM による表現
Fig. 3 A representation of histories of image processing in OEM

理演算によってある画像から他の画像が作成されたことを，その画像処理オブジェクトの名前で表現する．例えば，画像処理オブジェクト ColorToGray を用いたことが，&3 から&12 への ip_ColorToGray で表現されている．画像処理オブジェクト名には接頭語として必ず“ip_”を付加させる．この接頭語が必要な理由は，5章で述べる．また，画像処理前後の画像は 1 対 1 に対応するとは限らず，複数枚の画像を入力として，それらを融合する場合もある．例えば，ip_SuperImposing は 2 枚の画像&17 と&22 を入力とし&26 を出力としている．

value 画像処理演算の結果として得られた数値データを表す．例えば，ip_ThreshholdSelection を&12 に対して実行した結果は，int 型の 125 であり，その値を処理結果に対応するノード&13 の属性 *value* として保持させている．

parameters および〈パラメータ名〉 *parameters* は，画像処理演算に用いたパラメータの集合を表す．また，個々のパラメータを表すラベルは，そのパラメータの意味を表す文字列で表現する．例えば，図 3 では，&12 から&19 への画像処理演算 ip_EdgeDetection のパラメータの集合として&20 を用いていることを表している．この画像処理演算は一つのパラメータだけをとり，そのパラメータ window_size に用いられた値が 5 であったことを&21 が表している．また，&12 から&17 への画像処理演算 ip_Binarization で用いたパラメータ threshhold は，&18 から&14 を指している．これは，ip_ThresholdSelection によって求められた値 125 を threshold に指定したことを意味している．

IOR 各ユーザが管理する版データベース内に格納されている HIPO の識別子．版データベースに格納されている HIPO は，CORBA オブジェクトとして実装されており，その識別子として IOR(Interoperable Object Reference)[13] を持っている．IOR は 16 進数に符号化した文字列に変換され，ラベル *IOR* の指すオブジェクトの属性値として保持される．ただし，図 3 では，図を単純化するために&27 の持つ値は単に HIPO-A としている．*IOR* は，実行した画像処理演算の先頭と末尾のノードだけに保持される．図 3 では，HIPO-A に含まれる画像処理演算列の先頭と末尾はそれぞれ&3，&26 であり，この二つのノードが&27 を参照する *IOR* を保持する．末尾のノードが保持するラベル *IOR* は，画像処理

履歴が終端したことも表現している．画像処理手順を表す他のノード，例えば&12や&19からもラベル *IOR* が&27を参照するようにした場合，その履歴が終端したときの画像処理名を求めることができなくなるため，他のノードは *IOR* を保持できない．ただし，先頭のノードが保持するラベル *IOR* は，*IOR* の値を直接参照するために必要である．

context 最初の版として用いた画像データに関する情報．どのような画像を対象とした画像処理履歴であるかを判別するために用いられる．コンテキストは，画像処理履歴を作成したときに用いられた画像データオブジェクト[3),9),15)]の属性値として保持されている．図3は単純なコンテキストの例であり，*context* は撮影日時，撮影場所と撮影したセンサのバンド番号から構成されている．

category 画像処理履歴の大まかな分類を示す属性．どのような目的を持った画像処理履歴であるかを判別するために用いられる．カテゴリは，画像処理履歴を作成したユーザによって与えられる．この例では，HIPO-Aのカテゴリは属性値として土地被覆区分を表す値classificationを保持している．

どの画像処理演算が実行されるかは，画像処理履歴毎に決定される．したがって，図3に示すような構造は，画像処理履歴毎に全く違ったものとなる．このように画像処理履歴は不規則なため，あらかじめスキーマを定義することができない．このため，画像処理履歴の表現には半構造データモデルが適している．

画像処理履歴を表現するためにはラベル名の定義の他に，上記のラベルが保持する属性も定義する必要もある．この定義の一覧を**表1**に示す．この表では，c，aはそれぞれcomplex型，atomic型を意味し，atomic型の場合は，そのオブジェクトが保持する属性の型を示している．complex型の場合は，そのオブジェクトが属性として保持するラベルの集合を示している．また，記号?，*，＋はそれぞれ任意選択のオプション，0個以上，1個以上を意味する．以上のラベル名の定義と各オブジェクトが保持する属性の定義によって，画像処理履歴を表すグラフの構造が表現される．

以上で説明した画像処理履歴は，単一枚の画像を処理したときに生じる版を管理するためのGIOによって管理されるものである．しかし，リモートセンシングでのマルチバンド画像や医療画像でのCT断面画像の集合のように同時に複数枚の画像を扱う必要もある．本論文では詳細な説明は省略するが，複数枚の画像を組合せて利用するために，拡張汎画像オブジェクト（Extended Generic Image Object，EGIO）および複合汎画像

表1 オブジェクトの属性
Table 1 Attributes of objects

オブジェクト	型	属性
HDB	c	*input_1,...,input_n*
input_n	c	{*root*}+
root	c	*category*, {*context*}?, IOR, {*ip_⟨...⟩*}+, {*component*}*
ip_⟨...⟩	c	{*parameters*}?, {*value*}?, {*ip_⟨...⟩*}*,{IOR}?,
value	a	**string** | **int** | ...
parameters	c	{⟨パラメータ名⟩}+
⟨パラメータ名⟩	a	*value* と同上
IOR	a	**string**
context	c	アプリケーション毎に定義
category	a	**string**
component	c	*root* と同上

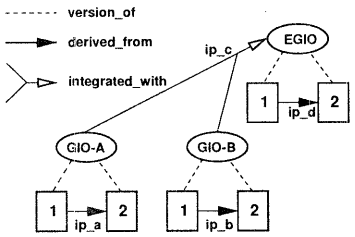

図4 EGIOの例
Fig. 4 An example of EGIOs

オブジェクト（Composite Generic Image Object，CGIO）を定義している[9)]．

図4にEGIOによる版管理の例を示す．この例では，個々の画像データはそれぞれGIO-A，GIO-Bによって管理され，さらにGIO-AとGIO-Bの結果を画像処理演算ip_cで統合するためにEGIOが用いられている．EGIOは，さらにその統合した結果を画像処理演算ip_dを用いて加工している．

図4をOEMで表現した場合，**図5**に示すようになる．GIO-A，GIO-BおよびEGIOは，それぞれHIPO-A，HIPO-B，HIPO-Cに変換される．点線で囲まれた部分がそれぞれ画像処理履歴HIPO-A，HIPO-BおよびHIPO-Cに対応する．HIPO-A，HIPO-Bはそれぞれ GIOによって管理された画像処理履歴であるから，入力として用いる画像は各1枚ずつ&4と&9である．したがって，&4，&9は，入力画像が1枚の画像処理履歴の集合を表すノード&2から *root* によって参照される．

一方，HIPO-Cはそれら二つの画像処理履歴の結果を画像処理演算ip_cで統合している．すなわち，HIPO-Cの入力として用いられる画像は，&4と&9の2枚である．HIPO-Cそのものが管理する画像処理履歴の先頭は，図4から分かるようにip_dの処理をする前

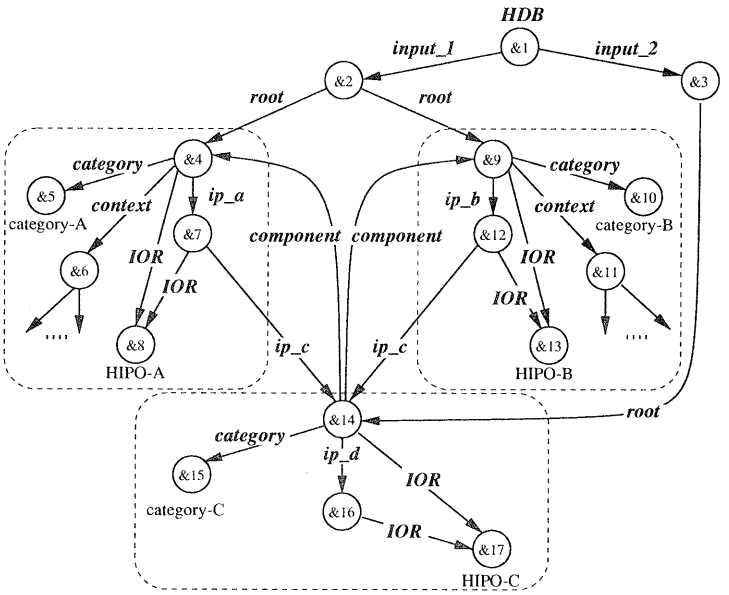

図 5 EGIO の OEM による表現
Fig. 5 A representation of EGIOs in OEM

の&14 となる．したがって，&14 は入力画像が 2 枚の画像処理履歴の集合を表すノード&3 から *root* によって参照される．さらに，ラベル ip_c によって二つの画像処理履歴を画像処理演算 ip_c で統合したことが表現される．また，HIPO-C が HIPO-A，HIPO-B の結果を用いていることを表現するため，EGIO の管理する画像処理履歴に対して次のラベルを定義する．

component EGIO の前処理を構成する GIO または EGIO の先頭を参照する．

この例では，ラベル *component* が，HIPO-C の先頭&14 から&4，&9 をそれぞれ参照している．

また，2 枚の入力画像それぞれがコンテキストを持つため，ラベル *context* はそれぞれ&4，&9 が保持する必要がある．さらに，category は HIPO-A，HIPO-B，HIPO-C それぞれが別の画像処理履歴のカテゴリに分類されるため，ラベル *category* を各 HIPO が保持する必要がある．したがって，*category* は，&4，&9 および&14 に保持される．

以上で説明した画像処理履歴を表現するグラフ構造は，半構造データモデルだけでなくオブジェクト指向データモデルでも実現が可能である．実際，Lore のシステムは O_2 システム上で実現されており，Lore の問合せ言語 Lorel は OQL の拡張として実装されている[2]．このため，図 3に示すような画像処理履歴のグラフ構造は，オブジェクトとして表現されている．ただし半構造データの不規則性を扱うため，これらのオブジェクトは参照先を示すラベルとその参照先の oid の対を属性として保持する汎用的なクラスによって表現される必要がある．例えば，図 3のオブジェクト&1 とオブジェクト&2 は，それぞれ違う型のオブジェクトを表しているが，そのオブジェクトの属すクラスは同一の汎用的なクラスになっている．前述したように画像処理履歴は不規則な構造を持つため，あらかじめスキーマを定義することは困難である．したがって，従来のオブジェクト指向データモデルのようにそれぞれのデータ構造に対するクラスを定義するのでは，画像処理履歴の半構造データとしての性質を扱うことはできない．

画像処理履歴を半構造データとして扱う必然性の一例として，図 3の画像処理演算 ip_ColorToGray の次に画像処理演算 ip_ThresholdSelection を実行することがそれらの画像処理演算を実行する時点まで決まらないということを挙げることができる．

我々もオブジェクトデータベース上に半構造データベースシステムを実装することが可能であるが，これには多大な労力を要する．我々の目的は半構造データベースシステムの実現ではなく画像処理履歴の検索システムの実現であることから，現時点で利用可能な半構造データベースシステム Lore を利用することにした．

5. 画像処理履歴の問合せ

Lore の問合せ言語である Lorel[2] による画像処理履歴の問合せ例を図 6 に示すデータベースを用いて説明す

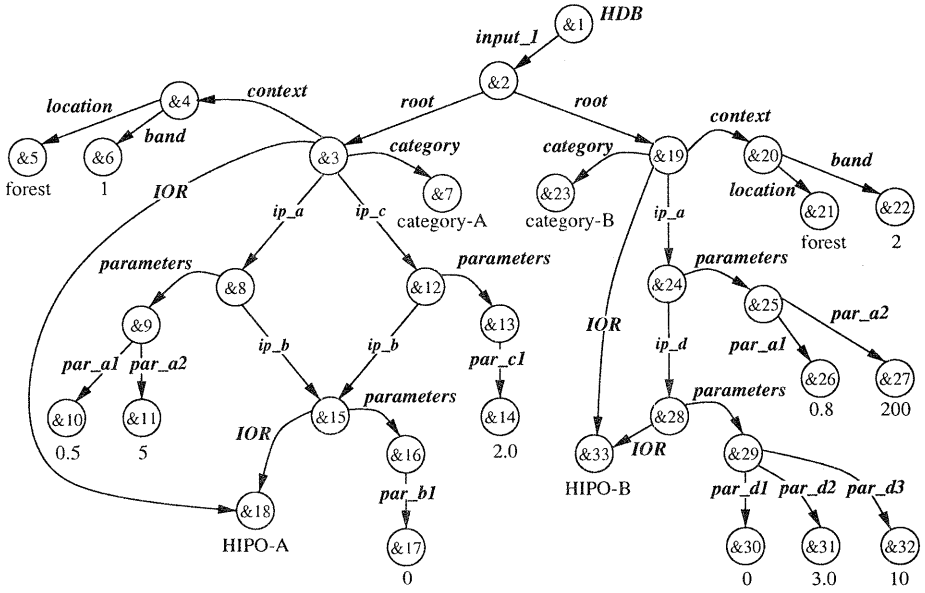

図 6 画像処理履歴のサンプル DB
Fig. 6 A sample DB for histories of image processing

る．これらの問合せによって画像処理履歴の検索がどのように実現されるかを示す．

5.1 画像処理演算の実行順序を指定した問合せ

まず，単純なパス表現を用いた画像処理履歴の問合せ例を以下に示す．パス表現は，$l_1, ..., l_n$ がラベルであり，Z がオブジェクト名またはオブジェクトを示している変数であるとき，列 $Z.l_1....l_n$ として表現される[2]．

[問合せ 1]

```
Q: select N
   from HDB.input_1.root.ip_a.ip_d.IOR N
A: IOR HIPO-B
```

問合せ 1は，from 節で root から ip_a，ip_d の順に画像処理を実行した処理履歴を指定している．したがって，この問合せの結果 HIPO-B が返される．このように，画像処理の実行順序そのものをパス表現で指定することが可能である．

[問合せ 2]

```
Q: select N
   from HDB.input_1.root R, R.IOR N
   where R.ip_a.ip_b = R.ip_c.ip_b
A: IOR HIPO-A
```

問合せ 2では，where 節で二つのパスが合流するノードが同じである条件を指定することによって HIPO-A のような画像処理履歴を求めることができる．

上記の二つの問合せ例は，画像処理履歴に含まれる画像処理演算の実行順序を正確に指定している．しかし，例えば，二つの画像処理演算のうちどちらを用いれば良いかが分からないときや，最初に実行する画像処理演算は分かっているがその後処理として必要な画像処理演算を知らないとき，画像処理履歴に含まれる正確な画像処理演算の実行順序を指定することはできない．このような場合，正規表現とワイルドカードを用いることによって，画像処理履歴の一部分だけを指定することができる．画像処理履歴の問合せはより強力な表現力を持つことができる．まず，正規表現を用いた問い合わせ例として以下のものを示す．

[問合せ 3]

```
Q: select N
   from HDB.input_1.root.
        ip_a(.ip_b|.ip_d).IOR N
A: IOR HIPO-A
   IOR HIPO-B
```

from 節で用いられている正規表現(.ip_b|.ip_d) によって，ip_b または ip_d を含んだパスが指定される．したがって，この問合せは最初に ip_a を実行し，その後 ip_b または ip_d を実行した画像処理履歴を求める．

[問合せ 4]

```
Q: select N
   from HDB.input_1.root(.ip_a)?.ip_d.IOR N
A: IOR HIPO-B
```

正規表現(.ip_a)?は，ip_a が 0 回または 1 回出現することを表している．したがって，この問合せでは ip_a，ip_d の順または ip_d だけを実行した画像処理履歴を求める．

全てのラベルを知らない場合や正確な順番を知らないといった場合がある．このため，ワイルドカードの概念を持つことも重要である．ワイルドカードは，次の二つが定義されている．%は0以上の任意の長さの文字列に，#は0以上の任意の長さのパスに一致するワイルドカードである．

ワイルドカードを利用することで，任意の長さの画像処理演算列を指定することができる．例えば，以下のようにfrom節でワイルドカード#を用いてパスを指定した場合，最後に画像処理演算ip_dを実行した画像処理履歴を求めることができる．

[問合せ 5]

```
Q: select N
   from HDB.input_1.root.#.ip_d.IOR N
A: IOR HIPO-B
```

しかし，この問合せ内で用いたワイルドカード#は画像処理名の他にparametersなどの画像処理名とは無関係なラベルとも一致してしまう問題が生じる．パス表現を用いて画像処理履歴を指定する場合，ワイルドカードを用いて履歴のある一部分の知識だけを指定することが多い．そこで，画像処理演算名には接頭語としてip_を付加させている．接頭語を用いることで，以下のように正確に画像処理名だけと一致するパスを指定することが可能になる．

[問合せ 6]

```
Q: select N
   from HDB.input_1.root(.ip_%)*.ip_d.IOR N
A: IOR HIPO-B
```

上記のfrom節は問合せ5の場合の#を(.ip_%)*に置き換えている．*は0回以上の繰り返しを表す正規表現であるから，この表現はrootとip_dの間に存在する画像処理演算名のパスと一致する．このような問合せの表現は，最後に実行する画像処理演算は分かっているが，その前にどのような画像処理演算を実行すればよいかが分からない場合に有効である．同様に，ある画像処理演算で開始する画像処理履歴やある画像処理演算を含んだ画像処理履歴も指定することもできる．以上のように，ワイルドカードと正規表現を用いることによって画像処理履歴に含まれる画像処理演算の実行順序の一部分だけを指定した問合せが可能となった．

5.2 画像処理履歴の構造に関する問合せ

これまでに示した問合わせ例は全て画像処理演算の実行順序に関するものであった．さらに，画像処理演算に用いたパラメータの値に関する問合せや画像処理履歴の構造そのものに関する問合せを記述することもできる．例えば，以下の問合せは，最初に実行したip_aのいずれかの引数に5.0を指定したという条件を与えている．

[問合せ 7]

```
Q: select N
   from HDB.input_1.root.ip_a R,
        R(.ip_%)*.IOR N
   where R.parameters.% = 5.0
A: IOR HIPO-A
```

次に，ワイルドカードを用いた画像処理履歴の構造に関する問合せを示す．以下の問合せは，rootとIORの間のパスを求めるものである．

[問合せ 8]

```
Q: select distinct pathof(P)
   from HDB.input_1.root(.ip_%)+@P.IOR
A: path ip_a ip_b
   path ip_c ip_b
   path ip_a ip_d
```

from節で1回以上の出現を表す正規表現+を用いて，rootからIORの間に存在する任意の長さの画像処理演算名の列にパス変数Pを割り当てている．pathof関数はパス変数に割当てられたパスを返すため，この問合せの結果は上記の集合となる．また，パス変数を以下のように用いてある特定のラベルを調べることもできる．

[問合せ 9]

```
Q: select distinct pathof(L)
   from HDB.input_1.root(.ip_%)*.
        parameters.%@L X
   where X = 10
A: path par_d3
```

この問合せでは，パス変数Lはワイルドカード%に対して割り当てられている．すなわち，Lは，長さ1のパスに対するパス変数であるから，そのラベルを表す変数であると考えることができる．したがって，この問合せは値が10であるパラメータに関するラベルを求めることを意味しており，結果として&29から&32を指しているエッジのラベルpar_d3を返す．

同様に，以下のような問合せで画像処理演算名のラベルを調べることによって，ある画像処理演算の次に実行された画像処理演算の種類を問合せることもできる．

[問合せ 10]

```
Q: select distinct pathof(L)
   from HDB.input_1.root(.ip_%)*.
        ip_a(.ip_%)@L
A: path ip_b
   path ip_d
```

この問合せでは，ip_aの次に行われたことのある画像処理演算名としてip_bとip_dを返す．したがって，あ

る画像処理演算の後処理や前処理として用いられたことのある画像処理演算を調べることが可能である．

以上のように画像処理履歴の構造を検索することによって，ある画像データがどのような画像処理演算を組合せて導出されたかを知ることができる．これは，他のユーザによって作成された画像処理履歴を理解することを容易にする．

5.3 カテゴリおよびコンテキストを用いた問合せ

画像処理のカテゴリに関する問合せは以下のように記述することができる．

[問合せ 11]

```
Q: select N
   from HDB.input_1.root R, R.IOR N
   where R.category = "classification"
```

同様にコンテキストに関する問合せの条件は，以下のように記述できる．この問合せでは，1999年に撮影した森林の画像に対する画像処理履歴を求める．

[問合せ 12]

```
Q: select N
   from HDB.input_1.root R, R.IOR N
   where R.context.location = "forest"
     and R.context.date.year=1999
```

カテゴリとコンテキストの概念を導入することによって，ユーザが必要とする画像処理履歴を絞り込むことができる．例えば，EGIOが管理する複雑な画像処理履歴を対象とした問合せを以下に挙げる．

[問合せ 13]

```
Q: select distinct N
   from HDB.input_2.root R, R.IOR N,
        R.componet C1, R.component C2
   where C1 <> C2
     and C1.context.band = 1
     and C1.context.location = "forest"
     and C2.context.band = 2
     and C2.context.location = "forest"
     and R.category = "classification"
```

この問合せで求められる画像処理履歴は，森林を撮影したリモートセンシング画像のバンド1およびバンド2の画像に前処理をそれぞれ行いそれらの結果から土地区分利用を計算するものである．このようにカテゴリとコンテキストの情報と画像処理演算の順序に関する条件を組合せることで，複雑な画像処理履歴を検索することが可能である．

以上のように，Lorelを用いることで画像処理履歴の強力な問合せが提供される．特に，正規表現やワイルドカードによる問合せを用いることで，画像処理履歴に関する部分的な知識だけでも画像処理履歴を検索することが可能となる．また，カテゴリやコンテキストを組合せることによって，より詳細な問合せを記述できる．

6. 関連研究

従来の半構造データに関する研究は汎用的なデータモデルの設計に重点を置いたものが多かったため，実際にそのデータモデルをどのようなアプリケーションで使うかが不明確であった[16]．具体的な応用例としては，遺伝子データ[17]，WWW[4),10)]および構造化文書[19]などを対象とした研究が行われてきている．我々の研究では，画像処理履歴を半構造データモデルによって表現することで，画像処理履歴の検索が実現できることを明らかにした．特に，画像処理履歴の検索では，画像処理演算の実行順序の条件を指定するためワイルドカードと正規表現が非常に有効であることを示した．正規表現を用いた検索方法は，従来よりグラフに基づくデータモデルにおいて提案されてきた[6),7),14)]．しかし，画像処理履歴は，不規則な構造を持つためあらかじめスキーマを定義することが困難であるため，グラフデータモデルでは対応できず，半構造データモデルによって表現する必要がある．

知識ベースに格納された画像処理演算に関するルールに基づいて画像処理演算の実行プランを推論機構が作成するアプローチが提案されている．Thonnat等は，ユーザがgoalと入力画像を指定することによって画像処理の実行プランを作成する手法を提案している[18]．また，Clouard等が提案する手法では，ユーザが入力画像に対してtask(goalに相当)，constraints，contextの三つを指定することで実行プランが作成される[5]．しかし，知識ベースに格納するルールを記述するには画像処理に関する高度な知識を必要とする．単に画像処理に関する知識だけでなく，そのルールを記述する能力も要求されるため，正確なルールを記述することは困難である．

一方，我々のアプローチでは，専門家が実際に一連の画像処理演算を実行した結果を画像処理履歴として保存し，その履歴を非専門家が再利用する．専門家が日常の業務で画像処理演算を実行していくことによって，画像処理履歴が蓄積されていく．この操作は通常の画像処理演算を行うだけであるから，ルールを記述するよりも簡単に実行できる．

また，我々と同様に履歴の管理に焦点を当てた研究にGaeaがある[8]．Gaeaは，GISにおける衛星画像データの導出履歴を管理することを目的としている．このシステムでは，オブジェクト指向データモデルに

concept, process および task の三つの概念を導入している. concept は Landsat, Land_Use 等の入出力データの種類を表し, process はある concept から新たな concept を作成する一連の手続きを表している. 我々の研究では, 入力データに対する concept がコンテキスト, 出力データに対する concept がカテゴリ, process が画像処理履歴にそれぞれ対応している. また, task は, ある process を再利用し画像処理の導出履歴を再現するものである. task と同様に, 我々の研究でも画像処理履歴で表される導出履歴を再現することができる[15].

Gaea と我々の研究との重要な違いとして以下のことが挙げられる. Gaea では導出履歴を管理することによってデータがどのような手順で作成されたか知ることができるが, その手順に関する条件を指定した検索は考慮されていなかった. しかし, 我々の研究では半構造データモデルを用いて履歴を表現することによって, 画像処理履歴に含まれる画像処理演算の条件を指定した検索を実現している.

7. おわりに

画像処理履歴は不規則な構造を持つため, あらかじめ画像処理履歴に対するスキーマを定義することは困難である. そこで, この論文では, 画像処理履歴をどのように半構造データモデルを適用して表現できるかを議論した. そして, 画像処理履歴の検索方法を例を用いて示した. 半構造データベースで提案されている問合せ手法を画像処理履歴の検索に適用することによって, 画像処理履歴に対する柔軟な問合せが実現できることを示した. 本論文では, リモートセンシング画像を対象とした例を示したが, この手法は医療画像等の他分野の領域に対してもカテゴリおよびコンテキストを定義することで応用が可能である.

謝辞 本研究の一部は文部省科学研究費基盤研究 (C)(2) (課題番号 10680333), 文部省科学研究費特別研究員奨励費 (受付番号 3473) による.

参考文献

1) Abiteboul, S.: Querying Semi-Structured Data, *Int'l Conf. in Database Theory(ICDT)*, LNCS, Vol. 1186, Springer-Verlag, pp. 1–18 (1997).
2) Abiteboul, S., Quass, D., McHugh, J., Widom, J. and Wiener, J.: The Lorel Query Language for Semistructured Data, *International Journal on Digital Libraries*, Vol. 1, No. 1, pp. 68–88 (1997).
3) Aritsugi, M., Tabata, M., Fukatsu, H., Kanamori, Y. and Funyu, Y.: Manipulation of Image Objects and Their Versions under CORBA Environment, *Proc. 8th Int. Workshop on Database and Expert Systems Applications (DEXA 97)*, pp. 86–91 (1997).
4) Arocena, G. O. and Mendelzon, A.: WebOQL: Restructuring Documents, Databases, and Webs, *Proc. of IEEE ICDE*, pp. 24–33 (1998).
5) Clouard, R., Elmoataz, A., Porquet, C. and Revenu, M.: Borg: A Knowledge-Based System for Automatic Generation of Image Processing Programs, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.21, No.2, pp.128–144 (1999).
6) Consens, M. P. and Mendelzon, A. O.: GraphLog: a Visual Formalism for Real Life Recursion, *Proc. of 9th ACM PODS*, pp. 404–416 (1990).
7) Gyssens, M., Paredaens, J., Van den Bussche, J. and Van Gucht, D.: Graph-Oriented Object Database Model, *IEEE Trans. on Knowledge and Data Engineering*, Vol. 6, No. 4, pp. 572–586 (1994).
8) Hachem, N.I., Qiu, K., Gennert, M. and Ward, M.: Managing Derived Data in the Gaea Scientific DBMS, *Proc. of the 19th VLDB Conf.*, pp. 1–12 (1993).
9) 川島亨, 田幡勝, 金森吉成, 増永良文: 画像オブジェクトの版管理モデル, 電子情報通信学会論文誌, Vol. J79-D-I, No. 10, pp. 843–852 (1996).
10) Konopnicki, D. and Shmueli, O.: Information Gathering in the World-Wide Web: The W3QL Query Language and the W3QS system, *ACM Trans. on Database Systems(TODS)*, Vol. 23, No. 4, pp. 369–410 (1998).
11) McHugh, J., Abiteboul, S., Goldman, R., Quass, D. and Widom, J.: Lore: A Database Management System for Semistructured Data, *SIGMOD Record*, Vol. 26, No. 3, pp. 54–66 (1997).
12) Object Design Inc.: *ObjectStore C++ API Reference Release 5.0* (1997).
13) Object Management Group, Inc.: *The Common Object Request Broker: Architecture and Specification, Revision 2.2* (1998).
14) Paredaens, J., Peelman, P. and Tanca, L.: G-Log: A Graph-Based Query Language, *IEEE Trans. on Knowledge and Data Engineering*, Vol. 7, No. 3, pp. 436–453 (1995).
15) 田幡勝, 有次正義, 金森吉成: 分散環境における画像オブジェクトの版管理機構の実現, 情報処理学会論文誌：データベース, Vol. 40, No. SIG5(TOD2), pp. 79–90 (1999).

16) 田島敬史: 半構造データのためのデータモデルと操作言語, 情報処理学会論文誌: データベース, Vol. 40, No. SIG3(TOD1), pp. 152–170 (1999).
17) Thirrry-Mieg, J. and Durbin, R.: Syntactic Definitions for the ACeDB Database Manager, Technical Report CB2 2QH, MRC Lab. for Molecular Biology, Cambridge, UK (1992).
18) Thonnat, M., Moisan, S. and Crubézy, M.: Experience in Integrating Image Processing Programs, *Proc. of the 1st Int'l Conf. on Visual Systems* (1999).
19) Yamashita, R., Ito, T. and Yao, H.: ES-SQL: An Enhanced Semi-Structured Query Language for Composite Document Retrievals, *Proc. of the 16th Annual Int'l Conf. on Computer Documentation(SIGDOC'98)*, pp. 120–126 (1998).

(平成1999年 9 月20日受付)
(平成1999年11月15日採録)

(担当編集委員 宝珍 輝尚)

田幡 勝 (学生会員)

1995 年群馬大・工・情報工卒. 1997 年同大大学院博士前期課程 (情報工学専攻) 修了. 現在, 同大大学院博士後期課程 (電子情報工学専攻) 在学中. 1997 年日本学術振興会特別研究員. 電子情報通信学会会員.

有次 正義 (正会員)

1991 年九大・工・情報工卒. 1996 年同大大学院博士後期課程了. 1994–1995 文部省派遣留学生として McGill 大 (院) 在籍. 博士 (工学). 1996 年から群馬大・工・情報工助手. 並列分散データベース, マルチメディアデータベース等に興味を持つ. 電子情報通信学会会員.

金森 吉成 (正会員)

1969 年東北大大学院博士課程修了. 工学博士. 東北大電気通信研究所助手, 同助教授, 仙台電波高専教授を経て, 1990 年群馬大・工・情報工教授. オブジェクト指向データベース, マルチメディアデータベースに関する研究に従事. ACM, IEEE-CS, 電子情報通信学会各会員.