データベースシステム 102－3
（1995．3．14）

画像DBアプリケーション開発言語の実装

松田 宜之[†] 大津 浩二[†] 金森 吉成[†] 増永 良文[††] 脇山 俊一郎[†††]

[†] 群馬大学 [††] 図書館情報大学 [†††] 仙台電波高専

画像データベースのアプリケーション開発が容易にできるオブジェクト指向のスクリプト言語を実現する方法を述べる。この言語の特徴の1つは、画像処理機能をスクリプトの中に埋め込んでいることである。複雑な画像処理では、ある画像処理を適用した同一の画像オブジェクトを複数回使用する場合がある。このような場合、既存の画像オブジェクトを再利用すれば、全体の処理時間が短縮できる。そこで、スクリプトで書かれた画像処理部分から再利用できる画像オブジェクトを見つけ出し、処理時間短縮のための最適化が必要になる。本研究では、画像処理過程を版階層構造と捉えて、その構造を解析することにより言語記述を最適化する方法を提案する。

An Implementation of Application Development Language for Image Databases

Takayuki Matsuda[†] Kouji Ootsu[†] Yoshinari Kanamori[†]
Yoshifumi Masunaga[††] Shunichiro Wakiyama[†††]

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

This paper describes a method to implement an object-oriented script language, with which application developments for image databases are easy. A feature of the script is to embed the image processing functions in the language. Complex image processings use several times a same image object obtained from the results during the process. In this case, if the processings can reuse the image object existed already, it is possible to reduce the total computing time. Then, we propose an algorithm to find out the objects, which are possible to reuse from the parts of image processings written in the scripts. It is necessary to optimize the scripts to save the computing time. In this study, we see the complex image processings as a version hierarchy of objects. We describes a method to optimize the script by analyzing the structure of version hierarchy.

1 はじめに

著者等は、画像データベースのアプリケーションを容易に開発することができるオブジェクト指向のスクリプト言語の設計 [1] 及びオブジェクト指向の画像データベースシステムのアーキテクチャ [2] について研究してきた。この言語の特長の1つは画像処理機能をスクリプトの中に埋め込むことができる点にある。複雑な画像処理においては、ある画像処理を適用した同一の画像オブジェクトを複数回利用する場合が生じる。このようなとき、スクリプトで書いた画像処理をそのまま実行すると計算時間の多大な浪費となる。そこで、スクリプト言語の処理系を設計する際には、既存の画像オブジェクトを再利用して全体の処理時間を短縮するための最適化が重要になってくる。

本研究では、スクリプト言語の処理系における最適化方法を提案する。

2 画像 DB アプリケーション開発言語

画像 DB アプリケーション開発言語 [1] は、プログラミング言語、データベース、画像処理関数、ウィンドウシステムなどの高度な計算機環境を統合したオブジェクト指向のスクリプト言語である。これにより、アプリケーションを容易に記述することができる。

本言語は、以下の内容の記述ができるように設計されている。

- ハイパーテキスト構造を持つ GUI
- データベースに対する問合せ
- 複雑な画像処理過程

特に、複雑な画像処理過程に関しては、図 1に示すように 1 枚の画像に複数の画像処理関数を適用する場合 (図中の 2) や、複数の画像を重ね合わせて 1 枚の画像にする場合 (図中の 9) など様々な形態があるが、画像処理オブジェクト (メソッド) を明確に導入することによって複雑な画像処理過程の記述を可能にしている。

3 画像 DB アプリケーションプログラムの解析

画像 DB アプリケーション開発言語によって記述されたプログラム (以下、シナリオと呼ぶ) は、各種のオブジェクトの表記からなる。そして、実行時にはオブジェクト群によるメッセージパッシングのみで進行する。従って、シナリオの解析はオブジェクト群の生成であると考えることができる。シナリオの解析過程を図 2に示す。

シナリオはシナリオ解析システムによってシナリオ動作環境記述言語 (3.2節) に変換される。そのうち、画像処理に関する部分は汎画像オブジェクト生成システムによって版階層構造記述言語 (4.2.1節) に変換される。それぞれの中間言語に変換されたシナリオはオブジェクト生成システムによってオブジェクト群に変換される。

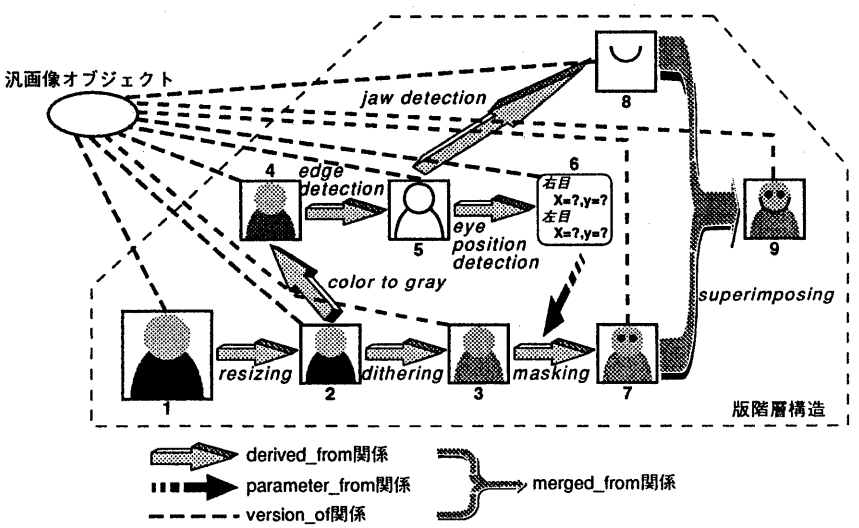

図 1: 画像オブジェクトの版階層構造 [2]

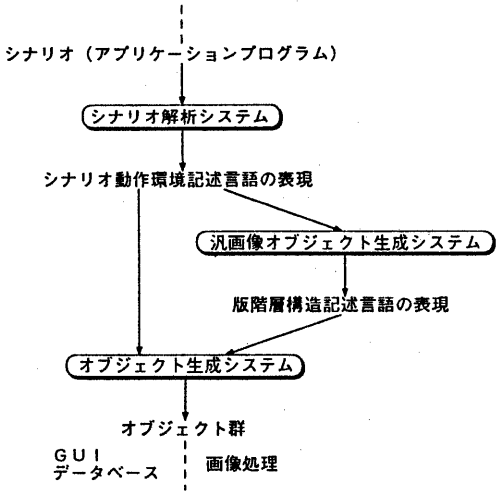

図 2: シナリオの解析過程

3.1 シナリオ

シナリオをオブジェクト指向分析すると、シナリオはシーンと呼ばれる画像データベースに格納されているオブジェクトを表示するためのウィンドウ、画像データベースに問合せを発するためのオブジェクト、そして複雑な画像処理過程を定義するためのオブジェクトから構成されている。また、シーンは個々のオブジェクトを表示するための基本オブジェクトとシナリオの動作を決定するためのリンクオブジェクトから構成されている。この様子を図3に示す。シナリオの実行はこの構造に基づいて生成されたオブジェクトのメッセージパッシングによって遂行される。

3.2 シナリオ動作環境記述言語

シナリオ動作環境記述言語は、シナリオを文法的・意味的に解析した途中結果を記述するための言語で、シナリオ内で利用されるオブジェクトや画像処理の様々な情報が種々の表によって表現されている。以下に、シナリオ動作環境記述言語を構成している表を列挙する。

- シーン表 (ST)
- 基本オブジェクト表 (BOT)
- グループ表 (GT)
- リンク表 (LT)
- メッセージ表 (MT)
- 問合せオブジェクト表 (QT)
- 画像処理オブジェクト表 (IPT)

3.3 シナリオの解析例

シナリオとその解析例を図4に示す。シナリオは医療症例データベースからユーザが与えた年齢の患者を検索するシナリオの一部を抜粋したものである。このシナリオがシナリオ解析システムに与えられると、図の下方に示した表群、即ちシナリオ動作環境記述言語に変換される。

4 画像処理に関するシナリオの最適化

図1のような複雑な画像処理を実行する場合は、単純な処理の場合とは比較にならない程膨大な時間が費やされてしまう。例えば、図1の2の画像は目的の画像 (図1の9) を得るために3回利用される。従って、全く同じ画像処理を3回も実行することは、無駄な処理時間を費やすことになる。そこで、このような時間的な冗

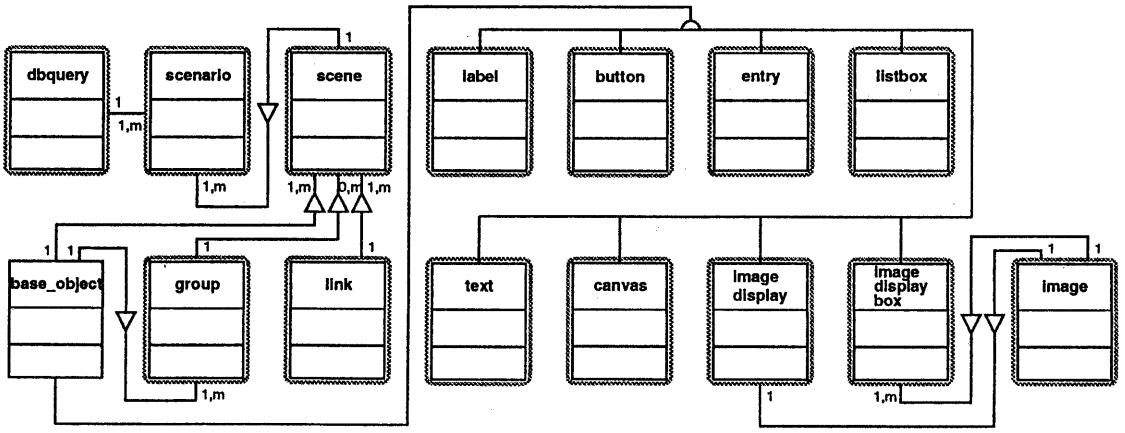

図 3: シナリオの構造 (Coad-Yourdon 法 [3] による記述)

長を避けるための最適化が必要となる。即ち、一度処理した結果を保存して再度それを利用することによる処理時間の最小化が必要である。本研究では、画像処理時間の最小化を目的としたシナリオの最適化手法を提案する。

4.1 画像オブジェクトの版管理

図1では、画像処理を適用する毎に新たな画像オブジェクトが生成される。このような画像オブジェクトを1つの版と捉え、それらを管理するための版管理モデルを提案した[2]。

このモデルでは、図1のような画像オブジェクトの階層構造を版階層構造 (Version Hierarchy: VH) とし、各画像オブジェクトの共通の属性と履歴構造を持つ抽象化したオブジェクトとして汎画像オブジェクト (Generic Image Object: GIO) を持つ。また、各版の間には意味的に異なる次の4つの関係がある。

1. 導出関係 (derived_from relationship)
2. パラメタ関係 (parameter_from relationship)
3. 融合関係 (merged_from relationship)
4. 版関係 (version_of relationship)

以下では、このモデルに基づくシナリオの最適化手法について述べる。

4.2 汎画像オブジェクトの生成法

画像処理時間を最小化するには以下の条件を満たす必要がある。

1. 利用可能な画像オブジェクトを再利用する。

また、一連の画像処理を実行して目標の画像を得るには次の3つの条件を満たさなければならない。

1. 画像処理を正確な順序で実行する (導出関係に相当する)。
2. 画像処理間の時間的順序関係を厳守する (パラメタ関係に相当する)。
3. 画像オブジェクト間の画像処理適用関係を厳守する (融合関係に相当する)。

これらの条件は版階層構造によって正確に表現できる。従って、シナリオの最適化とは以上の4つの条件を満足するような汎画像オブジェクトを生成することである。そこで、まず始めに版階層構造を記述するために版階層構造記述言語について述べる。次に、シナリオ動作環境記述言語を版階層構造記述言語に変換するために汎画像オブジェクト生成規則を提案する。

4.2.1 版階層構造記述言語

版階層構造の構成要素について考えてみると、

- **画像オブジェクトの版** (図1の1,2, …,9)
- **画像処理関数** (resizing, dithering 等)
- **版間の意味関係**

から成っていることが理解できる。従って、版階層構造を表現するにはそれらを記述できる記号系を与えればよい。この記号系を版階層構造記述言語と命名し、以下にその仕様を述べる。

画像オブジェクトの版

一般に、実行時には画像データベースから問合せ条件を満足する複数の画像オブジェクトが検索される。その結果、それぞれの画像オブジェクトについて、図1のような版階層構造が生成される。これは、同一の版階層構造を持つ画像オブジェクトが複数存在すると考えることができる。故に、画像オブジェクト及び版階層構造に識別子を付けて、管理、識別しなければならない。また同様に、1つの版階層構造に対応した汎画像オブジェクトが1つ生成されるので、汎画像オブジェクト識別子も必要となる。これらの記号からなる表VIO (図5の表(1)) が生成される。

- **汎画像オブジェクト識別子** (gioid1, gioid2, …)
- **版階層構造識別子** (vhid1, vhid2, …)
- **画像オブジェクト識別子** (ioid1, …)

画像処理関数

一般に、1つの画像処理過程は複数の画像処理関数を連続して適用させている。例えば、図1の2 → (*color_to_gray*) → 4 → (*edge_detection*) → 5という処理過程である。このような画像処理過程は通常、目的の画像を得るために複数回利用される場合がある。従って、この一連の画像処理過程に識別子を付けてそれを新たな画像処理関数名として定義し、以後はその識別子を利用する。これらの記号からなる表IPF (図5の表(2)) が生成される。

- **画像処理過程識別子** (ipid1, ipid2, …)
- **画像処理関数名** (resizing, dithering, …)

版間の意味関係

画像処理関数と意味関係との対応をあらかじめ定義することができる (例えば、resizing: 導出関係、superimposing: 融合関係等)。即ち、画像処理を適用した結果生じるオブジェクトの版は3つの意味関係のどれであるかを一意に決めることができる。これらの記号からなる表SBV (図5の表 (3)) が生成される。

- **版階層構造識別子**
- **画像処理過程識別子**
- **意味関係**

4.2.2 汎画像オブジェクト生成規則

汎画像オブジェクトは以下の手順で生成される。

(1) シナリオ動作環境記述言語によって記述された画像オブジェクト及び画像処理オブジェクト定義部分の全部を取り出す。
(2) (1) で取り出された記述と汎画像オブジェクト生成規則をパターンマッチングさせる。
(3) マッチングした規則の版階層構造と意味を版階層構造記述言語の各表に記入する。

これにより、画像処理時間を最小化する汎画像オブジェクトを生成することができる。以下に、汎画像オブジェクト生成規則を示す。

ただし、大文字は表、小文字はその属性を表す。

定義 1 (シナリオ動作環境記述言語の表現) シナリオ動作環境記述言語の表現 S とは、シナリオに出現するオブジェクトの名前、属性値及びシナリオを動作させるためのメッセージを表として記述したもので、以下の構造を持っている。

$$S = ST \cup BOT \cup GT \cup LT \cup MT \cup QT \cup IPT$$

ただし、

ST: ST (*sname, boname, gname, lname*)
BOT: BOT (*cname, boname,* $boatt_1, boatt_2, \cdots, boatt_n$)
GT: GT (*gname, cname, boname*)
LT: LT (*lname, eoname, event, msgid*)
MT: MT (*msgid, oname, method, msg*)
QT: QT (*qoname, db, dbms, query*)
IPT: IPT (*iponame, ipfname*)

とする。

定義 2 (版階層構造記述言語の表現) 版階層構造記述言語の表現 V とは、版階層構造の構成要素を表として記述したもので、以下の構造を持っている。

$$V = VIO \cup IPF \cup SBV$$

ただし、

VIO: VIO (*gioid, vhid, ioid*)
IPF: IPF (*ipid, ipfname*)
SBV: SBV (*vhid, vhid & ipid, semantics*)

とする。

定義 3 (汎画像オブジェクト生成規則) 汎画像オブジェクト生成規則 R は、以下の構造を持っている。

R: $S \longrightarrow V$
R = { *gio, vh, ip, sr* }

規則 1 意味関係は次の規則に従って生成される (シナリオの解析とは独立に予め定義しておく)。

$$sr : ipfname \longrightarrow semantics$$

例 1 *sr: color_to_gray* $\longrightarrow$ *derived_from*
sr: masking<> $\longrightarrow$ *parameter_from*
sr: superimposing $\longrightarrow$ *merged_from*

規則 2 画像処理過程は次の規則に従って生成される (図5の表 (b) から表 (2) を生成する)。

ip: $(ipfname_1, ipfname_2, \cdots, ipfname_n) \longrightarrow ipid\#$

ただし、$(ipfname_1, ipfname_2, \cdots, ipfname_n)$ はこの順序で画像処理関数を適用するという意味を持つ。また、$\# = 1, 2, \cdots, m$ は *ipid* の識別子番号で *ipid* の生成順に付けるものとする。

例 2 *ip:* (*dithering*) $\longrightarrow$ *ipid2*
ip: (*color_to_gray, edge_detection*) $\longrightarrow$ *ipid3*

規則 3 版階層構造は次の規則に従って生成される (図5の表 (3) 表 (2) から表 (3) を生成する)。

$vh : ipid1 \longrightarrow vhid1$
$vh : (vhid_1, vhid_2, \cdots, vhid_n, ipid\#) \longrightarrow vhid\#$

ただし、$(vhid_1, vhid_2, \cdots, vhid_n, ipid\#)$ は $vhid_1$, $vhid_2$, $\cdots$, $vhid_n$ に $ipid\#$ を適用するという意味を持つが、$ipid\#$ の画像処理関数が持つ3つの意味関係の違いによって画像処理の

適用順序が次のように適宜変化する。

1. 導出関係 ($n=1$ のときのみ)
   $(vhid_1, ipid\#)$ は $vhid_1$ に $ipid\#$ を適用して $vhid\#$ を生成するという意味を持つ。
2. パラメタ関係
   $(vhid_1, vhid_2, \cdots, vhid_n, ipid\#)$ は $vhid_1$ に $vhid_2, \cdots, vhid_n$ をパラメタとして与え、$ipid\#$ を適用して $vhid\#$ を生成するという意味を持つ。
3. 融合関係
   $(vhid_1, vhid_2, \cdots, vhid_n, ipid\#)$ は $vhid_1$ に $vhid_2, \cdots, vhid_n$ をその順番で $ipid\#$ に関して適用して $vhid\#$ を生成するという意味を持つ。

また、$\# = 1, 2, \cdots, m$ は $ipid$, $vhid$ の識別子番号で $ipid$, $vhid$ の生成順に付けるものとする。

例 3 *vh: ipid1* $\longrightarrow$ *vhid1*
vh: (vhid2, vhid4, ipid5) $\longrightarrow$ *vhid5*

規則 4 汎画像オブジェクトは次の規則に従って生成される (図5の表 (3) 表 (a) から表 (1) を生成する)。

$$gio : (ioid, vhid) \longrightarrow gioid\#$$

ただし、$(ioid, vhid)$ は $ioid$ に $vhid$ を適用するという意味を持つ。また、$\# = 1, 2, \cdots, m$ は $gioid$ の識別子番号で $gioid$ の生成順に付けるものとする

例 4 *gio: (ioid1, vhid5)* $\longrightarrow$ *gioid1*
gio: (ioid1, vhid7) $\longrightarrow$ *gioid3*

これらの規則は規則 1、規則 2、規則 3、規則 4の順に適用される。また、新たな規則が必要になった場合には、以上の 4 つの規則に従って追加される。

4.3 汎画像オブジェクトの生成例

汎画像オブジェクトの生成例を図5に示す。適用される画像処理は図1で表現したものである。シナリオは 1 つの画像オブジェクトを扱っている。このシナリオがシナリオ解析システムに与えられると、図5の右上に示した表群に変換される。さらに、それが汎画像オブジェクト生成システムに与えられると、図5の下方に示した表群、即ち版階層構造記述言語に変換される。例えば、図5の表 (1) にある gioid3 に着目すると、画像オブジェクト ioid1 に版階層構造 vhid7 を適用して生成された汎画像オブジェクトであることがわかる。次に、その vhid7 に着目すると図5の表 (3) を見てわかるように版階層構造 vhid5 と vhid6 に画像処理 ipid7 を適用した版階層構造であることがわかる。以下同様に、版階層構造の生成過程をたどっていくと、一番最初に生成される vhid1 については vhid2 と vhid3 の 2 つの版階層構造の生成元になっている。これは vhid1 によって生成された画像オブジェクトが vhid2 と vhid3 によって再利用されていることを表している。この再利用が処理過程の分岐点を通過する全ての画像オブジェクトに対して実行されるため、目的の画像オブジェクトに対応する vhid7 が生成される時には、画像処理時間が最小となる汎画像オブジェクト gioid3 が生成されている。即ち、シナリオの最適化がなされている。

5 おわりに

本稿では、画像 DB アプリケーション開発言語によって記述したシナリオの解析結果を記述するための言語として、シナリオ動作環境記述言語と版階層記述言語を提案した。また、画像処理時間の最小化を目的とした最適化手法として、汎画像オブジェクト生成規則を提案した。

謝辞

本稿をまとめるにあたり、汎画像オブジェクトの生成法について御助言いただいた群馬大学大学院の川島享氏に感謝致します。

参考文献

[1] 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. 情報処理学会研究報告, 100-7. 情報処理学会データベースシステム研究会, October 1994.

[2] 川島享, 金森吉成, 増永良文, 脇山俊一郎. 画像オブジェクトサーバにおける版管理モデル. 情報処理学会研究報告, 101-17. 情報処理学会データベースシステム研究会, January 1995.

[3] P.Coad, E.Yourdon. *Object-Oriented Analysis*. Prentice-Hall, 第 2 版, 1991.

シナリオ

```
scene sceneA -component {                                    # シーンの定義をする。
  label labelA -x 5 -y 5 -string Age:                        # ラベルを作成する。
  entry entryA -x 10 -y 5 -length 5                          # エントリを作成する。
  button buttonA -x 5 -y 15 -string Retrieve                 # ボタンを作成する。
  button buttonB -x 15 -y 15 -string Quit                    # ボタンを作成する。
  link linkA -from buttonA -condition buttonpress1           # 検索、シーンの表示 / 非表示
             -action {{set_patient -retrieve [entryA -string]} # を行なうリンク
                      {sceneA -unmap} {sceneB -map}}         # オブジェクトを作成する。
  link linkB -from buttonB -condition buttonpress1           # シナリオを終了するリンクオ
             -action {scenarioA -end}                        # ブジェクトを作成する。
}
dbquery set_patient -specify {ONTOS patientDB    # ユーザが指定した年齢の患者オブ
                              {select patient    # ジェクトを patientDB から検索す
                               from patient      # る問合せを指定する。
                               where age = $1}}
```

シーン表 **ST**

シーン名	基本オブジェクト名	グループ名	リンク名
sceneA	labelA, entryA, buttonA, buttonB	—	linkA, linkB

基本オブジェクト表 **BOT**

クラス名	基本オブジェクト名	X	Y	···	string	···
label	sceneA.labelA	5	5	···	Age:	···
entry	sceneA.entryA	10	5	···	—	···
button	sceneA.buttonA	5	15	···	Retrieve	···
button	sceneA.buttonB	15	15	···	Quit	···

X, Y: X 軸、Y 軸方向に関するシーン上のオブジェクトの位置
string: オブジェクト上に表示される文字列

リンク表 **LT**

リンク名	イベントオブジェクト名	イベント	メッセージ識別子
sceneA.linkA	sceneA.buttonA	buttonpress1	m1
sceneA.linkB	sceneA.buttonB	buttonpress1	m2

メッセージ表 **MT**

メッセージ識別子	オブジェクト名	メソッド名	パラメータ
m1	set_patient	retrieve	m3
m1	sceneA	unmap	—
m1	sceneB	map	—
m2	scenarioA	end	—
m3	sceneA.entryA	string	—

問合せオブジェクト表 **QT**

オブジェクト名	画像データベース名	DBMS	問合せ指定
set_patient	patientDB	ONTOS	select patient from patient where age = $1

図 4: シナリオの解析例 (シナリオ動作環境記述言語による表現)

シナリオ

```
define ip1  resizing(50,100)
define ip2  dithering
define ip3  color_to_gray \
            edge_detection
define ip4  eye_position_detection
define ip5  jaw_detection
define ip6  $ip1 $ip2
define ip7  $ip1 $ip3 $ip4
define ip8  masking<$ip6,$ip7>
define ip9  $ip1 $ip3 $ip5
define ip10 superimposing<$ip8,$ip9>
declare fa_image

imagedisplay i1 -get fa_image \
                -execute $ip8
imagedisplay i2 -get fa_image \
                -execute $ip9
imagedisplay i3 -get fa_image \
                -execute $ip10
```

シナリオ動作環境記述言語の表現

表 (a) 基本オブジェクト表 BOT

···	基本オブジェクト名	···	get	execute	···
···	sceneX.i1	···	fa_image	$ip8	···
···	sceneX.i2	···	fa_image	$ip9	···
···	sceneX.i3	···	fa_image	$ip10	···

表 (b) 画像処理オブジェクト表 IPT

画像処理オブジェクト名	画像処理関数名
ip1	resizing(50,100)
ip2	dithering
ip3	color_to_gray edge_detection
ip4	eye_position_detection
ip5	jaw_detection
ip6	$ip1 $ip2
ip7	$ip1 $ip3 $ip4
ip8	masking<$ip6,$ip7>
ip9	$ip1 $ip3 $ip5
ip10	superimposing<$ip8,$ip9>

版階層構造記述言語の表現

表 (1) 画像オブジェクトの版表 VIO

汎画像オブジェクト識別子	版階層構造識別子	画像オブジェクト識別子
gioid1	vhid5	ioid1
gioid2	vhid6	ioid1
gioid3	vhid7	ioid1

表 (2) 画像処理関数表 IPF

画像処理過程識別子	画像処理関数名
ipid1	resizing(50,100)
ipid2	dithering
ipid3	color_to_gray edge_detection
ipid4	eye_position_detection
ipid5	masking<>
ipid6	jaw_detection
ipid7	superimposing

表 (3) 版間の意味関係表 SBV

版階層構造識別子	画像処理過程識別子及び版階層構造識別子	意味関係
vhid1	ipid1	derived_from
vhid2	vhid1 ipid2	derived_from
vhid3	vhid1 ipid3	derived_from
vhid4	vhid3 ipid4	derived_from
vhid5	vhid2 vhid4 ipid5	parameter_from
vhid6	vhid3 ipid6	derived_from
vhid7	vhid5 vhid6 ipid7	merged_from

図 5: 汎画像オブジェクトの生成例