NAIST-IS-MT9751032

修士論文

オフィスロボットにおける
インタラクティブな属性学習を用いた
物体の位置情報管理

片山 顕正

1999 年 2 月 12 日

奈良先端科学技術大学院大学
情報科学研究科 情報処理学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
修士 (工学) 授与の要件として提出した修士論文である。

片山 顕正

指導教官： 西田 豊明 教授
横矢 直和 教授
武田 英明 助教授

オフィスロボットにおける
インタラクティブな属性学習を用いた
物体の位置情報管理*

片山 顕正

内容梗概

本研究の目的は、人間の活動支援としてオフィス内に散在し、絶えずその位置が変化し得るオブジェクトを自律的に認識・記憶して人間の要求するオブジェクトの提示を行い、取りに行くロボットの構築をすることにある。そのためのロボットシステムは、環境とインタラクションができること、動的で柔軟な対象の表現ができること、人間とインタラクションを行えることが必要である。

本研究で実現したロボットシステムは環境とのインタラクションとしての巡回行動、巡回中に撮影された画像からのオブジェクト認識、オブジェクトに対しての名前付けやオブジェクトの取得といったユーザとのインタラクションからなる。ユーザーは WWW ブラウザから取ってきて欲しいオブジェクト名もしくはオブジェクトの属性情報をロボットに伝える。ロボットはデータベースからオブジェクトを検索しその位置情報をもとにオフィス内を移動し、要求されたオブジェクトを取ってくる。

オブジェクトの対象表現は人間が日常に用いている見た目の形、色、模様といった属性を用いて行う。オブジェクトの認識においてはロボットが撮影した机等の画像からオブジェクトを切り出し、これらの属性を決定する。形と模様はパターンの類似度を用いて分類する。この方法による実験的結果として 27%の正答率で

*奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 修士論文, NAIST-IS-MT9751032, 1999 年 2 月 12 日.

同一のオブジェクトを識別することができる。このシステムを実装することにより、ロボットは人間とのインタラクションによって得た知識を用いてオフィス内に散在するオブジェクトを自律的に管理することができるようになった。

キーワード

人間、認識モデル、インタラクション、属性、学習

Management of Object Locations with Interactive Attribute Learning System for Office Robot.*

Akimasa Katayama

Abstract

The purpose of this study is to build a robot system that can recognize and manage objects in office. Such a robot system needs, interaction with environments, flexible and dynamic object representation, and interaction with human. Functions of our robot system are (i) rounding the office to look for objects, (ii) object recognition, and (iii) naming objects and ordering to take objects by human. User can tell the system either an object name or some attributes. Then robot can retrieve an object to bring it from its recorded position in the database. Object representation is based on attributes like shape, color, and pattern that are easy understandable by human. In object recognition process, the robot can cut object in pictures that are capture in rounding, and determine their objects. Shape and pattern are determined to compose stored ones by calculating similarity. This method resulted in 27% in precision in a experimental test. The implemantal system enables autonomous managements of objects in office with knowledge. That can be acquired in human-robot interaction.

Keywords:

human being, recognition model, interaction, attribute, learning

*Master's Thesis, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT9751032, February 12, 1999.

目 次

図 目 次

表 目 次

第 1 章

はじめに

近年、さまざまな場所で人間の存在する環境の中で人間の支援を行なうロボットの研究がなされている。

麻生らによる事情通ロボット [1, 2, 3] は「オフィス等で自律的に情報収集し、人間に情報サービスを行う移動ロボットエージェント」として建物内部を巡回し、情報収集・案内等を行なう。また Molella らによるスミソニアン博物館における館内案内ロボット [4] は館内の人の密集度合いを見ながらその都度適切な案内ルートを決定する。これらのロボットに見られるようにロボットが人間のいる環境の中で作業を行なうためには、環境の変化を動的に認識・管理する機能が必要とされる。環境の変化と言ってもさまざまではあるが、ここではオフィス内に散在するオブジェクトの位置情報の管理についての研究を行った。

オフィス内において本やノートといった小物 (オブジェクト) は人間によって絶えずその位置が変化し得る。ロボットが自動的にオフィス内を巡回する時にこれらのオブジェクトを自律的に認識して位置情報とともに記憶しておくことができれば、人間の活動の支援に有効であると考えられる。

昨年までの研究成果として、ロボットは自動的に研究室内を巡回してスナップショットを撮影して周り、人間によって意味付けしてもらうことでオブジェクトの位置データを記憶し、人間の要求するオブジェクトを取ってくることが可能になった。そこで今回はこの研究を元に、ロボットがオフィス内を巡回する時にオブジェクトを自律的に認識できるシステムの構築を行った。

オフィス環境内に散在するオブジェクトを認識するモデルとしてオブジェクトの属性による分類手法を用いた。

これは人間がオブジェクトを認識する際に用いられる形や大きさといった抽象概念による属性認識モデルであり、ロボットはスナップショットから抽出したオ

ブジェクトデータを自らが持つ属性情報とマッチングを行わせることでオブジェクトを認識し、位置、時間情報を管理する。

オブジェクト画像のマッチングには画像のテンプレートマッチング手法を用いた。テンプレートマッチングの手法についてはアルゴリズムコンテスト [5] も開かれるなど盛んな研究分野である。本研究では実環境の動的認識に主眼を置き、なるべく単純な処理で高速に環境の認識が出来るよう決定木を用いたデータ分類に利用した。

また、未知のオブジェクトに関しては人とのインタラクションを通して属性情報の学習を行うことで後の認識に活かせるようにした。このようにして得られたオブジェクトの位置情報からロボットは人間の要求する物を持ってくることができるようになった。

1.1 研究の目的と方針

当研究室における昨年までの成果でロボットは人の背景知識を利用して人とコミュニケーションを取ることができる。そこではロボットはあらかじめカメラから取り込んだスナップショットに対してその中にあるオブジェクトの情報を人の手によって意味付け・データベース化を行われており、自らがオブジェクトの情報を抽出して背景知識を構築することは出来なかった。これはロボットに求められる自律性が満たされていないとも言える。

そこで、本研究ではロボットよる環境の自動認識と、対話による行動生成という視点でより自律性の高いロボットの構築を目指す。

具体的には属性認識モデルという人に近い認識モデルを使用したシステムを用いて、カメラから得られた環境画像に存在するオブジェクトの１つ１つを認識し、それがいつ・どこにあったかという情報をユーザに提供するシステムおよび、インターフェースとして発話機能によるコミュニケーションシステムを構築する。

人は視覚で物を認識する時に色、形、大きさなどの情報を用いてそれが何であるのか大まかな推定を行い、さらに触覚や嗅覚などの五感を用いてオブジェクトの確定を行う。属性認識とは、人がものを認識する過程で行われる、形、色、大きさ等の抽象概念の認識を指す。本研究では中でも、形、色について焦点を当てて話を進める。

本論文ではこの属性認識システムをロボットに実装し、ロボットの環境把握とオブジェクトの位置情報管理について評価を行う。

1.2 本論文の構成

2 章では、過去における研究について述べ、3 章では、本件に用いるシステムの説明を、4 章 5 章において、システムのロボットへの実装と検証、考察を述べる。また、付録に本件で用いたプログラムのアルゴリズム説明とハードウェアの仕様書を付す。

第 2 章

人とロボットとの共存

本研究は昨年までに当研究室で行われていた研究を引き継ぎ「人間と共存するロボット」を構築することを目的とする。そこで、今一度オフィス環境ロボットが克服して行かなければならない問題を昨年までの成果とともに確認していく。

2.1 オフィスロボットと産業ロボットとの違い

「オフィスロボット」とはその名のとおり「人間のために構築された環境下で動作するロボット」のことであり、ロボットのためにその作業環境が整えられている産業ロボットとはずいぶん立場が違う。そのためロボットにとって次のような点が問題となる。

- 作業を行なう環境は人間のために作られている
- 作業内容は決められたステップで表現できない
- 人間とのインタラクションを必要とする
- ロボットは自律的に移動できなければならない
- 人間の干渉を考慮に入れる必要がある

これらは全て「人間のために」という大前提によって起こる問題があり、オフィスロボットは産業用ロボットとはまるで違う観点から話をはじめなければならない。まず、人間と共存するロボットを実現する際、以下の点を考慮しなければならない。

- 不特定多数の人間と共存する
- 環境の変化に対応するための柔軟な知識が必要

昨年度の研究では、これらの点を踏まえ次のような研究がなされた。

2.2 人との知識共有により物の位置を管理するロボット

昨年までの成果として、弱い知識構造 (後述) を用いて人間と背景知識を共有し、室内の備品など絶えずその所在が変化し得るものの管理をするロボットの構築が行われた [6, 7]。

これは、人の持つオブジェクトに関するあいまいな背景知識を弱い知識構造を用いて記述し、ロボットが撮影して回ったスナップショットとの関連付けをすることでオブジェクトの位置を特定し人に提示させるシステムで、これによりロボットは人とのインタラクションをすることによって人の持つ弱い背景知識を利用することが出来るようになった (図 2.1)。

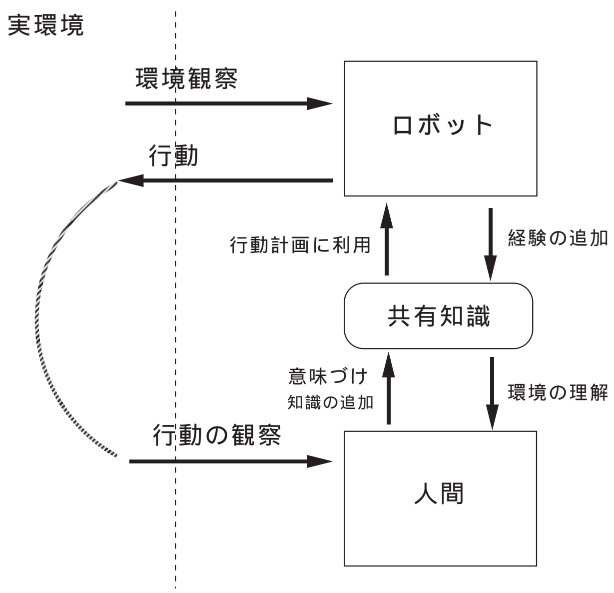


図 2.1 人間の背景知識を持つロボット概念図 ([6] より)

2.2.1 弱い知識構造

弱い知識構造は弱連想構造とも呼ばれ、述語論理における意味的関係の定義や構造化文書における章や節のような物理的階層構造などを用いず、概念同士の関係の有無のみを用いた知識表現である。

連想構造は key と value という 2 種類のユニットによって表現される。key と value の間には「key によって value が想起される」という規則があり、この間の連想関係は厳密なものでなくてよく、主観的な定義をも許容し、表現の意味解釈を書き手と受け手にいっさい任せるという非常に緩やかな関連構造を形成している。この研究では当研究室で開発が進められている CoMeMo ワークベンチ [8] を用いて背景知識の記述が行われている。

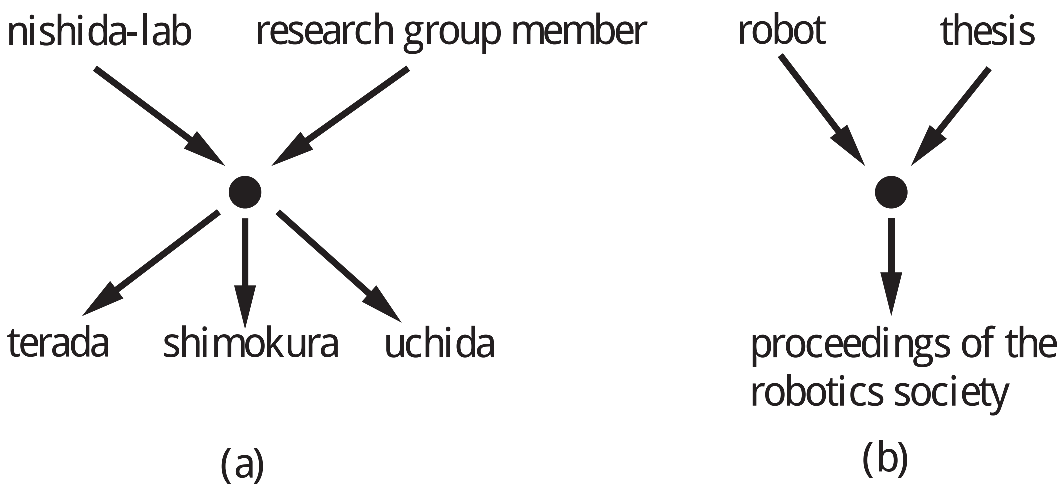


図 2.2 連想構造の例 ([6] より)

図 2.2(a) は、key「nishida-lab」「research groupmember」が与えられると value「terada」「shimokura」「uchida」が想起されることを示す。図 2.2(b) は key「robot」「thesis」が与えられると value 「proceedings of the robotics society」が想起されることを示している。

2.2.2 弱い知識構造を用いた検索

実際のデータは Lisp の s 式で記述される。図 2.2(b) を実際のデータ形式で書くと図 2.3のようになる。

(unit-1 (type concept)(label “robot”))
(unit-2 (type concept)(label “thesis”))
(unit-3 (type concept)(label “proceedings of robotics society”))
(unit-4 (type association)(label “a4”)(keys unit-1 unit-2)(values unit-3))

図 2.3 CoMeMo のデータ形式 ([6] より)

まず連想構造の記述の方法に、ある程度のルールを決めておく。人間が検索の対象にしそうなもの、例えば本などを連想構造の中に書いた場合、そのものと「収

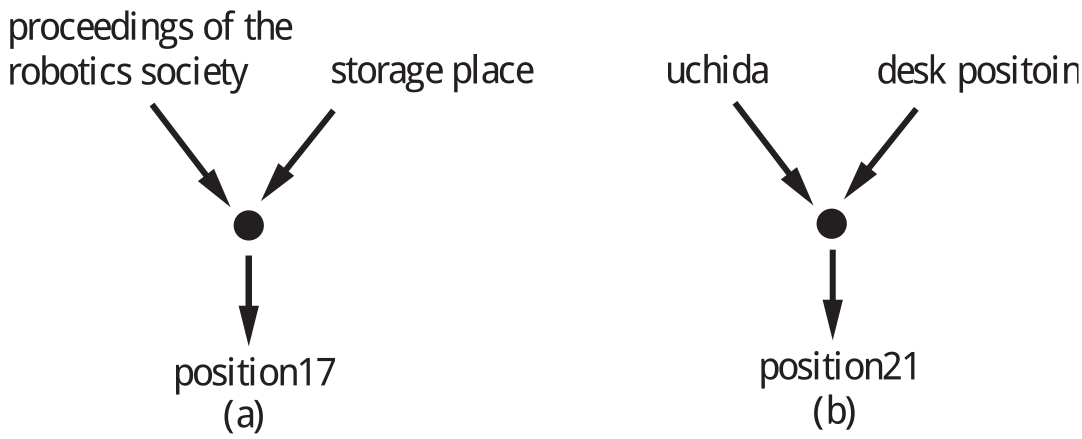


図 2.4 ロボットのための連想構造 ([6] より)

納場所 (storage position) 」を key、そのものが置かれている場所を value として、連想構造を記述することにする。図 2.4(a) にその例を示す。

また、人間の名前を記述した場合にも、その人が普段使っている机の場所を連想構造に書き入れる。この場合、key はその人の名前と「机の場所 (desk positon) 」、value は机のある場所とする。図 2.4(b) にその例を示す。

さらに、連想構造を検索に用いるためにオブジェクトの名前、存在する場所や、それをよく保持している人の名前などの情報が付け足される (図 2.5)。人の名前は、オブジェクトをそれに関連する人から探す場合に用いられる。図 2.3のデータを例にすると、ユニット間の連結に IS-A 情報によって情報を適宜追加して行き、ロボットがオブジェクトの場所を見つけるために連想構造を検索する場合、IS-A 構造を持つユニットのクラスを頼りにして検索を行なわせる。

(unit-1 (type concept)(label “robot”))
(unit-2 (type concept)(label “thesis”))
(unit-3 (type concept)(label “proceedings of robotics society”)(is-a missing-item))
(unit-4 (type association)(label “a4”)(keys unit-1 unit-2)(values unit-3))

図 2.5 特別な情報を付加した CoMeMo のデータ ([6] より)

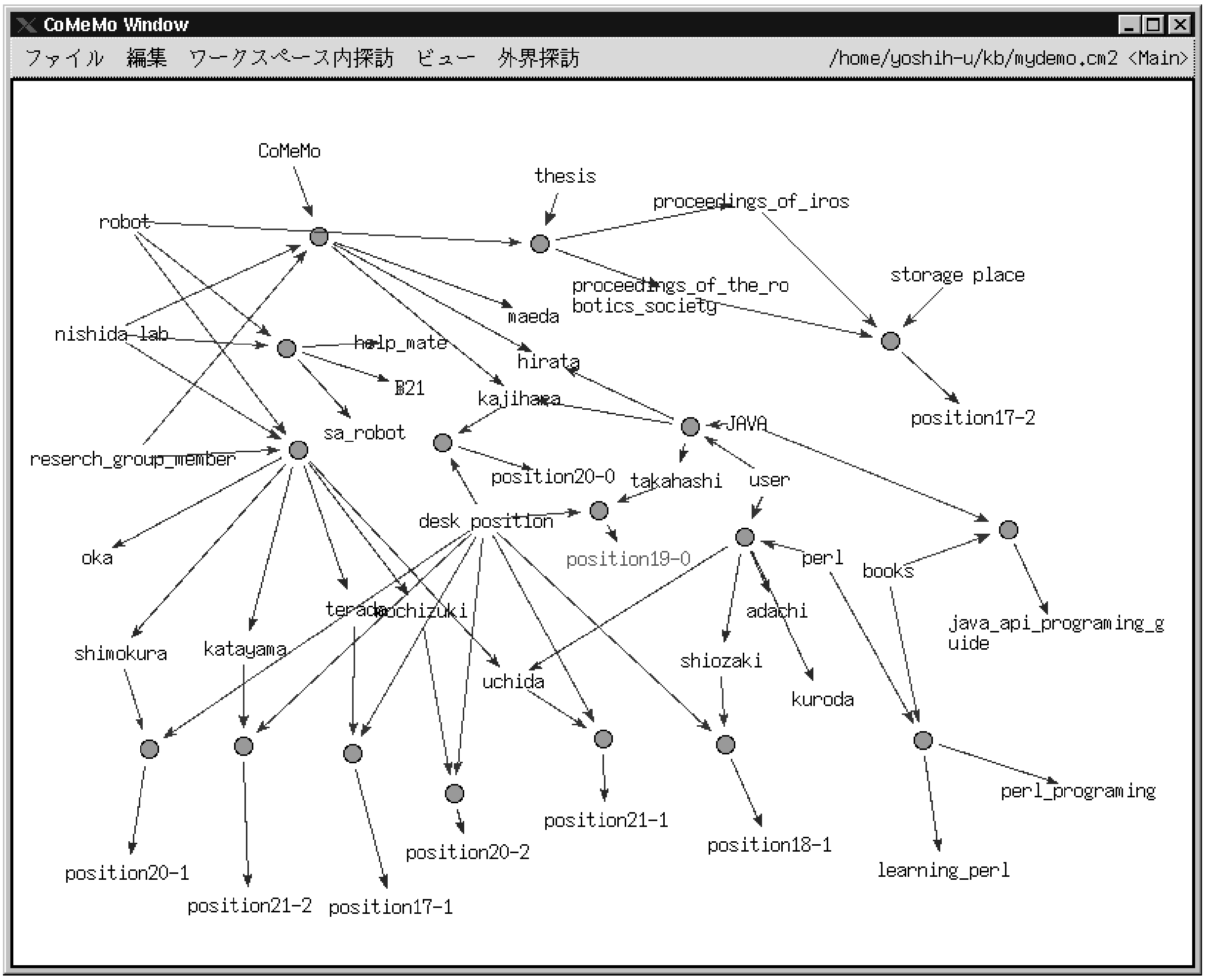


図 2.6 CoMeMo ワークベンチを用いた弱い知識構造の記述例 ([6] より)

2.3 実環境で人間の作業を手助けするロボット

以上のようにして「オフィスのような実環境において、人間から頼まれた「もの」を持ってくることができるロボット」の構築は行われ、有効性が確かめられた。この研究結果からも、ロボットが人と共存していくためには、ロボットに人の認識を理解させるシステムがより必要とされると考えられる。そこで、今回はオブジェクトが持つ抽象概念を理解するロボットの構築を考えることにする。

第 3 章

オブジェクトの位置を把握するロボット

本研究では、カメラから送られてくる画像からロボットが自ら属性情報を抽出し、自らの知識と照らし合わせてオブジェクトに対して意味付けを行い、データベース化することを目的とする。ここでは人がオブジェクトを認識する際のモデルを模倣したシステムを構築することにした。この章ではこの認識システムの動作に付いて実際の実行プロセスに沿って説明を行なう。

3.1 人間のオブジェクト認識モデル

それでは人はいったいどのような方法で、オブジェクトの認識を行うのであろうか？

人がオブジェクトを認識する際にはいくつかのアプローチがある。それら全ては五感によるもので、たとえば臭いであるとか、手触りであるとか音であったり等である。ここでは画像処理に関して人の認識モデルを構築することを考えるため、人の視覚によるオブジェクトの認識に用いられる問題点を書き出してみる。

- 文字は書いてないか？
- 大きさはどのくらいか？
- どのような形をしているか？
- どんな色をしているか？
- 材質は何でできているか？

もし、オブジェクト上に文字によるタイトルが書いてあれば話は早い。能力があるならそれらを読み取ってしまえば良い。

しかしロボットのカメラによる文字認識精度が余り高くなく、またそういった情報そのものが曖昧な場合なら、次に重要度が高いのは、「色」、「形」、「大きさ」といった属性情報であると考えられる。人はまずオブジェクトの大まかな形を見、色や材質からそれが何であるか判断を下すであろう [9] 。このシステムをロボットに模倣させるために、本論文では属性学習という手法を提案する。属性学習とは、オブジェクトの持つ属性情報を学習していく方法で、これによって作成された属性テンプレートとのマッチング (属性認識と呼ぶ) を行うことでオブジェクトの認識と分類を行うことができる。

大まかな手順としては以下の通りである。

1. どんな形をしているのか？
2. 何色で構成されているのか？
3. どのような模様 (色分布) なのか？

これらをロボットカメラによるスナップショットに対して用いて属性認識を行うシステムを構築する。次に実際のシステムについて述べる。

3.2 属性学習システム

本研究では自動認識モデルとして属性学習・認識システムの提案を行う。属性学習システムとはオブジェクト認識の際に用いる “大きい・小さい”、“丸い・四角い”、“青い・白い” などといった抽象的な概念をロボットに学習させるシステムでこれによって得た知識をもとにオブジェクトの属性認識を行うことでオブジェクト情報を効率よく分類し、検索にかかる時間もデータベースへの全検索にくらべて高速になっている (後述)。
まずは要の学習システムである属性学習システムについての概要を述べる。画像処理にかかるアルゴリズムの詳細は、付録として付した。(付録 A)

3.3 カメラ画像からのオブジェクト情報の抽出

まず、ロボットカメラで撮影された画像に対しロボットはオブジェクト単位での画像の切り分けを試みる。その後に抽象的な意味合いをもつテンプレート (プリミティブテンプレート) を参照させ、色、形、大きさなどの意味情報を加えて行く。さらに、保持しているデータの中にそのオブジェクトと近似しているものがあれば、その情報も付け加えて行く。

3.3.1 画像からのオブジェクト領域の切り出し

カメラからの画像はフルカラーであるため処理がしにくい。そこで、まずは幾つかの色領域を用いてこれを数個の色情報でカテゴライズする事を考える。本研究では、RGB と HSV による色情報を処理 ([10] および付録 A.1) することで、カメラから送られる画像中の近似色をまとめ、さらにそれらをあらかじめ定められた有限個数 (ここでは 8 つ) の色領域に分けることで、画像をカテゴライズし・単純化することにした。

本研究ではオブジェクト領域の切り出しには幾つかのルール (仮定) を与えて話を進める。

- 画像のカテゴライズとオブジェクトの切り出しに関するルール
  - カメラの視野に収まる程度の小物であること
  - 背景色はグレーとし、床、壁、机、天井を示す
  - オブジェクトは原画像中に重ならず存在すること
    (背景色中に色が隣り合って存在する領域はひとつのオブジェクトを構成しているとみなす)

3.3.2 システムの構成

まず、システムの概要を図示する (図 3.1)。

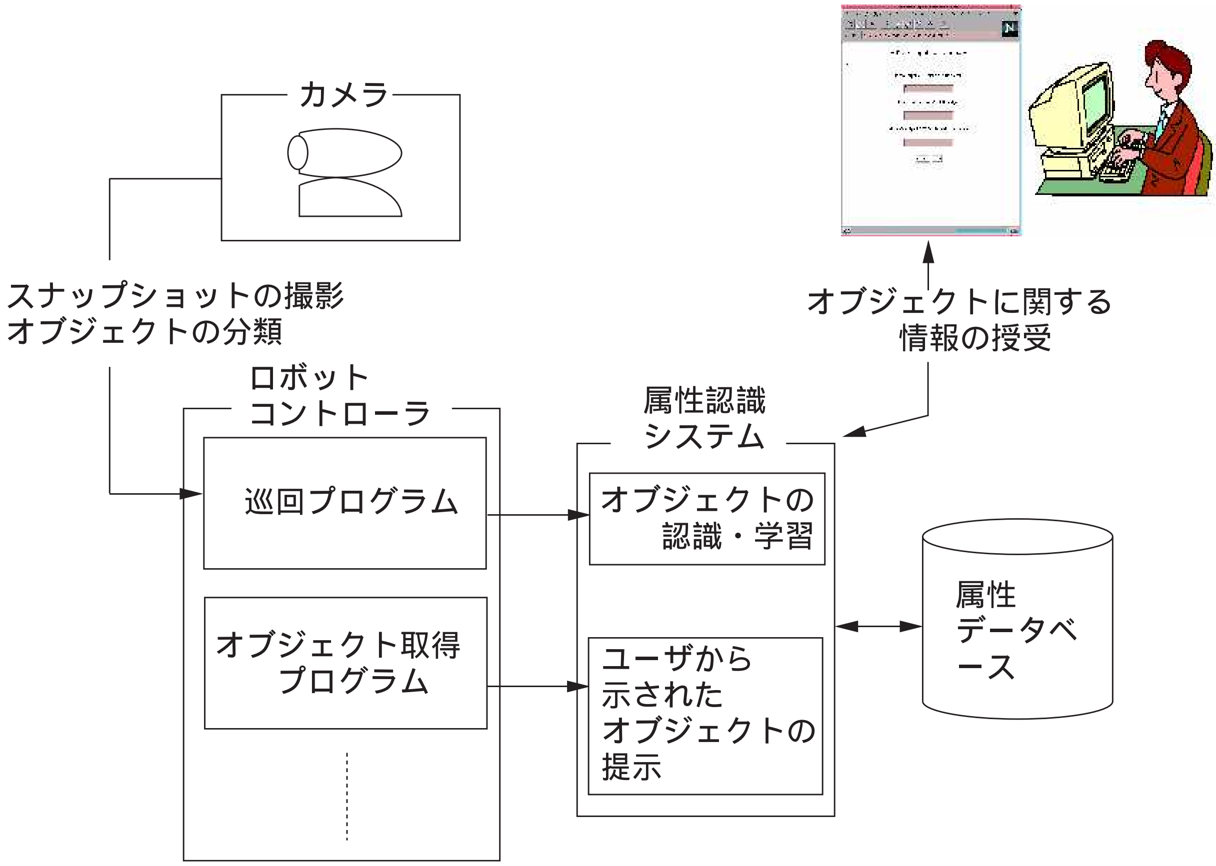

カメラ
スナップショットの撮影
オブジェクトの分類
ロボット
コントローラ
巡回プログラム
オブジェクト取得
プログラム
属性認識
システム
オブジェクトの
認識・学習
ユーザから
示された
オブジェクトの
提示
属性
データベ
ース
オブジェクトに関する
情報の授受

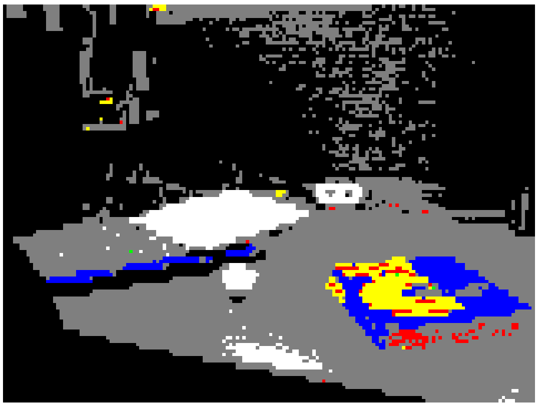
図 3.1 システムの構成

3.3.3 カメラ画像からオブジェクトの領域を切り分ける

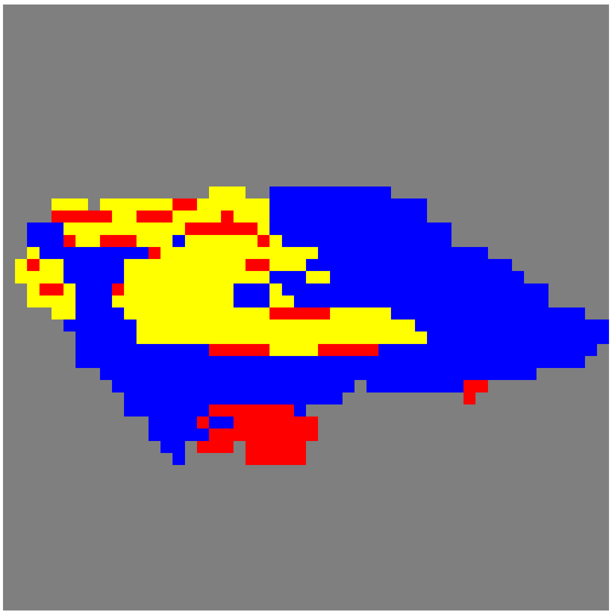
図 3.2.(a) にロボットカメラから得られた 1 枚のスナップショットを用意する。色領域分割を当てはめてこれをカテゴライズする (図 3.2.b)。さらに、これをもとに背景からのオブジェクト領域の抽出を行い (付録 A.2)、形状データを保存する (図 3.2.c)。このファイルの中にはまだ認識が完了していないことを示す "unknown" というラベルとともに時間・撮影場所・原画像の情報等、この時点で分かっている全ての情報 (今回は見つけた場所、時間、切り出したスナップショット名の情報。必要に応じて拡張可能) を書き加えておく (図 3.3)。また、この切り分けられたオブジェクト画像単位をフレームと呼ぶことにする。

本研究では画像入出力形式を xpm フォーマット [11] とした。これは、ファイルフォーマットがアスキー形式である為に人の手による編集が用意であること、そしてファイルに対して自由に情報の付与が行えるというメリットがあるためである。

(a) 原画像



(b) カテゴライズされた原画像



(c) 領域の切り出し

図 3.2 画像からオブジェクト情報の切りだし

```
/* XPM */
static char * unknown_xpm[] = {
"160 120 8 1",
"0      c #7F7F7F7F7F7F",
"1      c #FFFFFFFFFFFF",
"2      c #000000000000",
"3      c #FFFF00000000",
"4      c #FFFFFFFF0000",
"5      c #0000FFFF0000",
"6      c #00000000FFFF",
"7      c #FFFF0000FFFF",
"0000000000000000000 …
            :
            :
         〈中略〉
            :
            :
0"};
date: 02:08:03:49:36
pixelorg: 0208034936
snap point: 21
```

図 3.3 フレームデータの概形

3.4 ロボットによるオブジェクトの推論と学習

ここでは、実際にロボットがどのようにオブジェクトに関する知識の獲得をし、また、環境情報を獲得するのか、流れに沿って話を進める。

3.4.1 属性学習フローチャート

最初に、作業の流れを簡単にフローチャートとして図示する (図 3.4)。

3.4.2 オブジェクトの認識

まずフレームをマッチングテストにかけて基本的な形状情報を導き出し、これまでの経験からそれが何であるかを推測する。たとえばオブジェクトの大きさが中くらいで、四角形、青い色をしているという情報からここでは [本、マウスパッド] が導き出されるという具合である。分からなければ、形状・色の情報のみを記録しておく。このフレームと合わせて場所、時間、色、形など、ここまでの経過で分かっている情報をまとめて記録しておき、後に人との対話の中でカテゴライズして行く際に利用する (図 3.5)。

- プリミティブテンプレートとのマッチング

  ロボットは形状に関する幾つかのテンプレートをあらかじめ持っている。それは、四角であったり丸であったりといった世界を構成する基本的 (プリミティブ) な形状を表している (図 3.6)。切り出されてきたフレームはこのプリミティブテンプレートとのマッチングで整合率を計算する。マッチングは背景ピクセル値を "0"、オブジェクト領域ピクセル値を "1" とし、テンプレートと合致するピクセルとしないピクセルの出現度合いから整合率を求める。具体的には向きの変化を考慮してオブジェクトの影を回転させ、テンプレートと "and"、"xor" にかけ、ピクセルの $and = 1$ となった場合点数を与え、$xor = 1$ となった場合には減点すると言う手法を取る (付録 A.2)。これによって求められる整合率がもっとも大きく、かつあらかじめ定められる閾値より大きければそのテンプレートをオブジェクトの基本形状として認識する。もし閾値以下の整合率なら、該当無しとしてユーザに新規テンプ

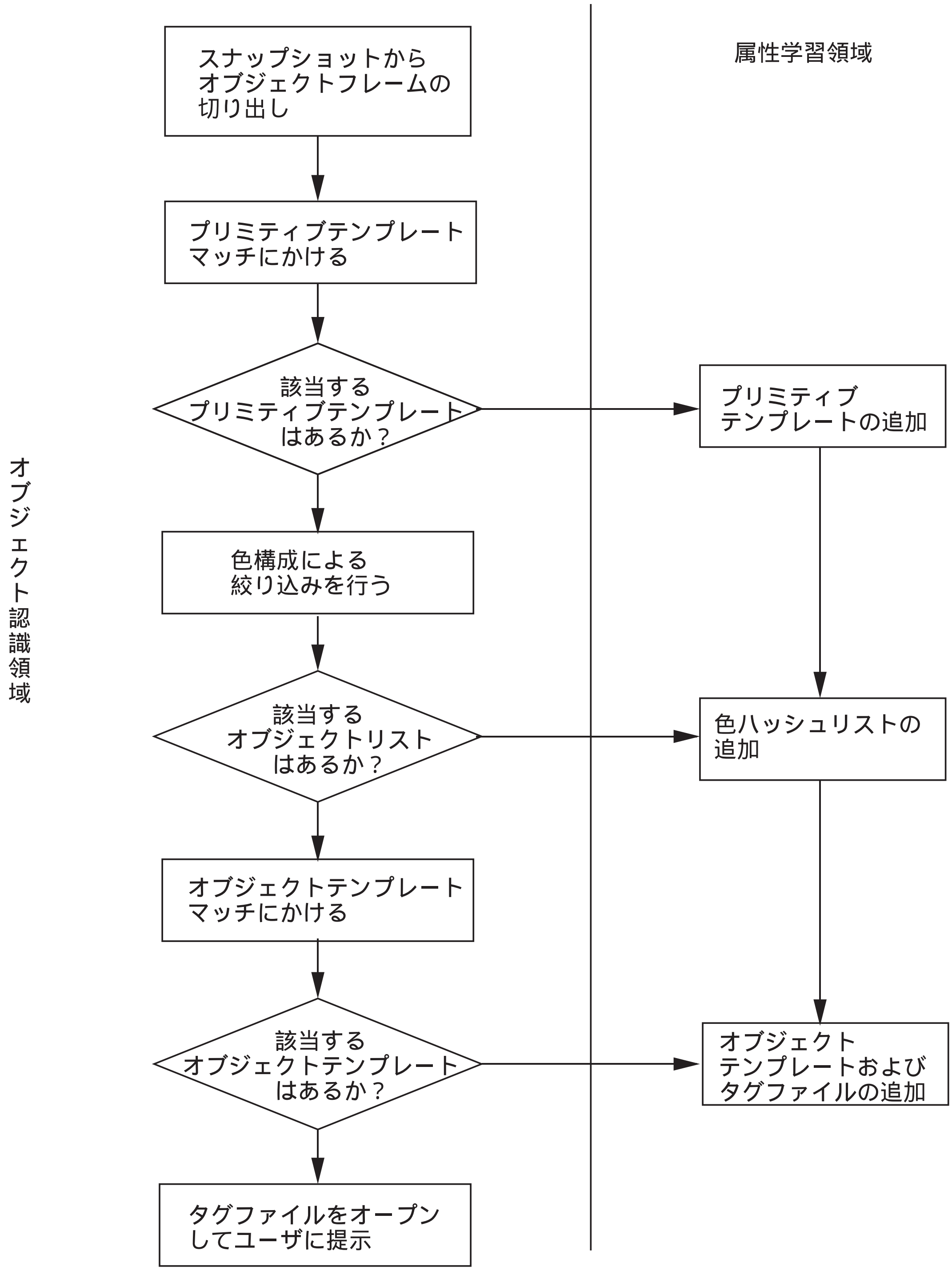


図 3.4 属性学習フローチャート

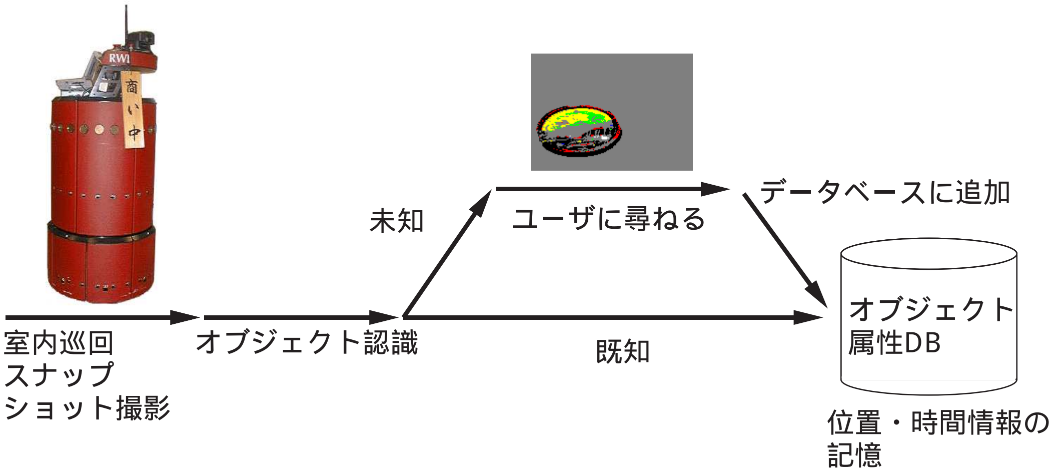


図 3.5 オブジェクトの認識

レートの情報提供を求める。例えば実際に図 3.2に対してオブジェクトに対しテンプレートマッチングを行った所、長方形として認識される (図 3.7)。

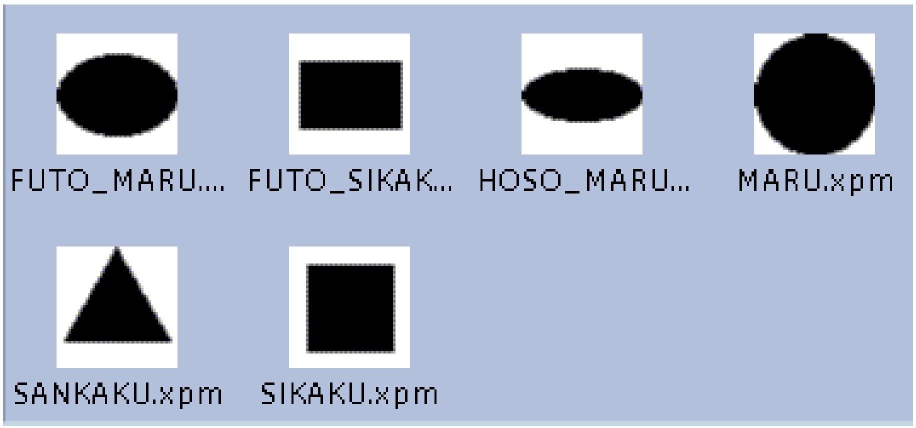


図 3.6 プリミティブテンプレート

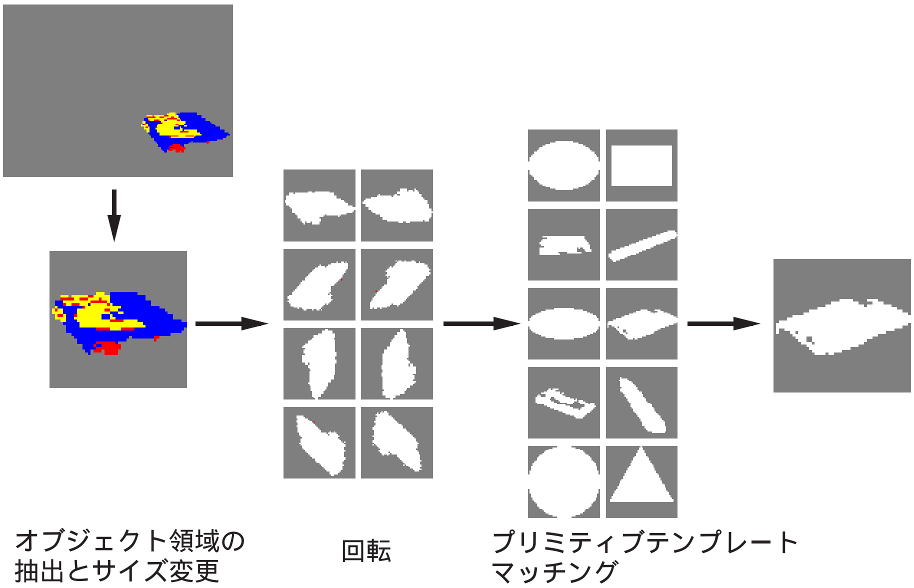


図 3.7 プリミティブテンプレートマッチング

- 色構成による絞り込み

  プリミティブテンプレートとのマッチングを終えたフレームは、次にオブジェクトを構成する色による絞り込みをする。各プリミティブテンプレートは、その形状を持つオブジェクトの色構成別に分類されたリストを持っていて、ここでは入力されたオブジェクトの色構成のオブジェクトのリストがあるかどうかを調べる。たとえば、図 3.2.(c) では (青・黄色) といった具合である。この情報はオブジェクトフレームデータを読み込む際に同時に調べておくので、このオブジェクトは「青・黄色で構成された四角い何か」という認識が行われたことになる (図 3.8)。

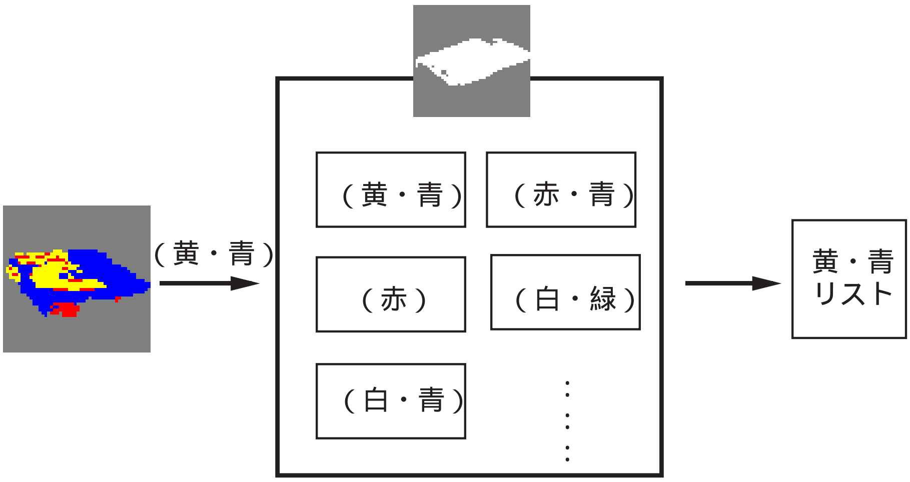


図 3.8　色構成による絞り込み

- オブジェクトテンプレートとのマッチング

  色構成による絞り込みを終えたら「青・黄色で構成された四角い何か」が自分の知識の中にあるかどうか調べる。ここで、自分の保持している「青・黄色で構成された四角い何か」に該当するオブジェクトテンプレートとのマッチングを試みる。この時のテンプレートマッチングには色情報も用いる。つまり、プリミティブテンプレートは 2 値化された “影” を用いて判定していたものを、今度はカテゴライズされた色情報も用いて当該色の分布

度合の近似度をも判定に利用する (図 3.9) のである。これにより、似た形はしていても、色分布の違うオブジェクトについても明確に分類でき、より原画像に近いオブジェクトの割出しができる。そして、このマッチングテストで整合するものがあれば、該当オクジェクト情報のデータファイル (タグと呼ぶ) を読み込んで、関連付けして行く。

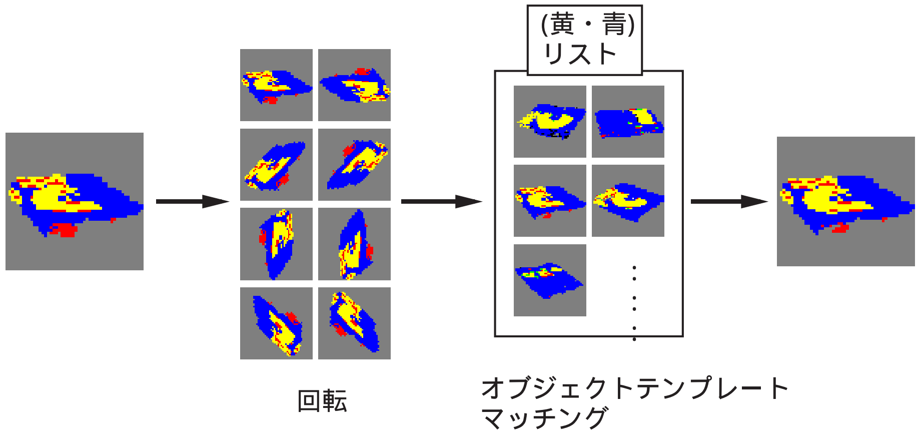


図 3.9 オブジェクトテンプレートマッチング

3.4.3 オブジェクトの学習

マッチングテストに失敗した場合、そのオブジェクトは見知らぬオブジェクトであったとして、人に対してそのオブジェクト画像を提示し新たなオブジェクト情報を得てデータベースへの追加をする。

- ユーザに対して未知オブジェクト情報を提示し新たな知識を得る

  ロボットはスナップショットを取り続けて行くうち、さまざまなオブジェクトを認識して行くが、“unknown” のラベルがついたままの未知のオブジェクトもまた溜め込んで行ってしまう。そこでロボットは機会を見つけては、人にそのフレームを提示し、情報を求める。人は、このオブジェクトに関して知り得る情報を教示してゆく。

これにより、ロボットは情報とともに、このオブジェクトのテンプレートを作成して行く。(図 3.10, 図 3.11)

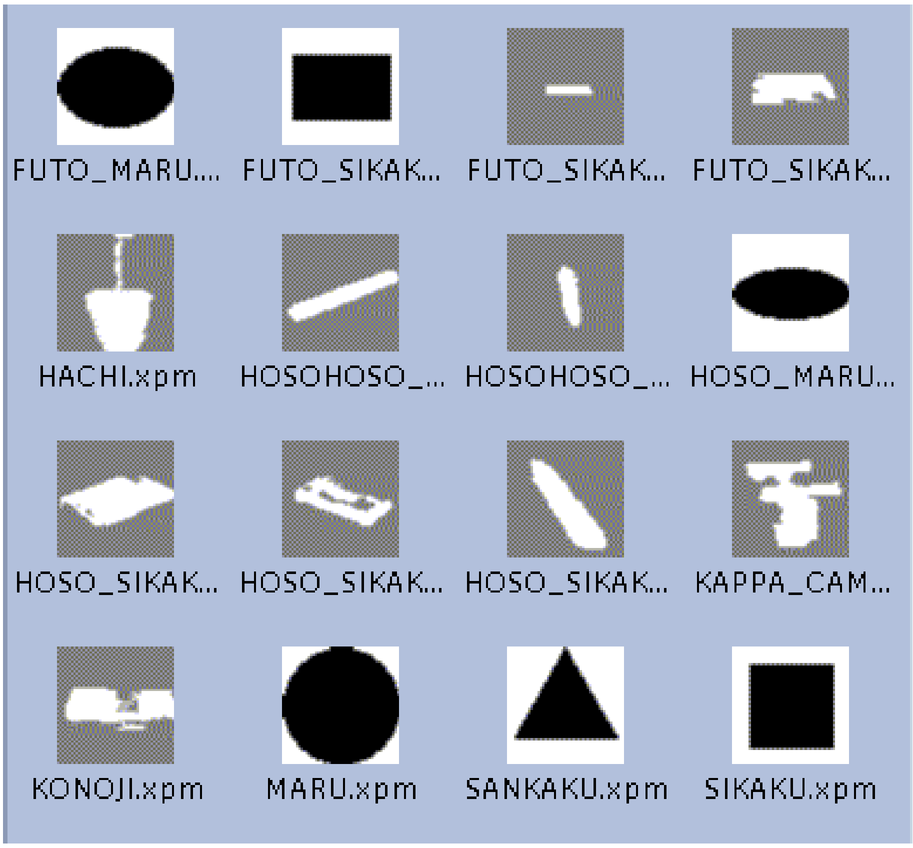
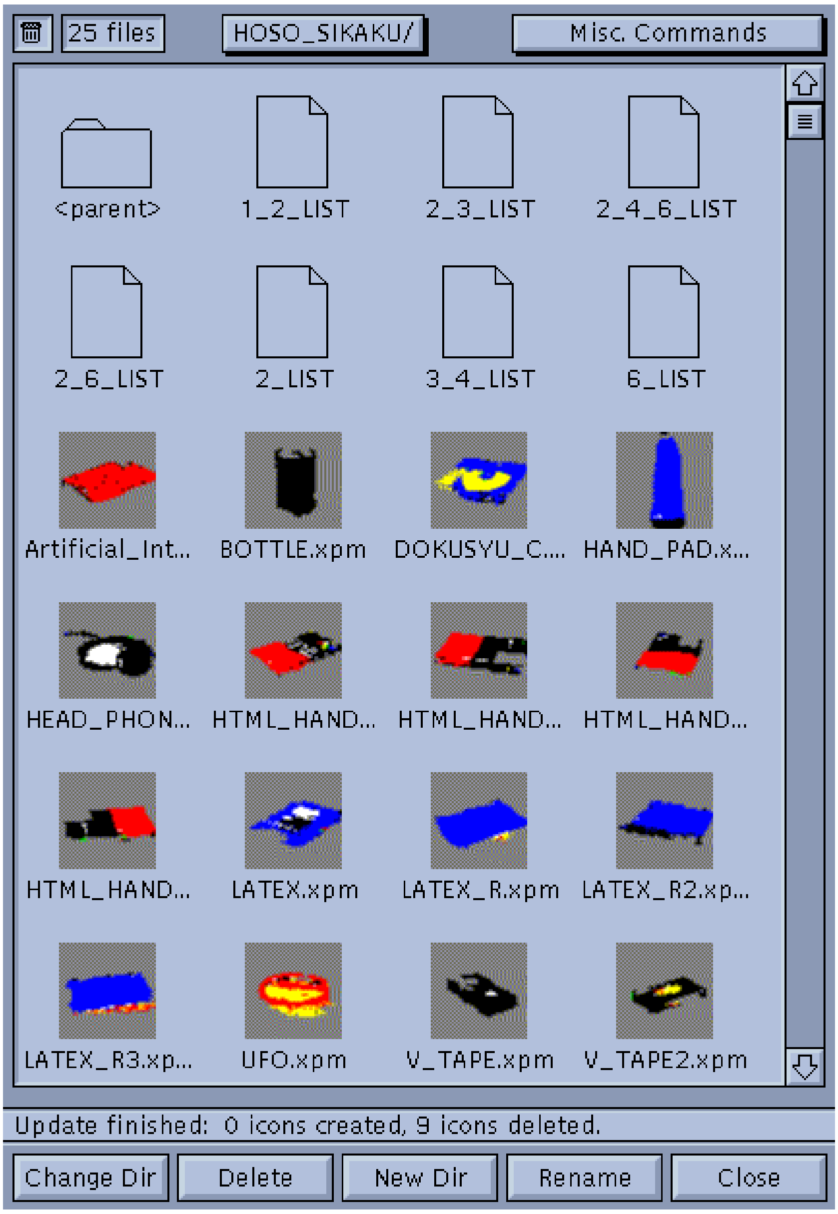


図 3.10 学習後のプリミティブテンプレート

- タグ情報の更新

  オブジェクトの情報は時々刻々と変化する。オフィス環境における備品は常にその所在が変化する可能性を含んでいる。そのため、オブジェクトタグにはそのオブジェクトがいつ、どこで発見されたかという情報とともに、そのスナップショットの名前を付け加えていく。今回はタグ情報に以下のものを用いた。

  – オブジェクト名



学習され増えて行くオブジェクトテンプレート

図 3.11 オブジェクトテンプレート

– プリミティブテンプレート名

– タグの名前

– 関連情報
(一般的な背景知識、“普段は○△にあるらしい” 等、何を書いても良い)

– 被参照フレーム名

全システムの内、学習に使用される部分を図示する (図 3.12)。

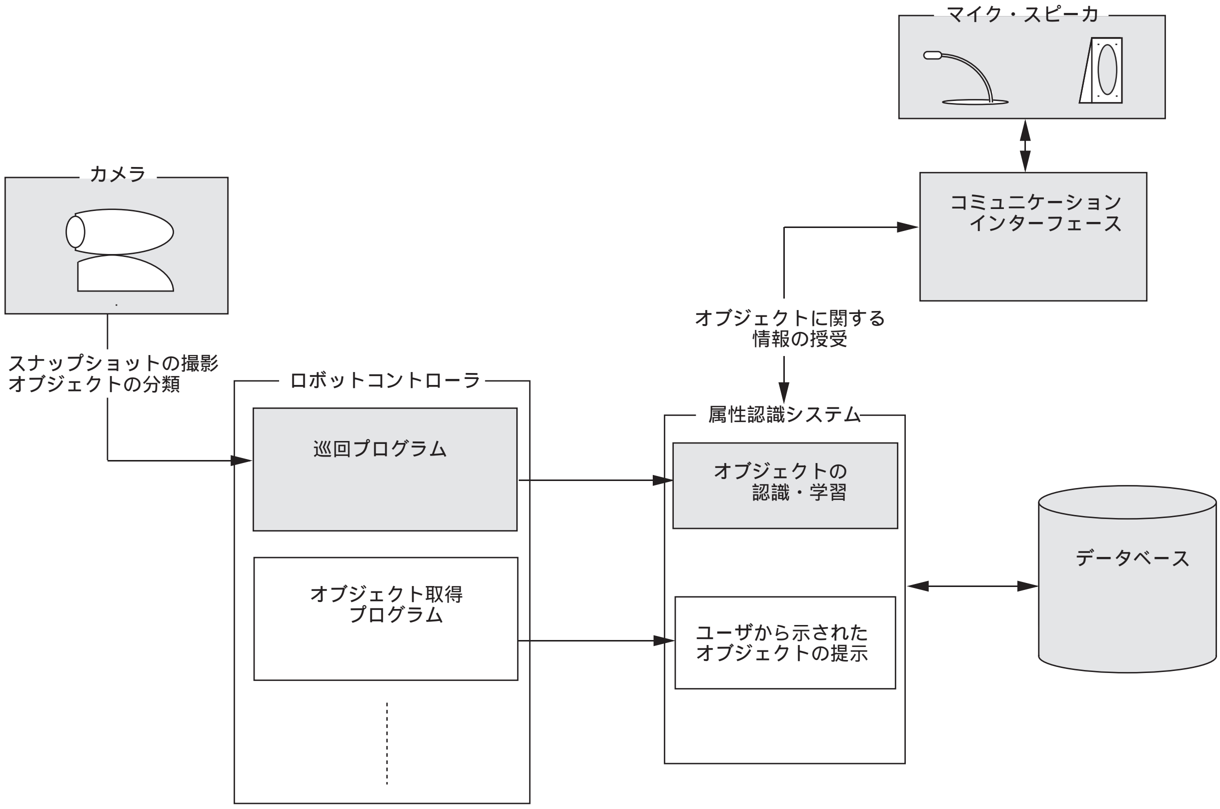


図 3.12 学習にかかるシステム

- ユーザから示されたオブジェクト情報を提示する・新たな知識を得る

  ユーザはロボットに対しあるオブジェクトを運んで来て欲しいと要求する。ロボットは自分の保持しているタグにキーワード名のものがあるか調べ、

マッチするものがあればタグに書かれている情報を提示し、ユーザからの指示をあおぐ。キーワードにマッチするタグが無ければ、今度は色や形状などの抽象的な情報を提示してもらい、“unknown” ラベルがついたオブジェクト情報の中からマッチングを試みる。ここでマッチングするものが発見され、ユーザの要求するものと確認が取れた場合、このオブジェクト情報に新たにキーワードによる意味付けをし、オブジェクトテンプレートの追加と、タグの被参照フレーム名情報の追加を行なう。

3.5 実験

ここでは “ロボット自身が人の手を借りず集めた知識の中からユーザの指示したものを提示させる” というタスクを通して属性学習システムの有効性について検証する。

実際にシステムの認識率を調べるために、以下のような実験を行った。

- 研究の主旨を知らせていない学生に「卓上の小物」という条件だけを与えてオブジェクトを 100 個集めさせる
- それらのオブジェクトに対して各 5 回ずつ別方向からスナップショットを撮り、属性情報を学習させる (パターンマッチにかかる閾値は 100 とした)
- 各オブジェクトに対して再びスナップショットを撮り認識システムにかける (プリミティブテンプレートマッチングの閾値は 60、オブジェクトテンプレートマッチングの閾値は 80 とした)

この実験によって正しい認識を示したオブジェクトは 27 種類あった。認識精度の高かったオブジェクトの特徴としては形状がシンプルなもの (図 3.13.(a))、模様がはっきりしているもの (図 3.13.(b)) が上げられる。また、認識精度の低かったものの特徴としては、形状が複雑なもの (図 4.6.(a))、形が不定のもの (図 4.6.(b))、背景色と見分けが付かないもの (図 4.6.(c)) ということが上げられる。

実世界におけるオブジェクトには複雑な形状のものが多く、環境は次々と変わるため、ロボットが認識できないものはなかなか減ることはない。同じオブジェクトでも、奥行きや、回転、厚みなどで、スナップショットから得られる見た目の形状情報は一定ではない。そのため、一つのオブジェクトの認識精度を向上させるだけでも複数回の属性学習を要するが、それでも学習を繰り返すうちに単純な形 (3 次元的に複雑な形状をしていないもの) についての認識率は高くなり、ユーザの求めるものを正しく表示できるようになった。これらのオブジェクトに関しては、早いもので二回、多くても五回程度の属性学習で識別できるようになることが分かった。

閾値はユーザの手によって変更が可能であるが、上げすぎると整合率が高いものでも見知らぬオブジェクトとして学習を進め、効率が悪い。逆に下げすぎると

別のオブジェクトなのに間違った識別をする可能性が高くなる。環境に応じて適切な値を求めることも重要である。

図 3.13　認識精度の高かったオブジェクト

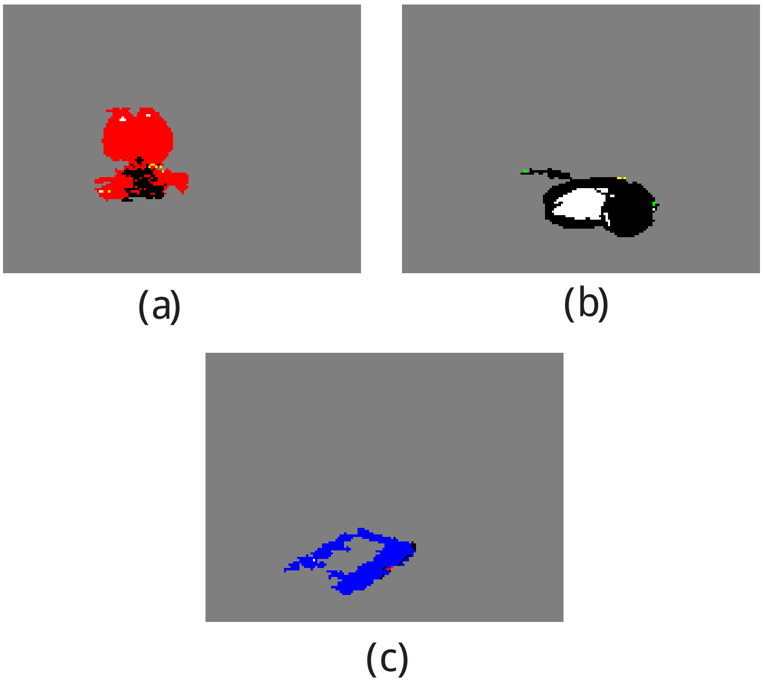


図 3.14　認識精度の低かったオブジェクト

3.6 計算オーダー

十分な学習により、全てのプリミティブテンプレート (i) に対し均一にオブジェクトの色構成リストが生成 (j) されたとし、また、各色構成分類によるリストに対して均一にオブジェクトファイルが生成 (k) されたものとする。全学習オブジェクトテンプレート数を N とすれば

$$\begin{aligned} N &= i(\text{プリミティブテンプレート数}) \\ &\quad \times j(\text{色構成分類によるリスト数}) \\ &\quad \times k(\text{オブジェクトテンプレート数}) \end{aligned}$$

で表される。

全検索にかかる計算オーダ $O(N) = O(i \times j \times k)$ に対し、属性認識における計算オーダーは $O(i + 1 + k)$ である。

(注) 色構成による絞り込みにかかる計算オーダーはそのリストがあるかないかを調べるだけなので、常に 1 である。

第 4 章

システムの実装と検証

ここではこれまでに述べてきたシステムを実際にロボットに実装し、その動作の検証を行う。

4.1 実装

今回使用したロボットシステムのハードウェア構成は以下の通り。

- モバイルロボット： RWI(Real World Interface) B21 モバイルロボット
- カメラ ： SONY EVI-D30
- カメラ制御端末 ： 東芝 LIBRETTO60
- 音声認識システム： SONY VAIO-C1、IBM ViaVoice、スタンドマイク

ロボット制御ソフトウェアには RWI 社製 BeeSoft ver 1.3.4 を用いた。各ハードウェアの仕様については付録として後述する (付録 B)。

4.2 検証

ロボットに以下のような作業をさせることを考える。

1. ロボット自身が人の手を借りず集めた知識の中からユーザの指示したものを提示させる
2. その物品を受け取りに行く、もし求めるオブジェクトがすでに移動していたなら、周辺にいる人にどこへ行ったか尋ね、新たに取りに行く

このタスクを実行させるため以下のようなステップを踏む (図 4.1)。

1. スナップショットの撮影
   ロボットに研究室内を巡回させ、任意の場所でスナップショットを撮らせる
2. 属性認識
   スナップショットからフレーム分けされたオブジェクトをマッチングテストにかけ、既知のオブジェクトかどうか調べる
3. 属性学習
   未知のオブジェクトが検出された際にはユーザによりオブジェクトに関する知識を得る
4. 蓄えた情報をもとに、ユーザの求めるオブジェクトの提示を行う
5. ユーザからの依頼によりロボットはそのオブジェクトのある (であろう) 場所へ移動
6. 目的地に着いたらその場にいる人からオブジェクトを受け取る
7. オブジェクトが移動していたなら、その位置情報をもらって新たな目的地とする
8. オブジェクトの取得を指示したユーザの所へもどる

4.2.1 ロボットの巡回とオブジェクトの把握

ロボットは研究室内の地図情報を持っており (図 4.2)、この地図情報をもとに位置情報を管理し、また室内の移動を行うことができる。

ロボットは巡回中定められた場所で、首を振ってそれぞれの場所ごとのスナップショットを撮り、オブジェクトの情報を収集し、認識できるものであれば、時間、場所情報とともに記憶して行く。認識に失敗したものは “unknown” 情報タグをつけて別途保存し、巡回終了時に人に対してオブジェクトの情報を求める (図 4.3)。

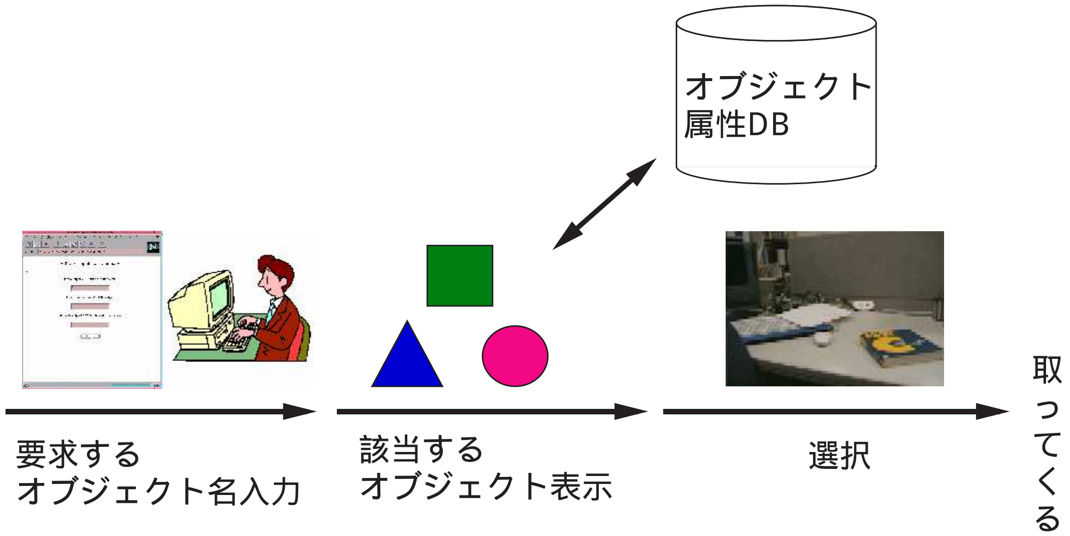


図 4.1 オブジェクトの取得

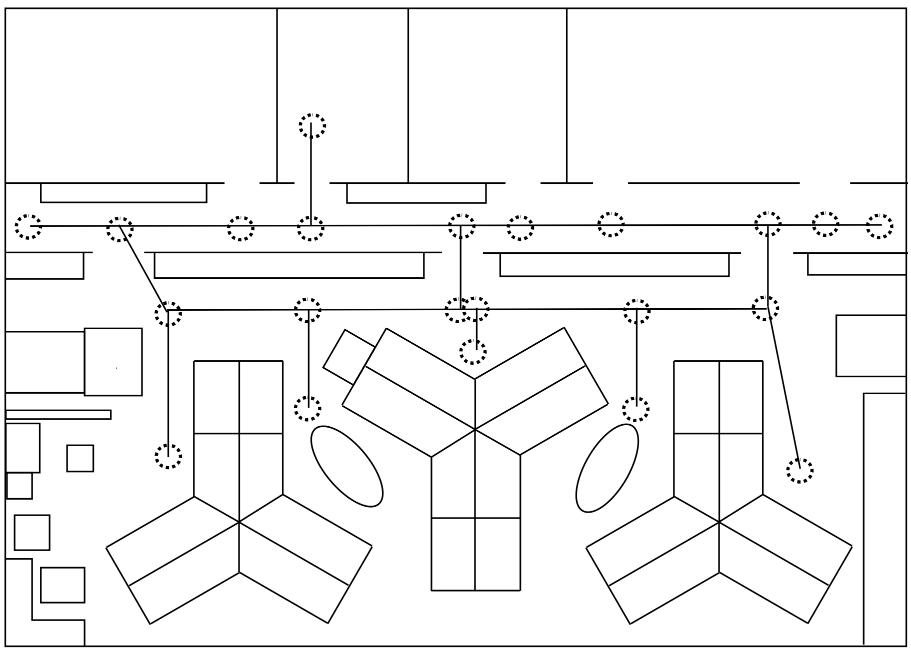


図 4.2 環境地図 ([6] より)

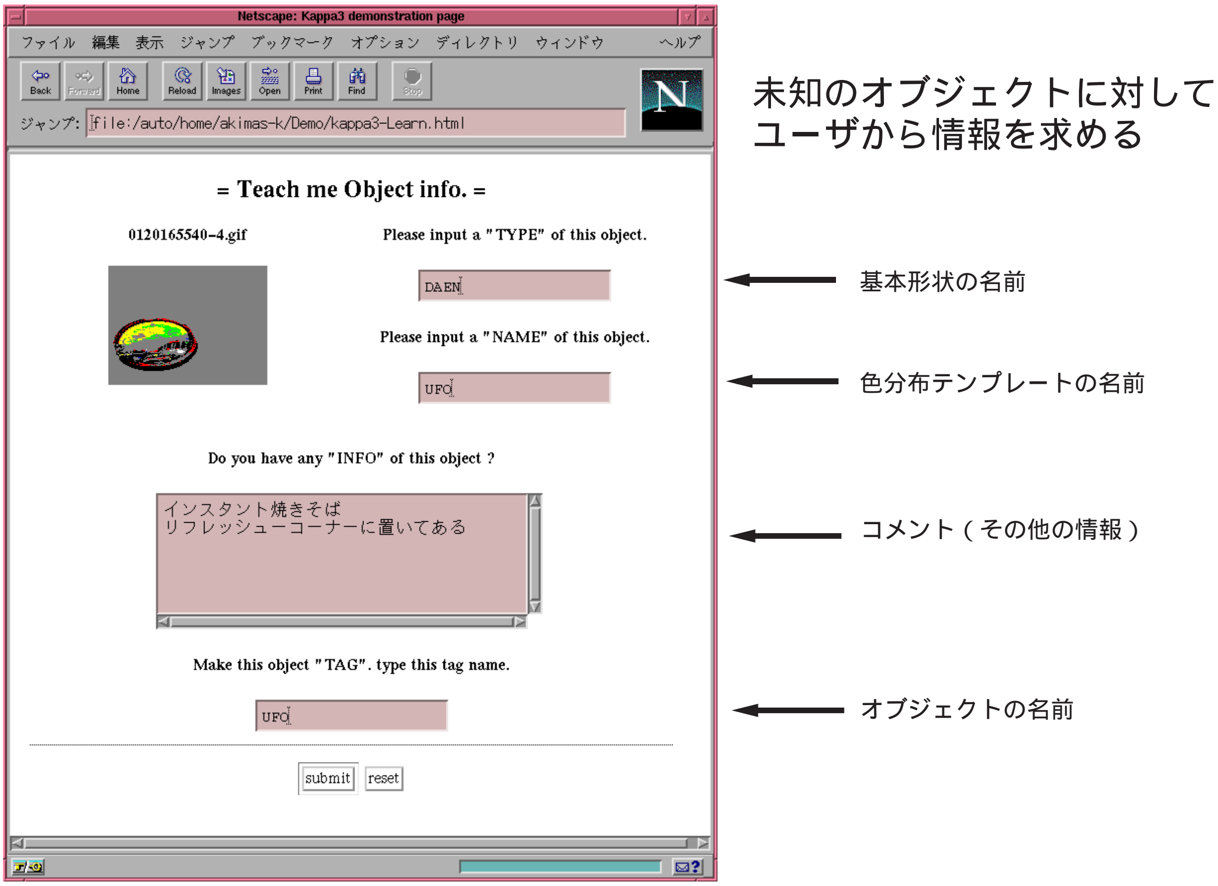


図 4.3 属性学習インターフェース

4.2.2 ユーザの求めるオブジェクトの提示

本件で用いているオブジェクトデータは先の認識で木構造の形に分類されている (図 4.4)。この構造を用いれば、曖昧な概念による検索も可能である。以下にその手法を述べる。

インターフェースとして提示の方法としてはいくつかあるが、ここではユーザにとって分かりやすいように WWW ページによるスナップショットの表示という方法を取る。

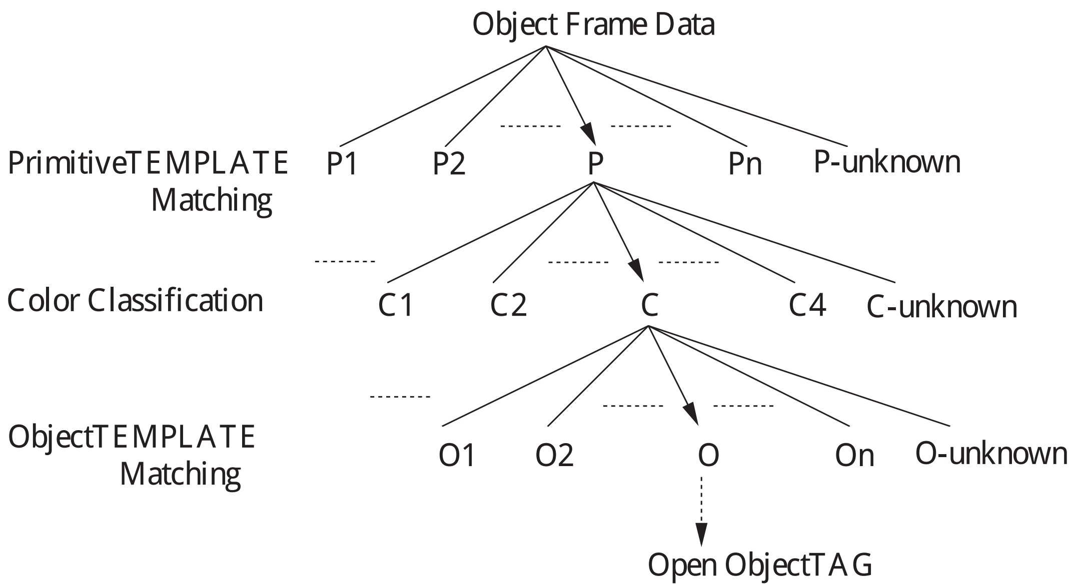


図 4.4 木構造に分類されたオブジェクトデータの概念図

まず、ユーザはロボットに対し求めるオブジェクトの名前、もしくはその属性情報をキーワードとしてロボットに尋ねる。たとえばキーワードとしてオブジェクト名 “DOKUSHU_C” を用いると、“DOKUSHU_C” という名前のタグがあるか調べ、あればそれを表示する (図 4.5)。

もしなければ、ロボットはその旨ユーザに伝え、一段抽象的な表現によるオブジェクトの教示を求める。ここでユーザはプリミティブテンプレート、色分類情報をロボットに伝えればよい。たとえば「形は四角で色は青である」と言った具

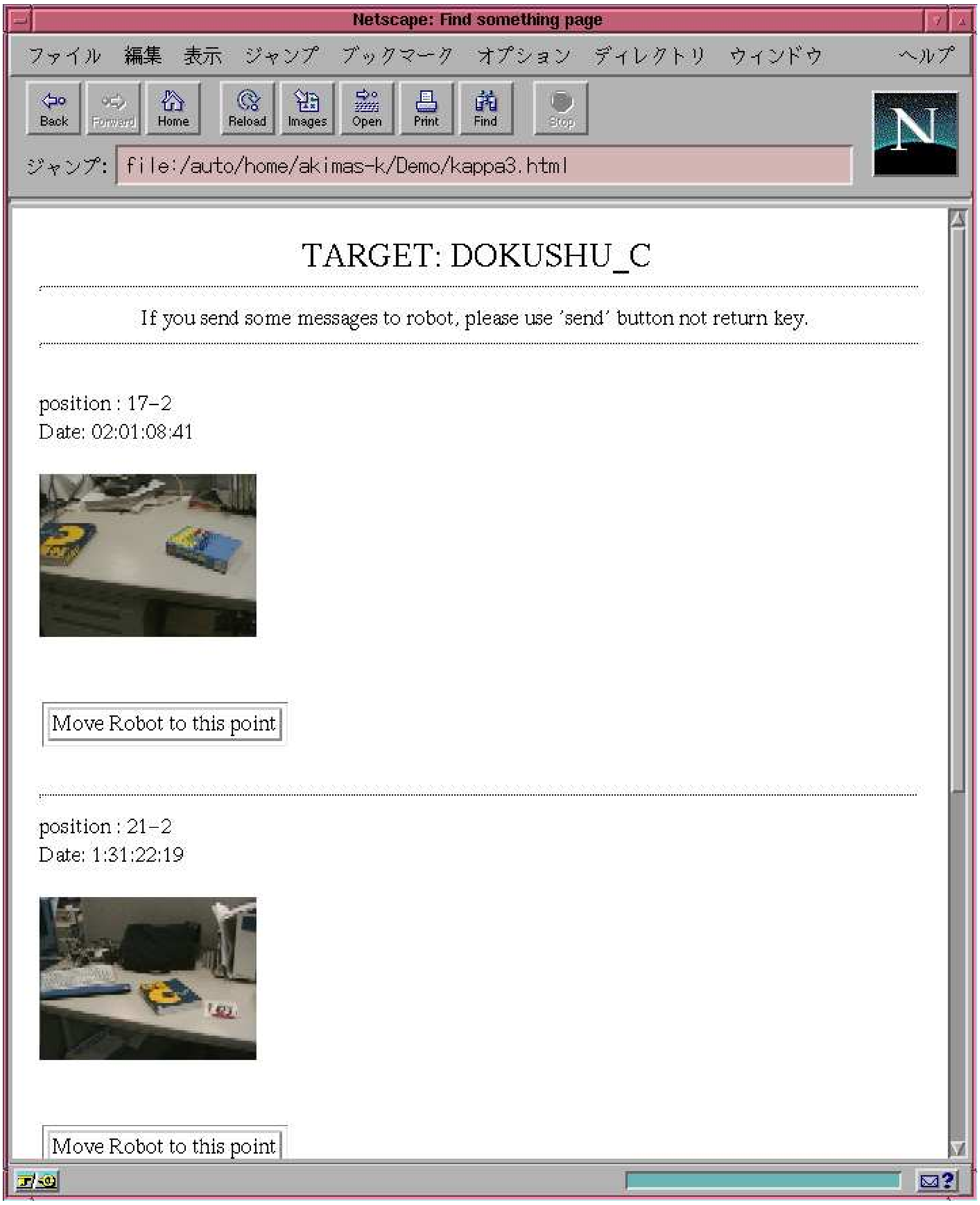


図 4.5 WWW によるオブジェクトの表示

合である (図 4.6)。

このときロボットはオブジェクトテンプレートマッチングに失敗して “unknown” のラベルが付けられているオブジェクト (図 4.4における O-unknown に相当) の中からユーザの求めるものに近いものをスナップショットとともに一覧として提示する。

ここでも見つからなければ色分類に失敗したもの (図 4.4における C-unknown に相当) の中から「四角い」といった形状情報のみの情報提示を受けることができる。

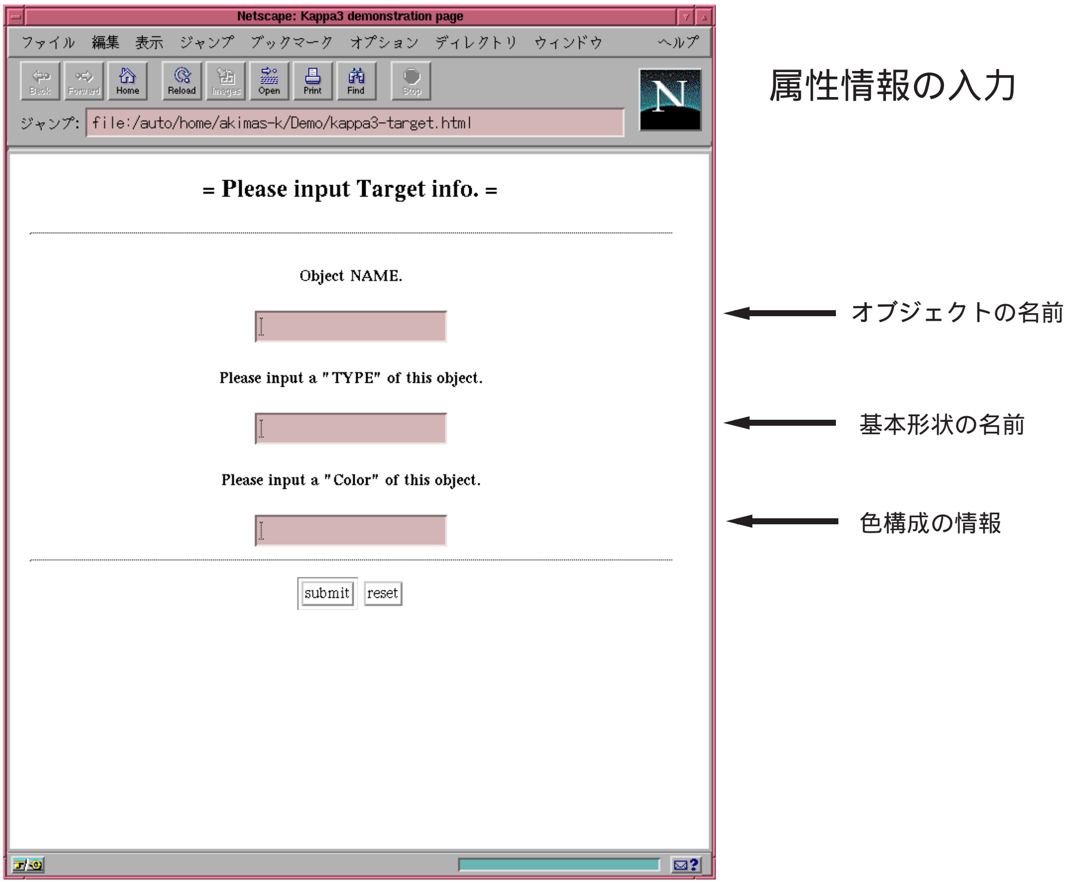


図 4.6 概念レベルでのオブジェクトの検索

ユーザはこの中から自分の探しているものがあればそれをロボットに伝える。

このとき、ロボットはこのオブジェクトデータから “unknown” ラベルを外し、ユーザから得た知識をもとに、再学習し、次回からの検索に活かすようにする。

システムを図示する (図 4.7)。

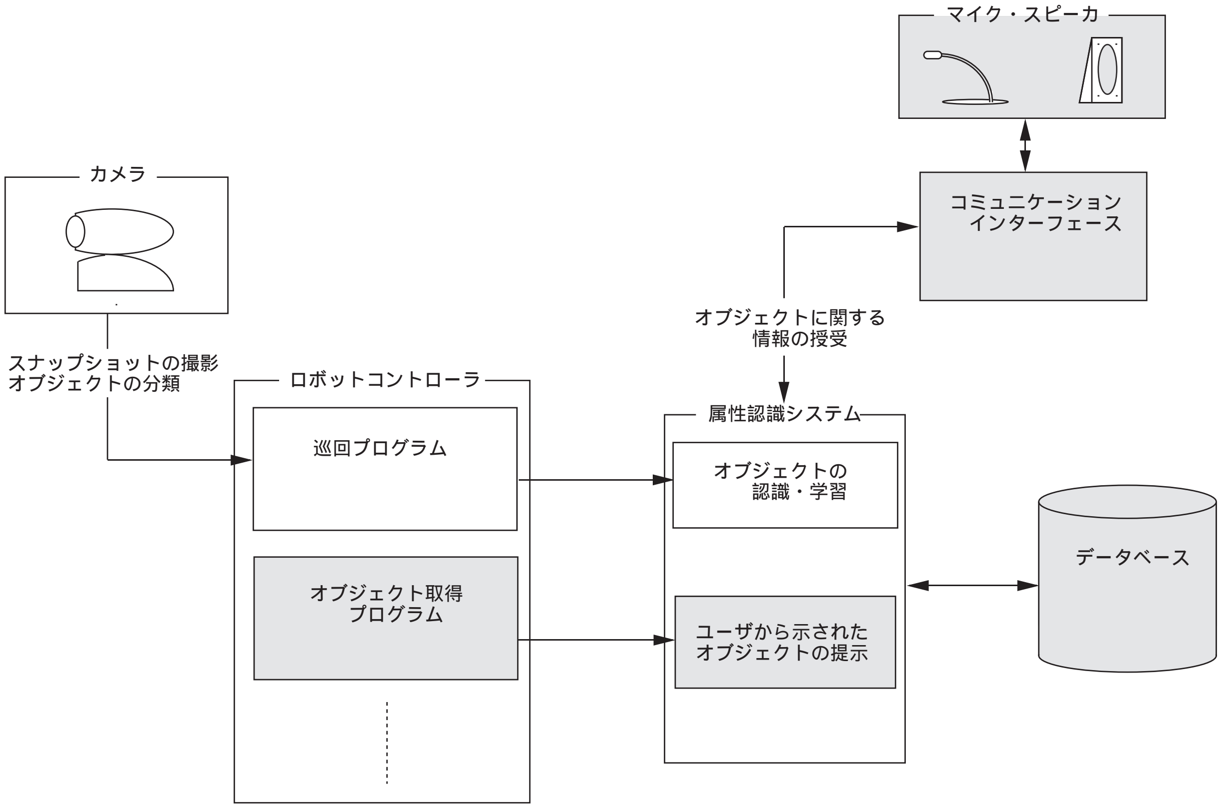


図 4.7 オブジェクトの提示にかかるシステムの動作

4.2.3 オブジェクトのある場所への移動と取得

ユーザによってオブジェクトのスナップショットが選択されたなら、ロボットはそのスナップショットの位置情報をもとに、そのオブジェクトのある場所へと移動する。目的地に着いたら周辺にいる人に対してオブジェクトをくれるよう要求し、もしオブジェクトがすでに移動してその場所になければ移動先を教えてもらう。一定時間内に人から情報をもらえなければユーザの元へ戻る (図 4.8)。

環境の認識

オブジェクトの
位置へ移動

オブジェクトの受け取り

ユーザへの
受け渡し

図 4.8　オブジェクトの取得と受け渡し

- 対話するロボット

  ロボットが活動する場所には必ずしもインターフェースとしての計算機等があるとは限らない。また、人との関係をより近づける意味でも対話による意志伝達は重要であると考えられる。そこで、円滑なコミュニケーションのために市販の音声認識ソフトを用いて以下のような対話機能の提供も行った。

  – 物の所在に関するユーザの質問に対する受け答え
  – オブジェクト情報に関する知識のユーザとの交換

  対話による行動生成には音声認識ソフトによって、インタラクションシステムであるウェブページの所定の欄に文字を打ち込むという手法を用いる。これによりユーザにはキーボードと音声という二つの入力メディアが選択可能になり、置かれている環境に応じて使い分けることができるようになった。

第 5 章

結論と展望

5.1 結論

本研究では、人間の持つ属性認識モデルをロボットに実装し、室内にあるオブジェクトの位置情報管理を行わせ、その有効性を検証した。オブジェクトの条件が限られてはいるが属性学習・認識システムの実装により曖昧な情報でもユーザの求めるオブジェクトが効率的に検索でき、属性認識によるオブジェクトの分類法の有効性が確かめられた。また、昨年のシステムの上に構築したことで検索方法の幅が広がりユーザにとって利用しやすいシステムが構築できた。

5.2 展望

今回の実験で単純なテンプレートマッチにおいても、単純な形状のものであれば少ない学習で良い認識精度を示すことが分かった。また、3 次元的に複雑な形状をしていても、十分な学習を行うことである程度の認識精度を出せることが分かった。さらに先にも述べた通り、今回用いた属性認識システムは全検索に比べてオブジェクトの認識にかかる計算オーダーも少なく有効であることも分かった。

また、オブジェクトテンプレートのマッチングに失敗したとしても、それまでの過程から、おおよその概形、色構成が分かるので、それを元にユーザとのやりとりの間に目的のオブジェクトが発見できる可能性も高い。

しかし、現時点では次のような問題点もある。今回用いた方法では一眼による画像処理のため、スナップショットからオブジェクトを抽出した際に奥行きを無視しており、そのため同じオブジェクトでも、さまざまな角度における見た目の違いによって複数のプリミティブテンプレートにマッチしてしまう現象が見られ

た。また、オブジェクトの大きさも考慮には入れなかったためフロッピーディスクと ディスプレイを混同してしまうといった現象も起こった。

これらは、ステレオ視などの距離を計るシステムの構築による概形の推測によって改善がなされると思われる。そうなれば、見た目の形状に捕らわれずに効果的な学習がなされ、検索時間も改善されると期待される。

次に色分解の問題があるが、蛍光灯によるハイライト、窓から差し込んでくる太陽光の時間による変化などで、同じオブジェクトの色が環境の違いよって異なることが多々あることが分かっている。これもまた、学習回数を増やす大きな要因となっている。オブジェクトの色の正規化についてもこれからの更なる研究が必要とされる。

謝辞

主指導教官である西田豊明教授にはすばらしい研究の場を与えて頂いた上、さまざまな助言・指導を頂きました。

横矢直和教授にはお忙しい中本研究の副指導教官になって頂きました。

武田英明助教授には、研究の方向性や内容について数多くの助言を頂き、また、本論文をまとめるにあたり、適切な指導と助言を頂きました。

上野敦志助手には本研究を進めるにあたり、さまざまな助言を頂きました。博士課程 2 年の寺田和憲氏には、研究会等を通じて様々な意見を頂きました。

修士課程 1 年の吉村 俊哉、宮本 圭 両氏はじめ、修士過程 1 年の方々にはロボットの環境設定ほか、プログラミングの手伝いをして頂きました。

この場を借りて深くお礼申し上げます。

また、すべての研究室の皆様には楽しい研究生活の場を与えて頂いたことに感謝し、お礼申し上げます。

最後に、研究活動を支えて頂きました谷村優香里秘書に感謝致します。

参考文献

[1] 麻生英樹 本村陽一 松井俊浩 速水悟 原功：
対話ベース学習によるオフィス環境での自律ロボットナビゲーション, 人工知能学会研究会,AI シンポジウム'95 (1995).

[2] 松井 俊浩 他：
学習型所内事情通ロボットの計画, 日本ロボット学会大会 (1995).

[3] H.Asho et.al：
Acquiring a Probabilistic Map with Dialogue-Based Learning, Proc. of ROBOLEARN-96 (1996).

[4] Smithsonian Institution Lemalson Center:
Minerva, The Robotic Tour Guide,
http://www.si.edu/nmah/lemel/

[5] 電子情報通信学会情報・システムソサエティ パターン認識・メディア理解研究会 アルゴリズムコンテスト:
情報・システムソサイエティ大会ソサイエティ併催事業として実施 (1998).
http://yindy1.aist-nara.ac.jp/PRMU/contest/schedule.html

[6] 内田恵久：
弱い知識構造を用いた人間とロボットとのインタラクションに関する研究, 奈良先端科学技術大学院大学情報科学研究科 修士論文 (1999).

[7] 内田恵久, 武田英明, 西田豊明: 弱い知識構造に基づくロボットの行動生成, 情報処理学会研究報告 知能と複雑系 111-4 (1998).

[8] 西田豊明 前田晴美 平田高志：
日常記憶の共有によるコミュニティインタラクション支援を目指して -CoMeMo, システム/制御/情報 Vol.41, No.8 (1998).

[9] 長尾真:
エレクトロニクス モノグラフシリーズ 2 画像と言語の認識工学, コロナ社 (1989).

[10] Kazunori Terada, Takayuki Nakamura, Atsushi Shibata, Hideaki Takeda, and Toyoaki Nishida:
A Cognitive Robot Architecture for a Simple Hardware System, Advanced Robotics (Submitted) (1999).

[11] Arnaud Le Hors:
XPM Manual The X PixMap Format (1996).

[12] 日本ロボット学会 学生メーリングリスト編集委員：
パーソナルロボットに関するアンケート,
(1997). *http://gull.me.es.osaka-u.ac.jp/ mitsuda/anq/kekka.html*

[13] 長谷川純一 輿水大和 中山晶 横井茂樹 共著：
画像処理の基本技法, 技術評論社 (1986).

付録

A．画像処理にかかるアルゴリズム

今回用いた画像処理の手法のいくつかを紹介する

A.1 画像のカテゴライズ

参考文献 [10] でも述べられているが、カテゴライズには RGB 系と HSV 系の 2 つの色域処理系を用いる。まず、カメラから送られて来る画像データは RGB 系なので HSV 系に変換する前に HSV 系では処理仕切れない白、黒の領域を編集して残った領域を以下のアルゴリズムに当てはめて HSV 系へと変換する。HSV 系では色相環に置ける角度の設定で色分割が容易になっている。本件では、さらにこの機能を拡張し、RGB 処理の際に灰色領域もチェックできるようにした。

- 白黒領域の割り出し。白黒領域の割り出しには RGB 処理系において RGB のそれぞれに閾値を与えてやれば良い。すなわち、「 $(R, G, B) > (170, 170, 170)$ であれば白 」、「 $(R, G, B) < (70, 70, 70)$ であるなら、黒 」といった具合である。

- 灰色領域の割り出し。灰色領域の割り出しにも、先出の技法が使える。すなわち、「 RGB の各値が近ければ灰色 」である。具体的には、「 RGB のもっとも高い値 max と もっとも低い値 min の差が 40 以内で、max、min でない値 min が max、min の中間値付近 $((max - min)/2$ に対して誤差が ± 16 以内) にあればその色は灰色 」である。

- HSV におけるカテゴライズここまでの処理で白・黒・灰色のいずれにも属さなかった色に付いて HSV 処理系では色相環における角度の調整を用いて各色のカテゴライズを試みる。ここでは赤・黄・緑・青の各色について赤 (0 °~ 39 及び 340 °~ 360 °)、黄 (40 °~ 65 °)、緑 (66 °~ 181 °)、青 (182 °~ 260 °) に設定してある。

```c
/* RGB to HSV */
void rgb2hsv(unsigned char r,
             unsigned char g,
             unsigned char b,
             float *h,
             float *s,
             float *v)
{
  float maxv, minv, delta;

  if(r < g){
    maxv = g < b ? b : g;
    minv = b < r ? b : r;
  }
  else{
    maxv = r < b ? b : r;
    minv = b < g ? b : g;
  }

  *v = maxv;
  *s = (maxv - minv) / maxv;

  if (*s == 0.0)
    *h = -1.0; // -1 UNDEFINED
  else
    delta = maxv - minv;

  if (r == maxv) *h =     (float)(g - b) /delta;
  if (g == maxv) *h = 2 + (float)(b - r) /delta;
  if (b == maxv) *h = 4 + (float)(r - g) /delta;

  *h *= 60.0;
  if (*h < 0) *h += 360.0;
}
```

A.2 オブジェクトの抽出アルゴリズム

カテゴライズされたスナップショットからオブジェクト領域を切り出すために以下の操作を行った。

1. スナップショット内のノイズカット
2. 色領域の輪郭線を抽出する
3. 輪郭線の内部を塗りつぶした領域を別のフレームへ格納する
4. 塗りつぶされた箇所を原画像から消去する

単なる塗りつぶしから始めないのは、領域内に背景色と同色の領域があった場合これを無視する形になるためであり、輪郭線という堤防を設けることで、領域内の色領域は背景とは別として扱う。

もちろん、ドーナツ型の (穴のあいた) オブジェクトに関して処理が弱くなるが、ハイライトやグレーの絵柄の為に部分的に背景色として認識されるオブジェクトに比べるとその数は圧倒的に少ない。

- ノイズカット

  カメラで取った画像には常に小さなノイズが発生する。ノイズ除去にはさまざまな手法があり、どれも本研究で大きな違いを見せるものではないので、それらを参照して頂きたい。

- 輪郭線の抽出

輪郭線の抽出には文献 [13],pp 70-73 にある輪郭線抽出関数を用いた。このアルゴリズムについての詳細な説明はそちらを参照されたい。

```
/*************************************
  輪郭追跡
  ia : 入力の二値画像配列
  li, lj[ln] : 境界線座標リスト
  lp : リストのポインタ
*************************************/
border(ia, li, lj, ln, lp)
unsigned char ia[WIDTH][HEIGHT];
int li[], lj[], ln, *lp;
{
    int i, j, i1, j1, ds, nc;
    i1  = WIDTH  - 1;
    j1  = HEIGHT - 1;
    *lp = 0;

/*************************
  画像のはしを切り落とす
*************************/
    for(i = 0; i < WIDTH; i++) {
ia[i][0] = ia[i][j1] =0;
    }
    for(j = 0; j < HEIGHT; j++) {
ia[0][j] = ia[i1][j] = 0;
    }

/*********************************************************
  ある点 (i, j) が有色ピクセルとして検知された時、
  四方が有色ピクセルで囲われていなければ処理を進める
  (囲われていればその点は領域の内部であり、輪郭ではない)
*********************************************************/
    for(i = 1; i < i1; i++){
        for(j = 1; j < j1; j++){
            if(ia[i][j] == 1){
                nc = ia[i][j+1]&ia[i-1][j]&ia[i][j-1]&ia[i+1][j];
                if(nc == 0){
                    if     (ia[i+1][j] == 0) ds = 6;
                    else if(ia[i][j-1] == 0) ds = 4;
                    else if(ia[i-1][j] == 0) ds = 2;
                    else if(ia[i][j+1] == 0) ds = 0;
                    border_s(ia, i, j, ds, li, lj, ln, lp);
                }
            }
        }
```

```
    }
    for(i = 1; i < i1; i++) {
        for(j = 1; j < j1; j++) {
            if(ia[i][j] > 0) ia[i][j] = 1;
        }
    }
}

/**************************************
  ia            : 入力の二値画像配列
  is,js         : 追跡開始点
  ds            : 次検索開始点方向
  li,lj[ln]     : 境界線座標リスト
  lp            : リストのポインタ
**************************************/
border_s(ia, is, js, ds, li, lj, ln, lp)
unsigned char ia[WIDTH][HEIGHT];
int      is, js, ds, li[], lj[], ln, *lp;
{
    static char ip[] = { 0, -1, -1, -1, 0, 1, 1, 1},
                jp[] = { 1, 1, 0, -1, -1, -1, 0, 1};
    int i, j, iw, jw, nb, n,nc;

    i = iw = is;
    j = jw = js;

    n  = (ds + 4)&7;
    ds = -1;
    nb = -1;

    nc =  ia[i-1][j-1] + ia [i-1][j] + ia[i-1][j+1]
        + ia[i  ][j-1]               + ia[i  ][j+1]
        + ia[i+1][j-1] + ia [i+1][j] + ia[i+1][j+1];

    if(nc > 0)
        do{
            li[*lp]    = i = iw;
            lj[*lp]    = j = jw;
            ia[iw][jw] = 3;

            (*lp)++;
            n = (n + 4)&7;

            do{
                n  = (n + 1)&7;
                iw = i + ip[n];
                jw = j + jp[n];
            } while (ia[iw][jw] == 0);
```

```
            if(ds < 0){
                if(nb < 0)  nb = n;
                else        ds = nb;
            }

            if(*lp >= ln){
                *lp = -1;
                goto error;
            }
        }while ((i != is) || (j! = js) || (n! = ds));
    }(end if)

    else{
        li[*] = is; lj[*lp] = js; ia[i][j] = 2;
        (*lp)++;
    }

    li[*lp] = lj[*lp] = -1;
     (*lp)++;

    error:
}
```

- 輪郭線内部の塗りつぶし

  あるフレームデータ Frame[i][j] に対し 0 を背景、それ以外はオブジェクト領域とするフレームデータの端から調べてオブジェクト領域 (有色ピクセル) が見つかればその有色ピクセルに隣接する有色ピクセル (オブジェクト領域) 情報をFrame2[i][j] に格納しオリジナルの領域を塗りつぶしていく。

```
/**********************
  領域塗りつぶし関数
**********************/
void Fill(int i, int j)
{
    Frame2[i][j] = Frame[i][j];      /* Frame2 にコピー */
    Frame[i][j]  = 0;                /* 塗りつぶし */

    if (Frame[i][j-1] > 0)           /* 上に有色ピクセルがあるか */
        Fill(i, j-1);                /* あればその座標で再帰       */
    if (Frame[i+1][j] > 0)           /* 右 */
        Fill(i+1, j);
    if (Frame[i][j+1] > 0)           /* 下 */
        Fill(i, j+1);
    if (Frame[i-1][j] > 0)           /* 左 */
        Fill(i-1, j);
}
```

- テンプレートマッチング

  オブジェクトフレーム画像とテンプレート画像をマッチングにかけるオブジェクトデータとテンプレートデータの “and” と “xor” を取り、$and = 1$ なら得点を、$xor = 1$ なら減点をする。整合率の計算は 得点ポイント “U”、減点ポイント “D” として、以下の式であらわす。

$$整合率 = \frac{U}{U + D}$$

```
/* Frame[i][j]    : オブジェクトフレームデータ */
/* Template[i][j]: テンプレートデータ          */

double Matching_Test(double MMR)
{
    int i, j, r;
    double Matchrate;

    int MatchCountU;  /* テンプレートと and をとった時のマッチ率 */
    int MatchCountD;  /* テンプレートと xor をとった時のマッチ率 */

    for(r = 0; r < RADIAN; r++){
        MatchCountU = 0;
        MatchCountD = 0;
        for(j = 0; j < s_HEIGHT; j++){
           for(i = 0; i < s_WIDTH; i++){
              if((Frame[i][j] != 0) && (Template[i][j] != 0))
                 MatchCountU++;
              else if((Frame[i][j] != 0) != (Template[i][j] != 0))
                 MatchCountD++;
              }
              MMR = Calc_MMR(MMR, MatchCountU, MatchCountD);
        }
    }
    return MMR;
}

double Calc_MMR(double MMR,
                int MatchCountU,
                int MatchCountD)
{
    double Matchrate;

    Matchrate = (    (double)(MatchCountU)
                   / (double)(MatchCountU + MatchCountD) * 100);
    if(Matchrate > MMR)
       MMR = Matchrate;
    return MMR;
}
```

A.3 テンプレートマッチングにかけるまでの手続き

テンプレートの大きさは常に一定である。そのために、マッチングを行う際にはフレーム内のオブジェクト領域をさらに加工する必要がある。また、オブジェクトの向きを考慮して、回転させることも考慮しなければならない。

- オブジェクト画像の縮小・拡大

  テンプレートとマッチングさせるためにフレームからさらにオブジェクト領域を切り出し、テンプレートと同じサイズに修整する。このとき、オブジェクトを回転させることを考慮して、回転させても画像がテンプレートからあふれることのないよう、オブジェクトの中心から外縁に対して最長の距離をテンプレート半径に合わせるようにサイズの変更を行う (図 1.1)。

  `object-r` は先の領域抽出により求められた輪郭座標からオブジェクト領域の中心点に対して最長の距離のものを選択した。

  サイズの変更には以下の式を用いる。

  縮小率 `scale`：$scale = \frac{R}{object-r}$

  サイズ変更：

$$\begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} i'/scale \\ j'/scale \end{pmatrix}$$

$$ScaleFrame[i'][j'] = FrameOrg[i][j]$$

  「ピクセル (i’, j’) の縮小前座標 (i, j)」を求め、縮小前座標 (i, j) の色情報から縮小後座標である (i’, j’) の値を変更していく。

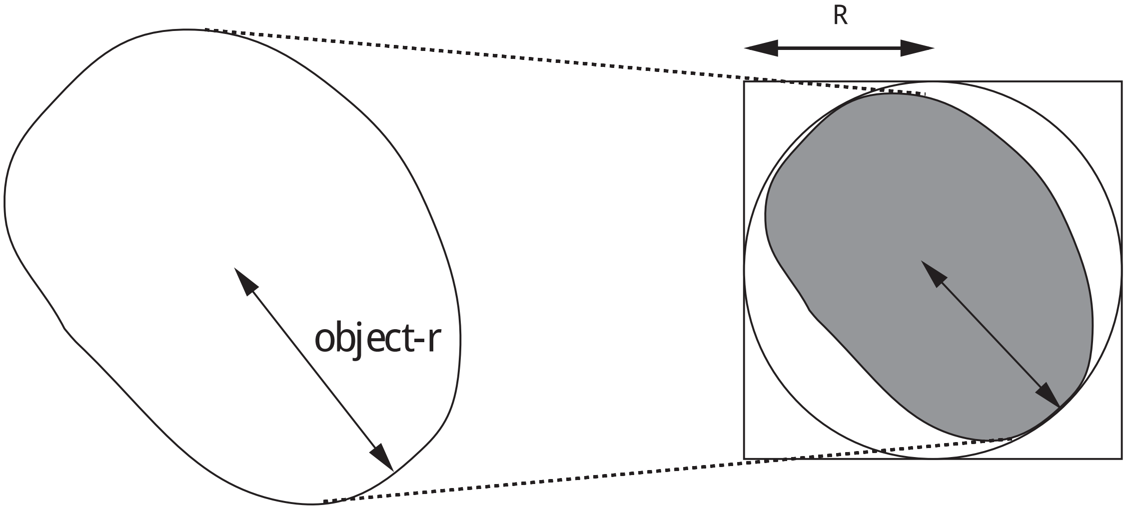


図 1.1 オブジェクトサイズの変更

- オブジェクト画像の回転

  抽出されたオブジェクトデータをテンプレートとマッチングさせるために、画像の回転を行う。画像の回転は以下の行列計算を行うだけなのでプログラムは割愛する。

  回転の計算 (回転の逆行列より)

$$\begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} i' * cos(\theta) - j\ ' * sin(\theta) \\ i' * sin(\theta) + j\ ' * cos(\theta) \end{pmatrix}$$

$$RotateFrame[i'][j'] = ScaleFrame[i][j]$$

  「ピクセル (i', j') の回転前座標 (i, j)」を求め、回転前座標 (i, j) の色情報から回転後座標である (i', j') の値を変更していく。

B. 各システムの仕様

B.1 RWI B21 モバイルロボットの仕様

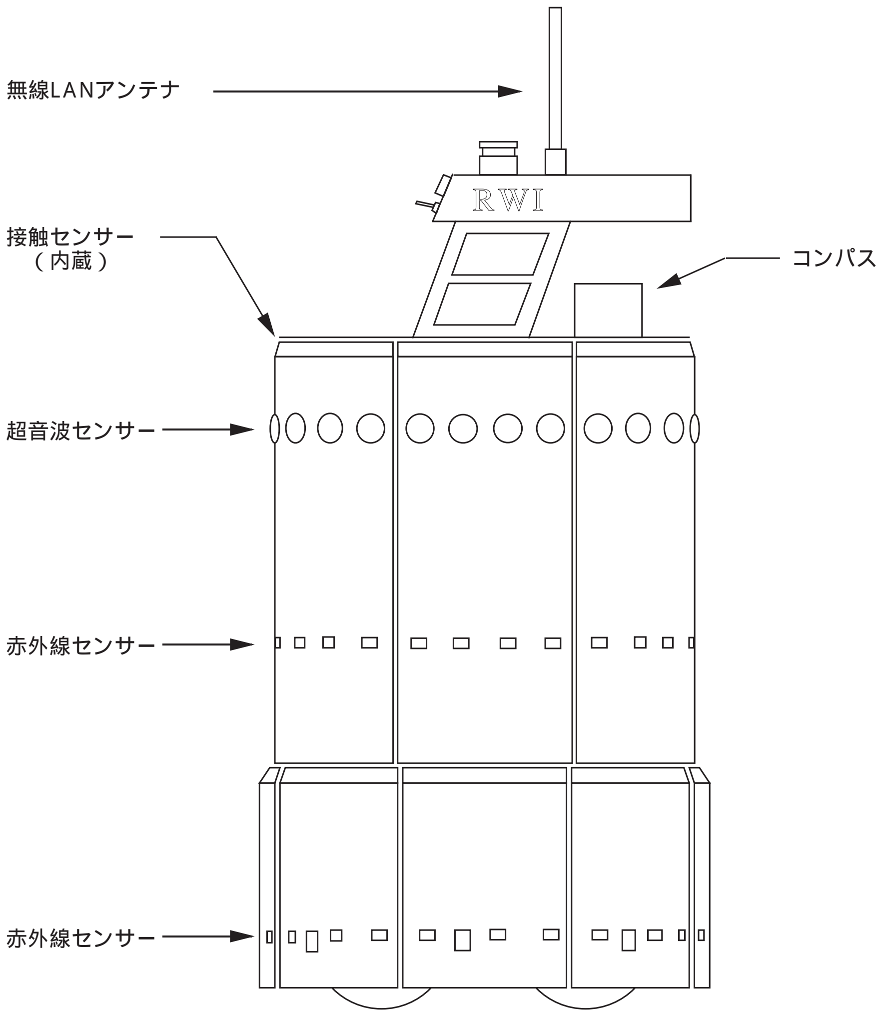


図 2.2 B21 センサー群

表 2.1 B21 の仕様

赤外線センサー	56 個（ベース部分：32 個うち 8 個は下向き、エンクロージャー部：2 4 個）
超音波センサー	24 個（12cm ~ 8m まで有効）
衝突センサー	本体の周囲に 56 個
CPU	Pentium Pro 200Mhz × 4(DOS/V 互換 Dual Pentium マシン x 2 台）
通信	High speed serial / Ethernet
ネットワーク	Onboard 10baseT and 10base2
バッテリー	4 バッテリー、1440W-hr total
バッテリー持続時間	6 時間
モーター	4 high torque,48V DC servo motors
ドライブ	4-wheel synchronous
ステアリング	4-wheel steering
Translate Speed	90 cm/s
Rotate Speed	167 deg/s
Rotate Resolution	.35°
Translate Resolution	1 mm
直径	52.5 cm
高さ	106 cm
車輪直径	11.186 cm
床面からの高さ	2.54 cm
重さ	122.5 kg

表 2.2 SONY EVI-D30 の主な仕様

システム

映像信号	NTSC カラー EIAJ 標準方式
映像素子	1/3 インチカラー CCD
	(総画素数 約 41 万画素)
	(有効画素数 約 38 万画素)
レンズ	電動 12 倍ズームレンズ
	f=5.4 ~ 64.8mm
	F1.8 ~ F2.7
	水平画角 4.4 °~ 48.8 °
最至近撮影距離	10mm(WIDE 端)
	8000mm(TELE 端)
最低被写照度	7 ルクス (F1.8)/50 IRE にて
被写照度範囲	7 ~ 100,000 ルクス
シャッタースピード	1/60 ~ 1/10,000(VISCA コントロール時)
ゲイン切り換え	自動/マニュアル
水平解像度	460TV 本
映像 S/N	48dB
パン・チルト機能	水平 ±100 °、垂直 ±25 °

入出力端子

映像出力	ピンジャック (x1)
	1Vpp、75W 不平衡、同期負
S 映像出力	4 ピンミニ DIN(1)
音声出力	ピンジャック (モノラル)(x1)
	規定出力 327mV、
	出力インピーダンス 2.2kΩ 以下
コントロール	RS232C(各 1)、8 ピンミニ DIN、
入/出力端子	9600bps、データ 8 ビット、
	ストップ 1 ビット
マイク入力端子	ミニジャック (モノラル)(1)、
	規定入力 0.775mV、
	低インピーダンスマイク用 DC3V、
	入力インピーダンス 10kΩ 以上 (ϕ3.5)
電源端子	EIAJ type4