2000年度 修士論文

# 動的なカメラ制御を用いた物体間の定性的な位置関係の把握

名古屋大学人間情報学研究科物質生命情報学専攻
学籍番号 319901270
菅野 裕

指導教官：渡辺 崇 教授

2001年1月31日提出

# 目 次

# 第1章　はじめに

自然言語表現を受けて注目点を切り替えるカメラワークが求められるスタジオ撮影や遠隔講義, 自律型ロボットといったシーンでは, ユーザとの定性的なビジュアルコミュニケーションの結果として映像を抽出する機能が求められる.

これまでにも, アクティブカメラの制御を目的に, 自然言語を用いたカメラワークの理解 [1] や, 与えられた撮影対象にふさわしいズーム率を自動的に決定する方法 [2] などが報告されている. しかし, これらでは, 自然言語の持つ曖昧性と空間キャプチャのための定量的変換の方法が十分に検討されていなかったり, 取り込まれる空間情報が理解されてカメラの行動に反映されているわけではなかった.

映像の内容の理解としては, 宮崎ら [3] が, 教室での講義を対象に, 5 つの状況を捉えて, それぞれの状況において各ユーザがみたい映像を調停の形で決定する方法を提案しているが, カメラの動きは予め規定されたユーザの映像化ルールに基づいている.

また, 人間との対話のために, 画像の中に存在する文字情報を切り出し, 理解するための方法 [4] [5] も検討されてきた.

人間の用いる自然言語表現の多くは定性的である. 特に空間情報を扱う際にそれは顕著である. それは人間の心的空間が定性的であることに起因すると言われている. 自分から遠く離れて前に見えるものも, 比較的近く, 前方に見えるものも同様に前にある, と我々は考える.

人間同士のビジュアルコミュニケーションでは, 互いに定性的な言語表現で意思疎通を行うことができる. このことは, 互いの曖昧な心的空間の情報を交換するのに定性的な表現で十分であり, 妥当であることを意味する. 計算機や自律型ロボットにおいても, 人間が命令し制御する対象であるという点から, 定量的な表現よりも, 定性的な表現でコミュニケーションを行える方がよりよいと考える.

本研究は, 人間と計算機との空間情報の共有と円滑なコミュニケーションを目指し, 視野内あるいは視野外に存在する複数の物体を認識し, それらの間の位置関係を理解することについて検討する. 画像情報は, パンチルト機能とズーム機能を備えたカメラを用いて動的に取得する.

これまでにも, 能動的カメラを用いた物体の認識システムが構築されてきているが, そこでは主に移動物体の追跡に興味が持たれてきた [6] [7]. これに

対して本研究では, 視野内および視野外に存在する物体の配置状態を把握し, 必要に応じて注目点を制御するためにパンチルトカメラを用いる.

ユーザはカメラと空間を共有し, またカメラからの動画像をもとに定性的表現をもって命令を下す. そして, ユーザが言及する物体を同定することで, ユーザと計算機が空間を共通して理解できる環境を構成する. このような理解は, 話者の話の内容に応じて撮影対象を切り替えるカメラワークの自動化などに応用可能である.

以下, 第 2 章では, 視覚情報を用いる際に使われる定性的な表現について, 物体間の位置関係, 物体の特徴とに分けて述べる. 第 3 章では, 人間の定性的な表現による入力を定量的に評価する手法について述べる. 第 4 章では, ユーザの要望に対して, いくつかの定性的な表現をまとめ, 最適な物体を選出する手法について述べる. 第 5 章では, 本システムの構成を, ハードウェアや実験環境, ソフトウェア構成について述べる. また, ユーザの入力に用いる表現について述べる. 第 6 章では, パンチルトカメラを用いて実際に本手法を実空間に適用し, その有用性を検証する. 実験には被験者を用い, 静止画像に対して人間と本手法の理解を比較した. また本システムを動作してもらい, その有用性を調べた. 最後に, 第 7 章において本研究を総括するとともに, 今後の課題について言及する.

# 第2章　ビジュアルコミュニケーションにおける定性的空間表現

本章では, 視覚情報を用いる際に使われる定性的な表現について, 物体の位置関係, 物体の特徴とに分けて述べる.

## 2.1　物体の位置関係

我々は, 視覚情報を用いてコミュニケーションをとる際, しばしば物体の位置関係を利用して話す. それは視野や閉じた空間の中での絶対的な位置であったり, 2, 3 の物体間の位置関係であったりする.

図 2.1 では, B は A に対して, どの方向にあると言えるだろうか. B は A に対して右にあるとも言えるし, 上にあるとも言える. 右上にある, と言うのが最も適当ではあるが, 新語を設け, それに合わせて境界を設定しなおすことに意味はない. 特筆すべきは, 我々は B を, A に対して右にある物体としても, 上にある物体としても認知できるということである.

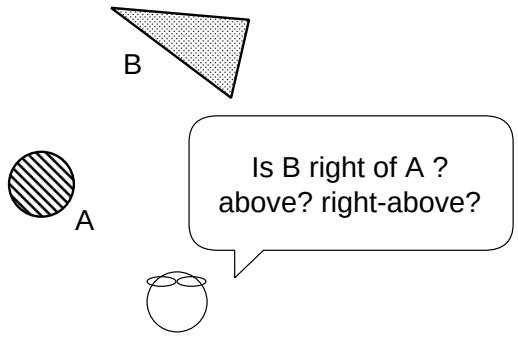



図 2.1: orientation of objects

図 2.2 では, B も C も共に A に対して右にあると言うことができる. しかし, 「A の右の物体を見よ」と言われたら我々は迷わず B を注目するだろう. 定性的な表現にはこのように対象となる候補が複数考えられる際は, より相

対的に適当な対象を指し示す性質をもつ.

また, 単に「右の物体を見よ」と言われても我々は B に注目する. ここでも「右」は, 複数考えられる候補から相対的に適当な唯一の対象を指し示している.

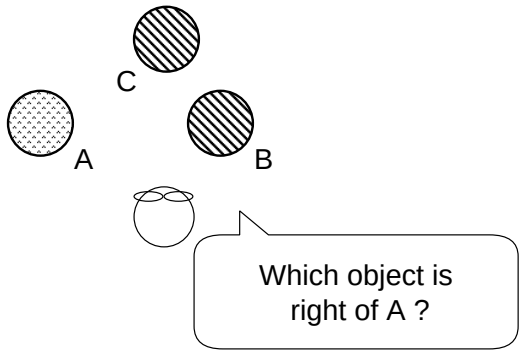



図 2.2: relation of objects

図 2.3 のように, 位置関係をつくる相手となる物体が特にない場合, 視野や部屋の全体に対しての位置を指し示す. 図 2.3 の A はグレーの領域に対して, 右にあると言える.

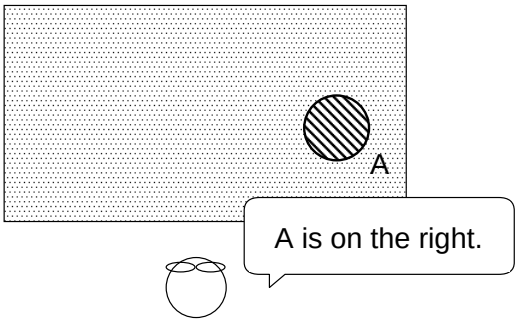



図 2.3: orientation of object in room

以上のことをまとめると, 位置関係の定性的な表現は次のような特徴を持つ.

- 空間, 平面をいくつかの領域に分割する明確な境界を持たない
- 候補となる複数の対象の中から, 相対的に最も適当な対象を指し示す
- 位置関係をつくる相手となる物体が特にない場合は視野や地の全体に対しての位置を指し示す

## 2.2 物体の特徴

視覚情報は膨大で雑多であり, 全てを記述することは難しい. 我々はその中から必要な情報を抽出し, それらを交換してコミュニケーションを行う. その際に取り出された情報は不完全ではあるが, コミュニケーションに十分な量である. それらの個々の物体を記述する表現の多くは定性的である.

我々は物体を記述するのに以下のものを用いる.

- 色
  色は色相, 彩度, 明度に連続的に分布しており, 明確な分かれ目を持たない. 我々は赤, 青, 緑, 黄などの代表的な色を中心に, それに近い色で近似して表現する. これは, その物体の絶対的な色を評価しているのではなく, 周りの物体と比較して相対的により近いことを示していることで可能となる. もし同じような色の物体が隣接している際は, より詳細な, より近似を許さない色の表現が用いられるだろう. また, 実際の物体は単色であることは稀で, 複数の色が用いられている場合が多いが, 我々はそれも代表的な一つの色で表現することができる. このように, ある物体を他の物体から区別し得る唯一の色を抽出する表現法を, 定性的な色の表現とみなすことができる.

- 大きさ
  物体の大きさを表す表現は少なく, ほとんど「大きい」, 「小さい」だけであろう. どちらも, 他の物体に対して, または空間や領域の全体に対しての相対的な大きさを意味し, 明確な境界をもたない定性的な表現である.

- 形状
  物体を指し示す際, 物体が固有の名称を持たない時や, その名称に共通の理解を持たない場合に, 我々は幾何学的な形状を用いて表現する. 物体の形状に関しては一概に定性的と言うのには語弊があるが, やや楕円気味であっても円と呼んだり, 正三角形も直角三角形も同様に三角形と呼べる様は, 状況に依存する相対的な曖昧さを持っていると言える.

また, 直接個々の物体の特徴を表すものではないが, ある状況において個物体が保有する情報として, 以下のものも物体の特徴として認める.

- 距離
  一つに, 物体間の距離の近さを表す「遠い」, 「近い」といった表現がある. これは明確の境界を持たず, 他物体との相対的な距離であり, 定性的な量である. それとは別に, 人間やカメラの視野や空間を考える際には, 物体がその領域の内にあるか外にあるかということも, 物体のもつ距離と考えられる.

# 第3章　定性的表現の定量的評価法

本章では, 2 章で挙げた人間の定性的な表現による入力に対して, 何らかの行動を決定する場合に必要となる定量的な評価手法について考える. コミュニケーションに用いる表現として, 物体間の位置関係と, 色, 形状, 大きさ, 物体間の距離で表す物体の特徴量について考える. また, ここで用いられる各動作制御パラメータについて説明する.

## 3.1　物体間の位置関係

ユーザとカメラは空間を共有しており, 同一の方向を向いている状況を想定する.

物体の相対的な位置を表す「〜の右」「〜の前」といった定性的な表現は離散的に表せない曖昧さを持つ.

ある向きに対して, その向きらしさを定量的に表したものを向き指向性 ($D^{orientation}$) と呼ぶことにする. 右, 左などそれぞれの向きに対して, 右指向性 ($D^{right}$), 左指向性 ($D^{left}$) と呼ぶ.

以下, 代表して右指向性について考える.

図 3.1 での, B の A に対する「とても右らしい」, 「あまり右らしくない」という量である右指向性 ($D^{right}_{B\to A}$) をシグモイド関数

$$f(u) = \frac{1}{1 + e^{(-u/u_0)}} \tag{3.1}$$

を用いて次式で定義する.

$$D^{right}_{B\to A} = \begin{cases} 1 - \dfrac{1}{(1 + e^{a(-|\theta| + \pi/4)})} & |\theta| \leq \pi/2 \text{ のとき} \\ 0 & |\theta| > \pi/2 \text{ のとき} \end{cases} \tag{3.2}$$

$a$ は曲線の勾配の緩急を定める定数, $\theta$ は B の A に対する角度 (図 3.1) である. B, A, それぞれ物体の重心を用いることにする.

これは $0 \leq D^{right} < 1$ の値をとり, 1 に近い方がより右らしいことを示す.

図 3.2 は $(x, y)$ 平面原点に対して $D^{right}$ を図式化したものである. ここで, $a = 8.0$ である.

同様に左, 上, 下などの他の向き指向性も (3.2) 式と同様に求められる.

また, 参照するものは物体ばかりとは限らない. 例えば, 部屋などの領域に対しての向きの場合もある. その際も, 参照する点を領域の中心におくことで同様に領域に対する向き指向性も求めることができる.

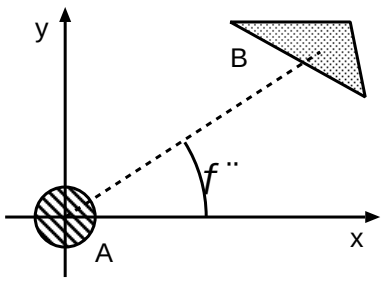



図 3.1: orientation of object

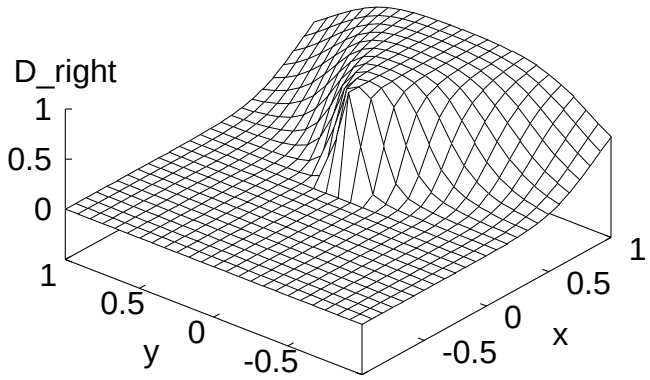



図 3.2: $D_{right}$("right"directivity) in 2-Dimension

## 3.2 物体の特徴量

ある物体が, ユーザの期待する物体とどれだけ特徴が類似しているかを表す物体の特徴量 $A$ を, 以下の要素 $a_{1\sim4}$ を用いて表す.

- 色指向性 $a_1$

- 形状整合度 $a_2$
- 大きさ $a_3$
- 距離 $a_4$

これら $a_n$ のそれぞれについては後で詳細を述べる.
$a_n$ の要素を全て $0 \sim 1$ で規格化し, 物体 $i$ の特徴量 $A_i$ を次式で定義する.

$$A_i = \sum_{k=1}^{4} w_k a_k \tag{3.3}$$

$w_k$ は重みで, 条件に応じて適当な値をあてる.

ユーザの用いた情報が不完全な場合, すなわち物体の特徴を述べるのに色, 形状, 大きさ, 距離のいくらかでも欠ける場合は, 欠ける要素にかける重み $w_k$ を 0 または小さくする. もしユーザが「近くの大きな赤い物体を見よ」と要求したのなら, 形状には触れてないため, $a_2$ にかける $w_2$ を 0 にすればよい.

また, 重みは精度が高く信用できる要素を強く反映させ, 精度が低くあまり信用ができない要素を弱くすることができる. 現状では, 色指向性 $a_1$ に比べて形状整合度 $a_2$ が期待通りの結果を出さない. その場合は, 特徴量 $A_i$ が期待通りの結果を出すまで, $a_2$ を小さくすることで解決できる. 精度の低いアルゴリズムは計算機, アルゴリズムの進歩により改善することを期待し, そのつど現行のアルゴリズムと交換し, 対応する重み $w_k$ の値を経験的に大きくすることで, 特徴量 $A_i$ をユーザの期待に近づかせることができる. $a_n$ の制約は, $0 \sim 1$ に規格化することだけである.

以下, $a_n$ のそれぞれについて説明する.

### 3.2.1 色指向性 $a_1$

色についての定量的評価を与える量. ユーザの望む色とどれだけ近いかを表す. 物体の RGB 値を事前に用意した標本の RGB 値との距離を出して算出する.

ユーザの望む色 $color$ の標本の RGB 値を $(R, G, B)$ と, 調べる物体 $i$ の RGB 値を $(r, g, b)$ とする. ここで, 調べる物体の色は, 物体全体の各画素を平均の値を用いる.

これに対して, その物体の $color$ に対する色指向性 $a_{1i}^{color}$ を次式で定義する.

$$a_{1i}^{color} = 1 - \frac{(R-r)^2 + (G-g)^2 + (B-b)^2}{R_{max}^2 + G_{max}^2 + B_{max}^2} \tag{3.4}$$

ただし,

$$Rmax = \begin{cases} R & R > depth/2 \text{ のとき} \\ depth - R & R \leq depth/2 \text{ のとき} \end{cases}$$

$$Gmax = \begin{cases} G & G > depth/2 \text{ のとき} \\ depth - G & G \leq depth/2 \text{ のとき} \end{cases}$$

$$Bmax = \begin{cases} B & B > depth/2 \text{ のとき} \\ depth - B & B \leq depth/2 \text{ のとき} \end{cases}$$

$depth$ は階調の最大値, すなわち $R, G, B$ それぞれがとり得る最大値である.

$0 \leq a_{1i}^{color} \leq 1$ の値をとり, この値が大きい方が, 物体の色がユーザの望む色の標本に近いことを示す. 一致したとき 1 になる.

一例を示す.

ユーザは「赤色」の物体を望んでいて, 物体 A, B2 つの赤色度を知りたい. 物体 A, B それぞれの RGB 値は 256 階調で $(150, 70, 40)$, $(80, 90, 120)$ だったとする「赤」は事前に用意してある標本で $(255, 0, 0)$ とわかっている. 物体 A, B の赤色度 $a_{1A}^{red}$, $a_{1B}^{red}$ は,

$$\begin{aligned} a_{1A}^{red} &= 1 - \frac{(255-155)^2 + (0-70)^2 + (0-40)^2}{255^2 + (0-255)^2 + (0-255)^2} \\ &\fallingdotseq 0.915 \end{aligned}$$

$$\begin{aligned} a_{1B}^{red} &= 1 - \frac{(255-80)^2 + (0-90)^2 + (0-120)^2}{255^2 + (0-255)^2 + (0-255)^2} \\ &\fallingdotseq 0.572 \end{aligned}$$

となる. $a_{1A}^{red} > a_{1B}^{red}$ より, A の方が B より「赤らしい」ということが言える.

### 3.2.2 形状整合度 $a_2$

形状についての定量的評価を与える量. ユーザの望む形状とどれだけ近いかを表す.

円らしさについては, 複雑度により評価する. 物体 $i$ の円に対する形状整合度 $a_{2i}^{circle}$ を, 複雑度 $e$

$$e = (\text{周囲長})^2/(\text{面積}) \tag{3.5}$$

を規格化した次式で定義する.

$$a_{2i}^{circle} = e/4\pi \tag{3.6}$$

$0 < a_{2i}^{circle} \leq 1$ の値をとり, この値が大きい方が円に近いことを示す.

3 角形らしさ, 4 角形らしさなどの多角形らしさについてはハフ変換 [8] による線分抽出により辺の数を評価する. ユーザの望む形状が $n$ 角形であるとする. 調べる物体 $i$ の形状が基本的な凸の多角形である場合, 辺の数が物体 $i$ の角数に一致する. 物体 $i$ の輪郭線にハフ変換を行い, 線分を抽出する. 経験的に与えられた適当な閾値を越えて強く抽出された線分の数が $m$ 本だったとき, 物体 $i$ の $n$ 角形に対する形状整合度 $a_{2i}^{n}$ を以下で定義する.

$$a_{2i}^{n} = \frac{n}{|m-n|+n} \tag{3.7}$$

$0 \leq a_{2i}^{n} \leq 1$ の値をとり, この値が大きい方が, 物体の形状がユーザの望む形状に近いことを示す. 一致したとき 1 になる. ただし, 物体 $i$ の複雑度が閾値より大きいときは $i$ を円とみなし, 線分抽出を行わない. その際, 円とみなした $i$ を 10 角形と近似し, 同様に計算することにする. 閾値は経験的に 0.8 と定めた.

その他の形状については, 本研究では対象としない.

### 3.2.3 大きさ $a_3$

他物体との相対的な大きさ. 面積を用いる.

視野内のすべての物体の中の, 最大のものの面積を $S_{Max}$ とする. ある物体 $i$ の面積が $S_i$ のとき, $i$ の大きさ $a_{3i}$ を以下で定義する.

$$a_{3i} = S_i / S_{Max} \tag{3.8}$$

$0 < a_{3i} \leq 1$ の値をとり, 1 に近い方が大きい物体であることを示す.

### 3.2.4 距離 $a_4$

対象物体が視野内にあるかの判定と, 視野内での物体間の距離の定量的評価を与える量. 物体の重心間の距離を用いる.

ユーザは実空間と同時に, モニタに映るカメラからの動画像を見て要求するため, その時にモニタに映っている物体をより話題に出しやすいと考える. よって, $a_4$ は視野の中にある物体の方が視野の外にある物体より大きくなる必要がある.

モニタに映っている範囲の物体の間の位置関係に言及して要求を出す際は, より参照する物体に近い物体を要求している可能性が高い, と考える. それはユーザが「～ の近く」という表現を使ったときにより顕著である.

以上のことを踏まえると, 距離 $a_4$ は視野に入っている物体に高く, 更に視野の中に入っている物体に関して, 参照する物体から近い物体が大きい値であることが望ましい.

$(x_j, y_j)$ にある物体 $j$ に対して, $(x_i, y_i)$ にある物体 $i$ の距離 $a_{4i}^j$ を向き指向性と同様にシグモイド関数 (3.1 式) を用いて以下で定義する.

$$a_{4i}^j = \begin{cases} p(1 - \dfrac{1}{(1 + e^{q(1-((x_i-x_j)^2+(y_i-y_j)^2)/S_j)})}) + (1-p)) \\ \qquad\qquad (x_i, y_i) \text{ が視野の中にあるとき} \\ 0 \qquad\qquad (x_i, y_i) \text{ が視野の外にあるとき} \end{cases} \tag{3.9}$$

$p$ は $0 < p \leq 1$, $q$ は曲線の勾配の緩急を定める定数, $S_j$ は $j$ の面積であり, 参照する物体が大きいほど, それに近いとされる領域が大きくなる. $(x_i, y_i)$ が視野の中にあるときの $a_{4i}^j$ は $1 - p \leq a_{4i}^j < 1$ である.

ユーザが他物体を参照しなかったときの物体 $i$ の距離 $a_{4i}$ は,

$$a_{4i} = \begin{cases} 1 - p \qquad (x_i, y_i) \text{ が視野の中にあるとき} \\ 0 \qquad (x_i, y_i) \text{ が視野の外にあるとき} \end{cases} \tag{3.10}$$

とする.

図 3.3 は $p = 0.7$, $q = 1.0$, $40 \times 40$ の大きさの視野の中心 $(0, 0)$ に対する $a_4$ を図式化したものである.

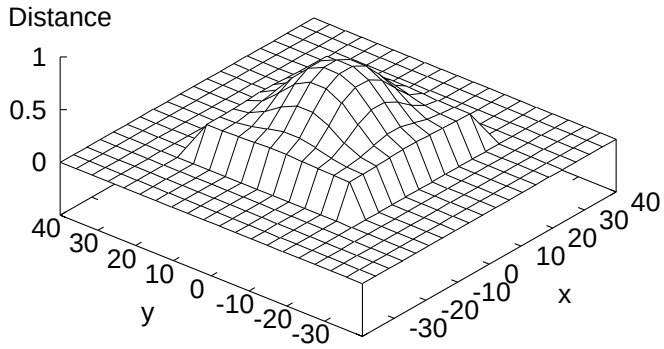



図 3.3: distance from $(0, 0)$

## 3.3 動作制御パラメータ

本章で使われた各動作制御パラメータを挙げる.

| | 定数名 | 値 | 役割 |
|---|---|---|---|
| (3.2) 式 | $a$ | 8.0 | 向き指向性の曲線の勾配の緩急を定める |
| | | | 大きいと急に, 小さいと緩やかになる |
| (3.9) 式 | $q$ | 1.0 | 距離の曲線の勾配の緩急を定める |
| | | | 大きいと急に, 小さいと緩やかになる |
| (3.9) 式 | $p$ | 0.3 | 視野内の物体の持つ距離の初期値 |
| | | | 視野外の物体よりもこの値だけ距離値が大きくなる |
| (3.3) 式 | $w_1$ | 2.5 | 色指向性の重み |
| | | | 強く反応するよう設定 |
| (3.3) 式 | $w_2$ | 1.5 | 形状整合度の重み |
| | | | やや強く反応するよう設定 |
| (3.3) 式 | $w_3$ | 0.1 | 大きさの重み |
| | | | ほとんど反応しないように設定 |
| (3.3) 式 | $w_4$ | 1.0 | 距離の重み |
| | | | ほどほどに反応するよう設定 |

# 第4章　複数の定性的表現の統合と候補対象物体の特定

本章では, ユーザの要求に対して最適な物体を選出する手法と, 物体に注目するズーム量の決定法, カメラ制御の際のパン, チルト移動量の決定法について述べる.

## 4.1　候補対象物体の特定

### 4.1.1　既知の領域からの選出

ユーザはカメラから得られる映像をもとに, 見たい物体を要求する. ユーザの要求は 2 章で述べたように, 物体に対して位置関係と特徴を定性的な表現を用いてなされ, その定性的な表現を 3 章で述べた手法で定量的に評価する. 我々はそれを用いて視野内外の全物体からユーザの望んでいると考えられる唯一の物体を選出する.

ある物体がユーザの要求にどれだけ適合するかを示す量を適合度と呼ぶ. 適合度が最大になる物体が, ユーザの要求を満たすものだと考える. ユーザの要求に対して, 物体 $i$ の適合度を, (3.2) 式で求められる向き指向性 $D_i$ と (3.3) 式で与えられる特徴量 $A_i$ との積で定義する.

$$M_i = D_i A_i \tag{4.1}$$

ユーザの要求が, $j$ に対して $orientation$ の方向にあるもの, であるときの物体 $i$ の $j$ に対する適合度は,

$$M_{i\to j}^{orientation} = D_{i\to j}^{orientation} A_i \tag{4.2}$$

である. $j$ は必ずしも物体とは限らず, 部屋などの領域である場合もある.

$j$ 自身が自明ではなく最適な物体を選出する必要があるときの $i$ の適合度は, $j$ の適合度 $M_j$ と $i$ の $j$ に対する適合度との積で求められる.

$$M_{i\to j}^{orientation} M_j \tag{4.3}$$

これを最大にする $i$ を, ユーザの要求を満たすものだと考える.

具体的に, ユーザの要求の例を示し, それに対して, 適合度を求め, 最適な物体を選択する手法を述べる.

- 例 1: 「右にある三角形を見よ」
- 例 2: 「左の緑の円の上にある赤いものを見よ」

例 1 の場合, 「右」とはカメラの視野中心に対する向きなので,

$$M_{i\to center}^{right}$$

が適合度であり, これを最大にする $i$ を選出する.

例 2 のような 2 物体間の位置関係を含む場合は, カメラ視野中心に対する物体 $j$ の, 「左の緑の円」の適合度 $M_{j\to center}^{left}$ とその $j$ に対して「上にある赤いもの」の適合度 $M_{i\to j}^{above}$ の積

$$M_{i\to j}^{above} M_{j\to center}^{left}$$

を最大にする $i$ を選出する.

### 4.1.2 未知の領域からの選出

システムとユーザは同一の空間を共有しているが, システムがまだ見たことがなく, 物体の検出が済んでいない, システムにとって未知の領域でもユーザは見えていることが多い.

ユーザが, システムの未だ知らない領域の物体を見ることを要求した際は, 一般的に前節で述べた適合度 $M$ は小さくなる傾向がある. システムが既に知っている物体がより適合するであろう表現は使われにくいためである. そのため, システムが未知の領域から候補を選出するには, 適合度がある閾値より小さい値であれば指示される方向の領域を新たに見, そこの物体を検出してその中の物体の適合度とも比較すればよいことがわかる.

(4.2) 式の適合度を規格化した量

$$M_{i\to j}^{'orientation} = D_{i\to j}^{orientation} A_i / \sum_{k=1}^{4} w_k \tag{4.4}$$

が全ての既知の物体に対してある閾値を下回っていたら, 既知の領域には候補が無いものとして, $orientation$ の向きに視野 1 画面分向き, その領域内の物体から候補を探すことにする.

(4.3) 式と同様に $j$ も候補選出の対象である場合の,

$$M_{i\to j}^{'orientation} M_j' \tag{4.5}$$

も閾値を定めて同様の操作を行う.

実際には, (4.4) 式では経験的に閾値を求められず, (4.5) 式の場合だけ,

$$M_{i\to j}^{'orientation} M_j' < 0.3 \tag{4.6}$$

としている.

## 4.2 ズーム量の決定

ユーザがある物体に注目して欲しいとき, カメラがその物体のある方を向くだけではなく, 適当な大きさにズームさせるのが望ましい.

図 4.1 のように画面中心に物体を見ている状態で, 物体を適度にズームさせたい.

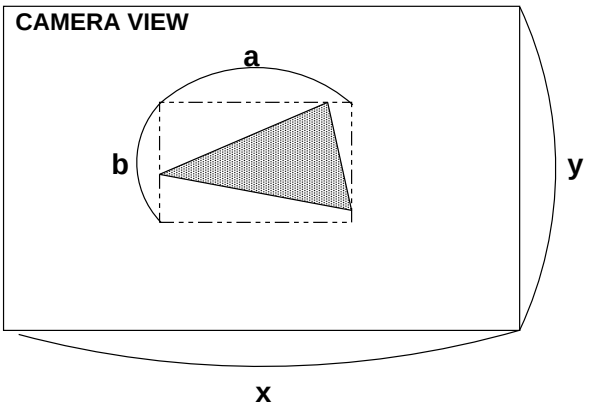



図 4.1: adjustment of zoom parameter

本システムに使用したカメラのズームは, 現在のズームパラメータの値が $zoom$ で, $l$ の長さのものを $l'$ にしたいときに, 以下の式にほぼ従っていた.

$$zoom' = zoom + 426log l'/l \tag{4.7}$$

物体に外接する長方形の各辺が図 4.1 のように $a, b$ のとき, 次のようにズーム後の期待する長さを考える.

$$\begin{cases} l' = mx, l = a & a/x > b/y \text{ のとき} \\ l' = my, l = b & a/x \leq b/y \text{ のとき} \end{cases} \tag{4.8}$$

として (4.7) 式に代入してズーム量を決定する. ただし, $m$ は $0 < m < 1$ の定数で, 画面全体のどれだけの大きさを示させるかを示す. $m = 0.5$ で画面の横, または縦の長さの半分の大きさにズームする. 本システムは経験的に $m = 0.5$ としている.

## 4.3 パン, チルト量とズーム値の相関による移動量決定法

対象とカメラとの距離や絞り値を利用しない場合、指定の物体に注目する際のパン、チルト量はその時のズーム値に依存し、カメラからの画像上での

距離のピクセル数からは特定できない。

本システムではパン、チルト量とズーム値との相関を事前に調べ、それらを用いて移動量を近似する。

パンとズーム値との相関について述べる。あるシーンにおいて、図 4.2, 4.3 のようにパン、チルトを固定し、ズーム値を変化させる。それぞれ、物体の中心を画面の中心に持っていくまでのパン量 $pan$ と画像上での距離 $d$、ズーム値を $zoom$ とすると、$zoom$ に対する $d$ と $pan$ の比は図 4.4 の折れ線のようになる。これを最小二乗法を用い、二次の多項式で近似する。図 4.4 の例では、

$$d/pan = 2.390352 * 10^{-5} zoom^2 - 1.384049 * 10^{-2} zoom + 4.361036 \quad (4.9)$$

相関係数 $|r|$ は 0.9994039 であった。この式に $zoom$ と $d$ を代入し、$pan$ を求める。

同様にチルトとズーム値との相関も求められる。

この例にある式 4.9 は本システムで使用したカメラに固有のものであり、汎用性はない。別のカメラを利用する際には同様の手法で式を算出する必要がある。

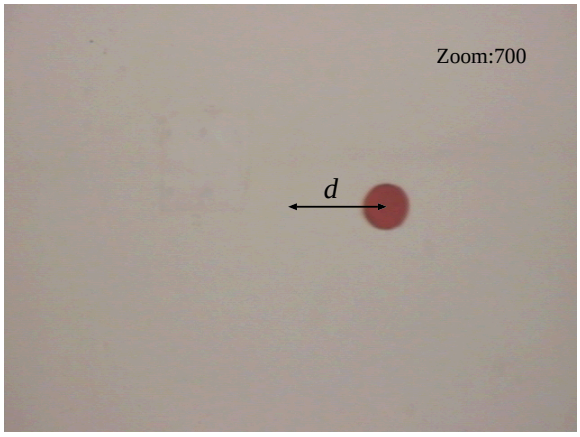



図 4.2: correlation of zoom parameter zoom: 700

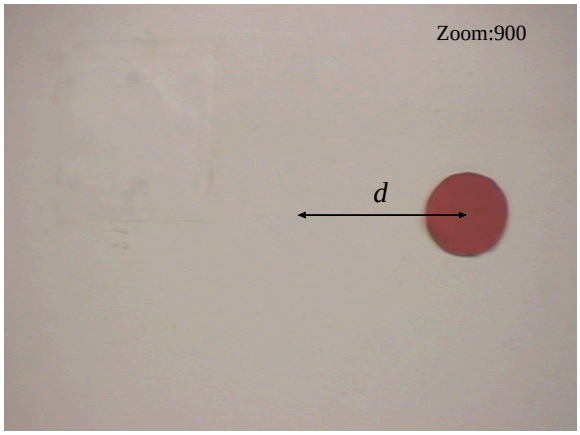



図 4.3: correlation of zoom parameter zoom: 900

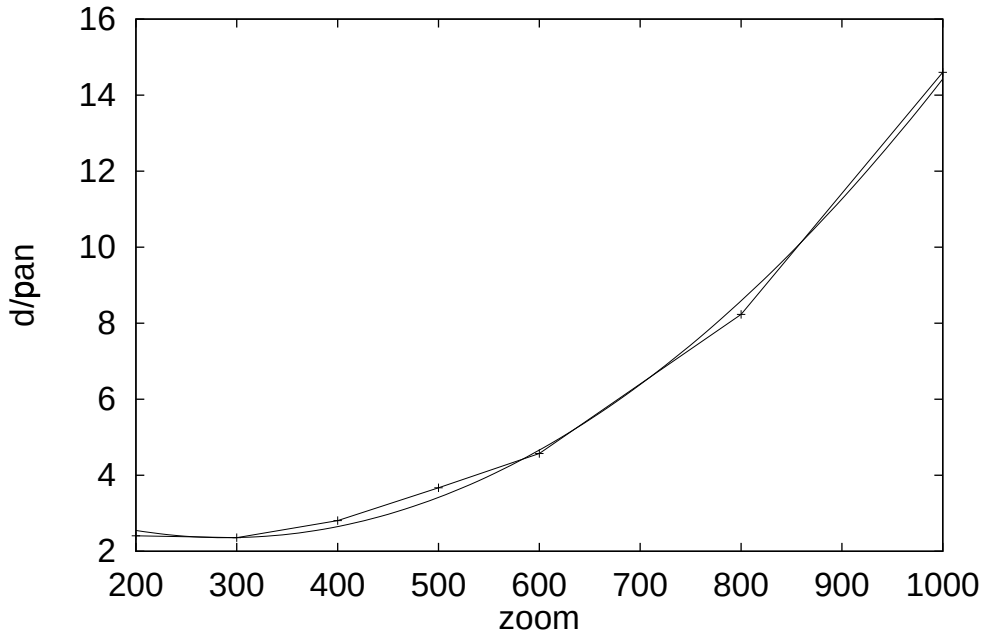



図 4.4: correlation-graph of pan-zoom parameter

# 第5章　システムの構成

本章では, 2, 3 章で提案した手法を実空間に適用した本システムの概要と, 本システムに対してユーザの利用する表現について説明する.

## 5.1　ハードウェアと実験環境

本システムは, パンチルトカメラに SONY 製 EVI-D30[9], ビデオキャプチャボードに ArgoCraft 社製のボード [10] を, OS に Linux と, グラフィックに X Window System を用いている.

図 5.1 は本システムのハードウェアの構成と実験環境の模式図である. パンチルトカメラで捕えた動画像はリアルタイムにモニタに映される. ユーザは実空間と, モニタに映される映像の両方を元にキーボードから入力する. カメラの制御は RS-232C を用いる.

特筆すべきは, パンチルトカメラとユーザが同じ向きで実空間を見ていることであり, 本システムに必須の条件である.

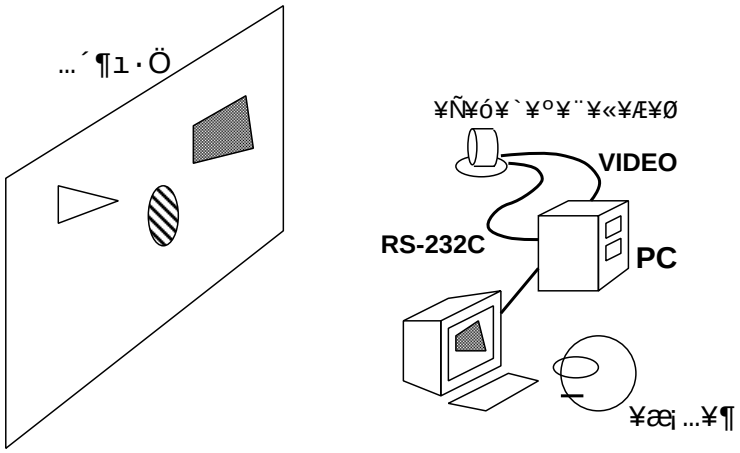



図 5.1: hardwares and environment

## 5.2 ソフトウェア構成

本システムのソフトウェア部は,

1. データベース表示モジュール
2. カメラ制御モジュール
3. 動画表示モジュール
4. インターフェイスモジュール

で構成される. 以下, 各モジュールの機能について説明する.

### 5.2.1 データベース表示モジュール

データベースを読み込み, それに従ってモニタに検出済みの物体の特徴と配置を可視化するモジュール. カメラ制御モジュールと連携しており, 新規に物体を検出した際に働く. C 言語で記述. X Window System を用いている.

ここでデータベースとは, 検出済みの物体の情報を記憶しているファイルであり, 物体それぞれの, 重心の絶対座標と, 物体に外接する長方形の左上, 右下の座標, 面積, 色の RGB 値の平均値, 複雑度, 多角形とみなしたときの角数（ハフ変換によって抽出された線分の数）が記述されたものである.

### 5.2.2 カメラ制御モジュール

インターフェイスモジュール, データベース表示モジュールを統括し, 物体の認識, 位置関係の推論, 候補対象物体の特定を行う. インターフェイスモジュールを呼び出し, 中継ファイルを介してユーザからの入力を受ける. ユーザの要求に対して, データベース中の物体の情報から 3 章, 4 章で提案した手法で評価し, カメラの動作を決定する.

カメラの制御には事前に用意されている EVI-D30 用のコマンドライブラリ [11] を利用する.

動画像表示モジュールとは中継ファイルを介して通信を行う. カメラを動かすと, 動画表示モジュールに動作後の新しい静止画像を要求する. 受け取った静止画像を 2 値化して背景から物体の抽出を行い, 新たな物体を検出したらその物体の情報を取得してデータベースに書き込む.

### 5.2.3 動画表示モジュール

カメラからの VIDEO 信号を読み込み, 動画像をモニタに映す. 他モジュールとはプロセスが独立している.

中継ファイルを介してカメラ制御モジュールと通信を行い, 要求に応じて

静止画像を渡す.

C 言語で記述. X Window System を用いている.

### 5.2.4 インターフェイスモジュール

ユーザの入力した文を単語レベルでマッチを行い, システムの理解できる命令と表現を抽出し, 適当な形に変換してカメラ制御モジュールに渡す. 入力はターミナルを用いてコマンドラインから行う.

Perl で記述. カメラ制御モジュール内から呼出され, 中継ファイルを介してデータを渡す.

図 5.2 は本システム起動時の画面の様子である. インターフェイスモジュールと, 動画表示モジュール, データベース表示モジュールによる表示部が見えている.

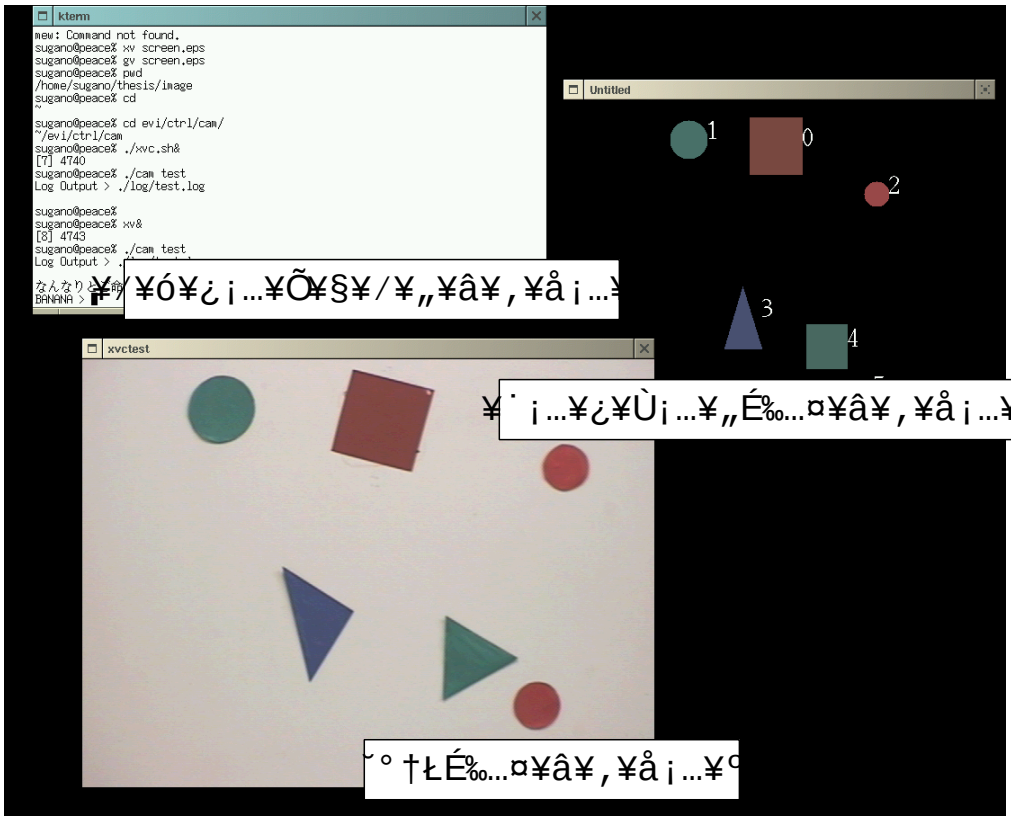



図 5.2: screen-shot of the system

## 5.3 ソフトウェア部のデータの流れ

図 5.3 はソフトウェア部の構成と, 入出力, データの流れを模式的に表したものである. これを順を追って説明する.

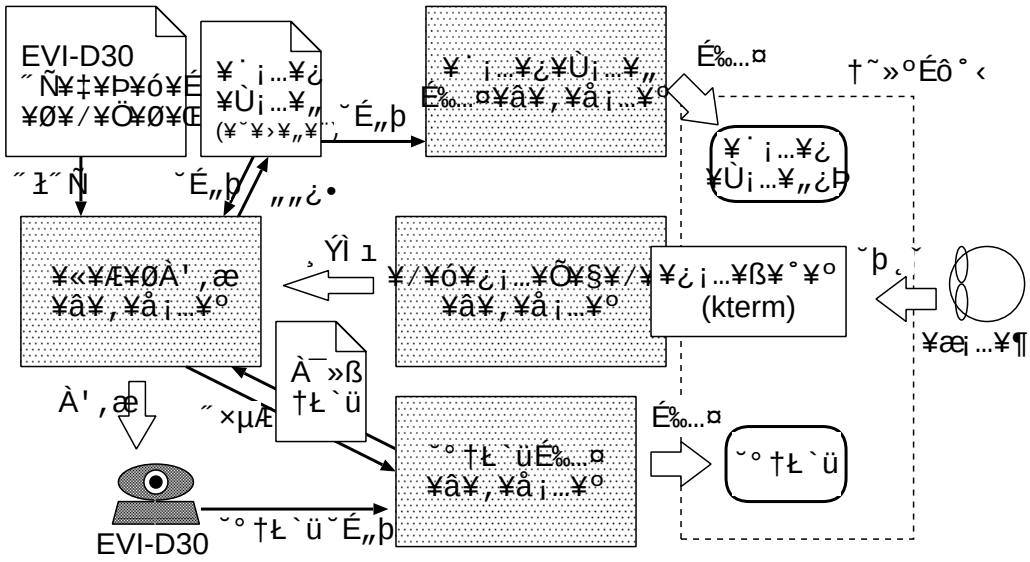



図 5.3: flow of the system

1. ユーザはモニタに映されている動画像を元に, ターミナルから命令を入力する. 動画像は, 動画像モジュールがリアルタイムにカメラから取り込んで出力している.
2. ユーザからの入力は, インターフェイスモジュールが, 理解できる表現に翻訳しカメラ制御モジュールへ渡す.
3. カメラ制御モジュールはデータベースを元に, ユーザの命令を把握し, カメラに制御コマンドを送る. 制御コマンドは, 事前に用意してある EVI-D30 用のコマンドライブラリを用いる.
4. カメラが向きを変え未知のシーンを映したら, カメラ制御モジュールは動画モジュールに静止画像を要求する.
5. カメラ制御モジュールは受け取った静止画像から物体を検出する. データベースと比較して未知の物体であればデータベースに新たに検出した物体を追加する.
6. データベースが更新されると, データベース表示モジュールがそれを図示化し, 表示する.

この流れを, データベースを逐次更新しながら繰り返す.

## 5.4 ユーザインターフェイス

5.1 節で見たように, ユーザはターミナルから命令を入力し, インターフェイスモジュールが, カメラ制御モジュールが読める表現に翻訳する. 本節では, 本システムに対してユーザの利用する表現と, その解釈について説明する.

ユーザの命令は大きく分類すると以下の 2 種類である.

1. カメラを期待する場所へ向ける
2. カメラを期待する物体に注目させる

1. は既定の場所や, 当て推量でカメラを向けたりする命令群である.

それらには以下のものがある.

| | |
|---|---|
| go home | 元の位置に戻れ |
| go back | 前の位置に戻れ |
| look(see) *direction* | *direction* の方を向け |
| zoom in, zoom out | ズームせよ, ズームひけ |
| more, again | もっと ~ して, もう一回 ~ して |
| show map | データベース図を見せて |
| quit, exit | 終了せよ |

home はシステム起動時のカメラの位置である. look(see) *direction* で向く量は, 現在モニタに映されている動画像の一画面弱分に設定した. zoom in, zoom out でのズームの量は一定の適当な量を与えた.

2. は物体を指定してその方にカメラを向けたり, フォーカスさせる命令である. 物体を指定するのにはある一つの物体だけに言及するときと, 複数の物体間の位置関係を用いることがあるのは 2 章で述べた通りである. 物体間の位置関係を用いる際は, 2 物体の間の位置関係のみ用いることができる. すなわち, ある物体を指し示すときには別の一つの物体を参照することができる.

これら場合, ユーザは以下の表現を用いる.

- 一つの物体のみ言及するとき
  look *object*　　*object* を見よ
  forcus *object*　　*object* をフォーカスせよ
- 物体間の位置関係を用いるとき
  look *object1*
  　　*direction* of *object2*　　*object2* の *direction* の方向にある *object1* を見よ
  forcus *object1*
  　　*direction* of *object2*　　*object2* の *direction* の方向にある *object1* をフォーカスせよ

*object* は色, 形, 大きさや距離を組み合わせた形で表現される.

これらの表現の文法や語彙に関しては, 付録 A で詳細に述べる.

# 第6章　検証

5.3 章で述べた本システムの実験を行い, その有用性を検証する．実験は最初に固定したシーンでの静止画像に対して行い, 次に動的にカメラを動かし, ユーザの要求を満足するように動作するかを検証した.

## 6.1　静止画像に対する検証

カメラから得た静止画像の物体に対して, ユーザの用いる表現で本システムがユーザの意図と同様な理解を示せるかを検証する．この実験では静止画像に図 6.1 を用いた.

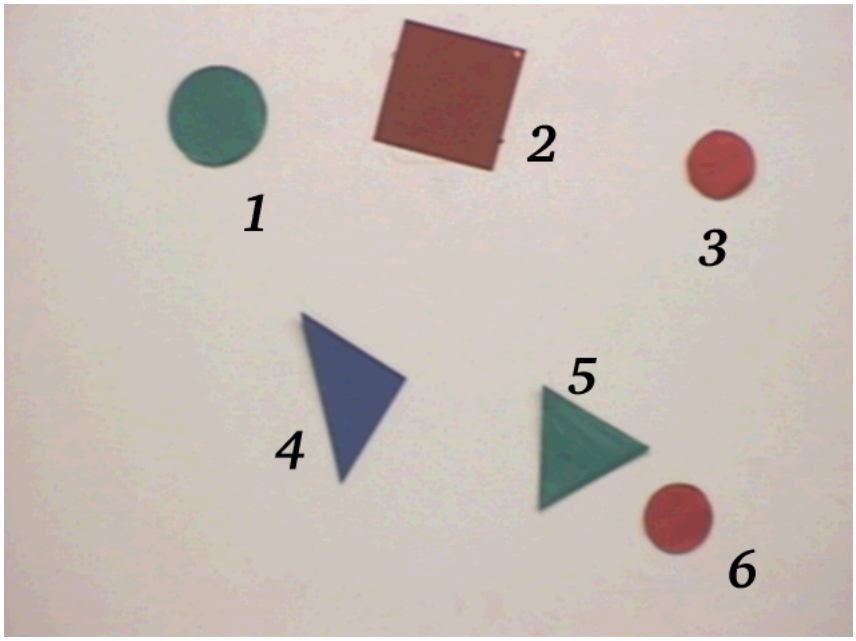



図 6.1: captured still image with numbers

### 6.1.1 人間の理解

被験者 10 人に, 図 6.1 を元に以下のアンケートに答えてもらった.

> 別紙の図に対して, あなたが他人, またはコンピュータに話している状況を考えます. 他人, またはコンピュータも同じ図を見ています.
>
> 他人, またはコンピュータに, 図のある番号の物体に注目して欲しいとき, あなたはどのように指示しますか？
>
> 「◯◯ を見よ」
> "look at *hoge*."
>
> *hoge* に入れる語句を下の語彙から作ってください. 一つだけ使っても, たくさん使っても構いません.
>
> right, left, above, below, near,
> red, green, blue, circle, rectangle(square),
> triangle, object, big, small, of
>
> 例 1：look at *green circle left of right red object*.
> 例 2：look at *left blue triangle*.
>
> 下に, 番号のある物体それぞれについて, 記述してください.
>
> 1.look at
> 2.look at
> ...

今回用いた図 6.1 からは以下のことが言える.

- 形状に注目すると, 2 だけがユニーク. あとは形状だけでは特定できない.
- 色に注目すると, 4 だけがユニーク. あとは色だけでは特定できない.
- 大きさは 2 がやや大きく, 他は判断しにくい. 3, 6 は同一である.
- 距離に注目すると, 5, 6 間が近いがあとは判断しにくい.
- 3, 6 はまったく同一の物体で, 他物体との位置関係を用いないと特定できない.

アンケートの結果を以下に示す.

| 表 1: 各要素の使用, 未使用の合計 | | | | |
|---|---|---|---|---|
| | 色 | 大きさ | 形 | 他物体との位置関係 (距離) |
| **1** | 8 | 1 | 9 | 4(0) |
| **2** | 5 | 3 | 8 | 4(1) |
| **3** | 8 | 1 | 9 | 8(1) |
| **4** | 8 | 0 | 9 | 5(0) |
| **5** | 8 | 0 | 9 | 6(3) |
| **6** | 9 | 1 | 9 | 8(3) |

| 表 2: 物体間の相互参照の合計 | | | | | | | 縦:主体 横:被参照 |
|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | **計** |
| **1** | ♠ | 4 | 0 | 0 | 0 | 0 | 4 |
| **2** | 3 | ♠ | 0 | 1 | 0 | 0 | 4 |
| **3** | 0 | 7 | ♠ | 0 | 0 | 1 | 8 |
| **4** | 0 | 3 | 0 | ♠ | 2 | 0 | 5 |
| **5** | 0 | 1 | 0 | 2 | ♠ | 3 | 6 |
| **6** | 0 | 0 | 0 | 0 | 8 | ♠ | 8 |
| **計** | 3 | 15 | 0 | 3 | 10 | 4 | 35 |

表 1 は各物体それぞれを指し示す際に用いた表現を, 要素別に 10 人分の統計をとったものである. 他物体との位置関係の括弧は, 距離 (near) を用いた人の数を表し, その残りが向きを用いた人の数を表す.

表 2 は他物体との位置関係を用いたときの, どの物体を参照したかの統計をとったものである. 縦軸の物体を指し示すのに, 横軸の物体を参照したことを意味する. 例えば, 1 を指し示すのに, 4 人の人が 2 との位置関係を用いている.

実際には, 被験者の表現だけではどの物体を参照したかは特定できないため, 筆者の判断が入っている. たとえば, *red circle* と参照した場合はこれが 3 なのか 6 なのかは断定できない. しかし, 今回用いた図では, この判断が間違ってはいないと信じても問題はないだろう.

表 1 から, 以下のことが読み取れる.

- ほぼ全ての人が形を用いている (*object* を使わない).
- 2 以外の物体には多くの人が色を用いている.
- 大きさが用いられたのはほぼ 2 のみ.
- 3, 6 は多くの人が他物体との位置関係を用いている.

- 距離が用いられたのはほぼ 5, 6 のみ.

表 2 から, 以下のことが読み取れる.

- 多くの人が, 多くの物体 ($1 \sim 5$) に対して 2 を参照している.
- 隣の物体 (1 なら 2, 5 なら 4 と 6 など) を参照しやすい.
- 3 は 2 を, 6 は 5 を参照する傾向が強い.

以上の結果から, 次のような推測ができる.

- 2 は形状, 大きさにユニークなため, 参照されやすい.
- 見かけ上合同な 3, 6 の物体はそのものの特徴では特定できないため, 他物体との位置関係を用いる傾向があるが, 必ず使うとは限らない.
- 距離の近い物体同士は *near* だけでなく位置関係でも参照しやすい.

### 6.1.2 システムの理解

図 6.1 を本システムに適用させた.
要求は 6.1.1 で被験者が用いた全て, すなわち $6 \times 10$ の命令を試した.
結果は, 60 の命令のうちユーザの期待通りに動いたものが 53 ケース. 成功率は 88%だった.
失敗したケースを 3 ケースに分類して, それぞれの実例を見ていく.

- ケース 1: ユーザの勘違い

  期待する物体: 3　　注目した物体: 1
  look at *small circle left of red rectangle*

  *red rectangle* は 2 のことだと考えられる. 3 は明らかに 2 の右手にあり, *left* はユーザの勘違いだろう. 向きに関しては望むものの反対を指示すると全く選ばれる可能性がなくなるため, この類いの勘違いは致命的である. 多少ユーザの勘違いや入力ミスを考慮できる必要を感じる.

- ケース 2: 怠慢な命令

  期待する物体: 6　　注目した物体: 3
  look at *red circle*

  *red circle* である 3, 6 の二つは見かけ上区別できない. この命令は人間に指示しても判断できかねるもので, 本システムの誤動作もやむを得ない. 一度選択に失敗しても, 2 度同じ命令を受けたら次は違う物体を選択する, などの方法で解決できるだろう.

- ケース 3: 少ない情報

  期待する物体: 5　　注目した物体: 1
  look at *object near red circle*

  実際のそれぞれの物体の適合度を比較してみると, 2 より 6 の方が *red circle* に適合していたが, それより 2 に対する 1 の距離 (near) が 6 に対する 5 のものよりも大きく差を開け, 結果 1 を候補としてしまった. 距離の重み $w_4$(3.3 式) を小さくすることで改善できる.

## 6.2　動的なカメラ制御に対する検証

被験者自身に物体を自由に配置してもらい, それに対して本システムを体験してもらった. 今回は 2 名それぞれの動作結果を順を追って説明する.

番号のあとのイタリック太字が被験者の入力した命令である. それを受けて動作した結果を左にカメラがキャプチャした画像, 右に更新後のデータベース図を並べて示す.

### 6.2.1　A 氏の場合

1. (物体の初期配置)

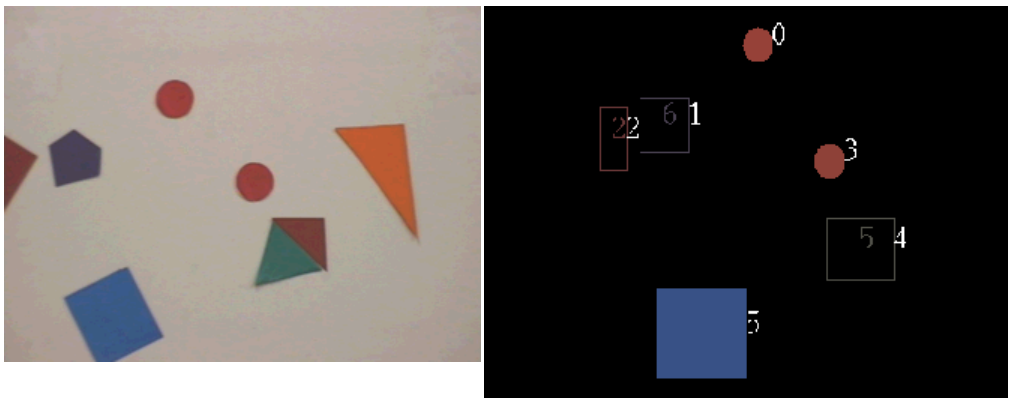



図 6.2: A1: A's default image(left) and database(right)

中央やや右下の赤と緑の三角形が重なっている物体を, まとめて 1 つの 5 角形として認識してしまっている (データベース図の 4. 以下番号のみ記述). 左手の紫の 5 角形は 6 角形 (1) に, また右手のオレンジ色の三角形は検出されていない.

2. ***look red something***

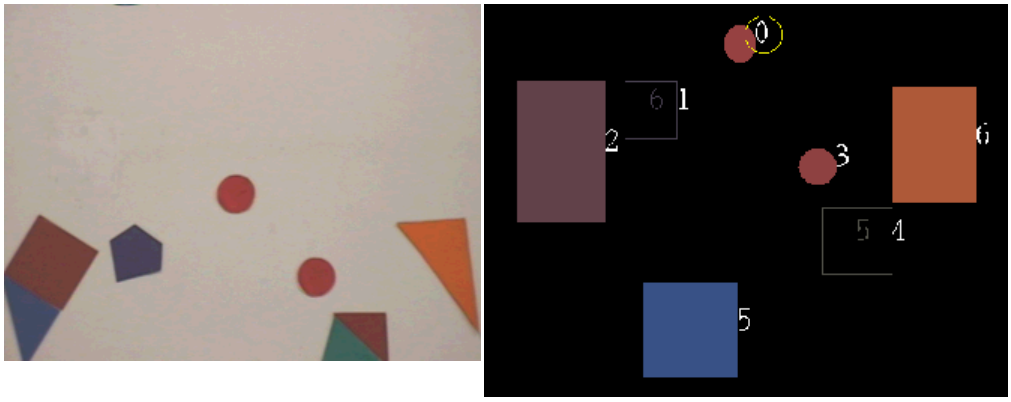



図 6.3: A2:*look red something*

赤い物体は (0),(2),(3) とあるように考えられるが, システムは (0) を見た. 彼の期待する結果だったようだ.
ここで右手のオレンジの物体を検出 (6). しかし四角形として認識している. 左手の赤い四角形と緑の三角形の重なったものも検出 (2) したが, やはり 1 つの物体として認識してしまった.

3. ***focus orange triangle***

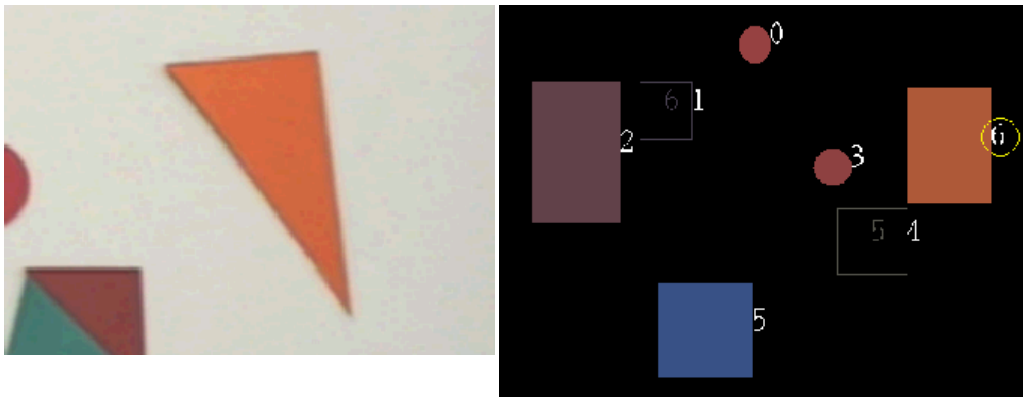



図 6.4: A3:*focus orange triangle*

彼の期待通り (6) に注目した. 実際はシステムは (6) を 4 角形と認識していたが, *orange* に強く反応し (6) を最適と判断した.

4. ***look red triangle below orange triangle***

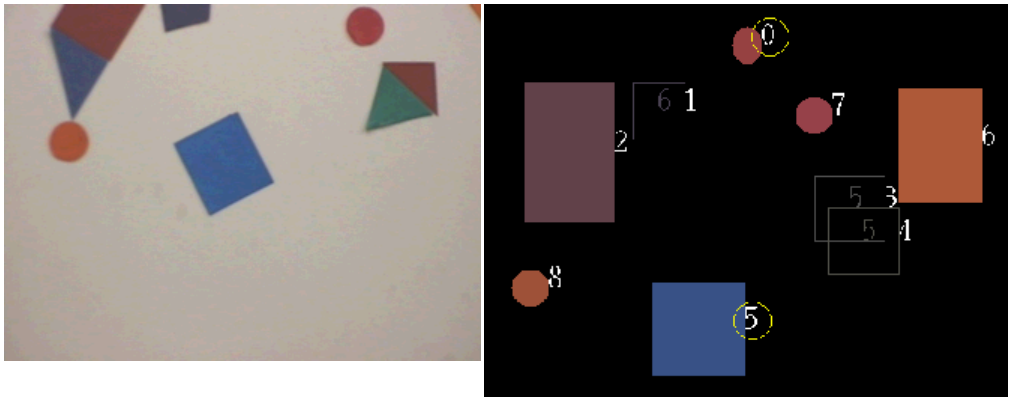



図 6.5: A4:*look red triangle below orange triangle*

彼は右手の緑の 3 角形と重なった赤い 3 角形を見て欲しかったようだが, システムはこれを (3) として 1 つに認識し, *red triangle* があることを知らない. 結果, *below* の関係に強く反応し, (0) の真下にある (5) を見てしまった. 物体の形状認識の未熟さから起きた失敗例である.

5. ***look purple something above blue square***

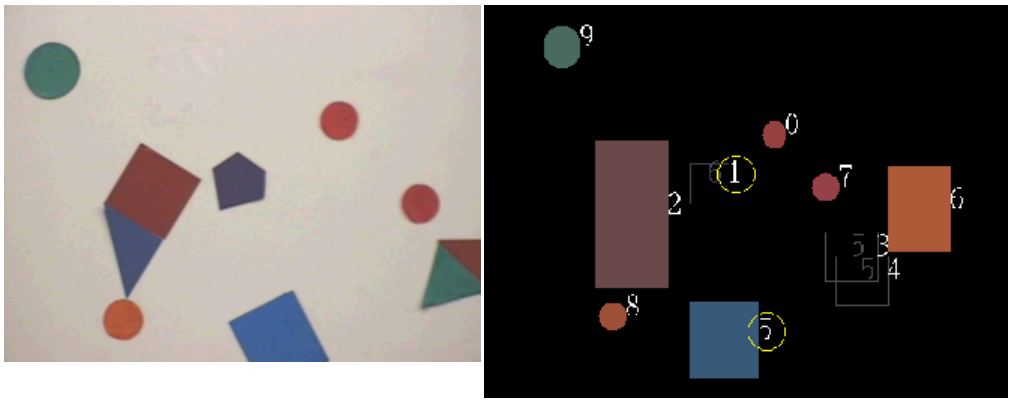



図 6.6: A5:*look purple something above blue square*

彼の期待通り (1) に注目した. 実際はシステムは (1) を 6 角形と認識していたが, *blue square* と, それとの *above* の関係から (6) を最適と判断した.

### 6.2.2 B 女史の場合

1. (物体の初期配置)

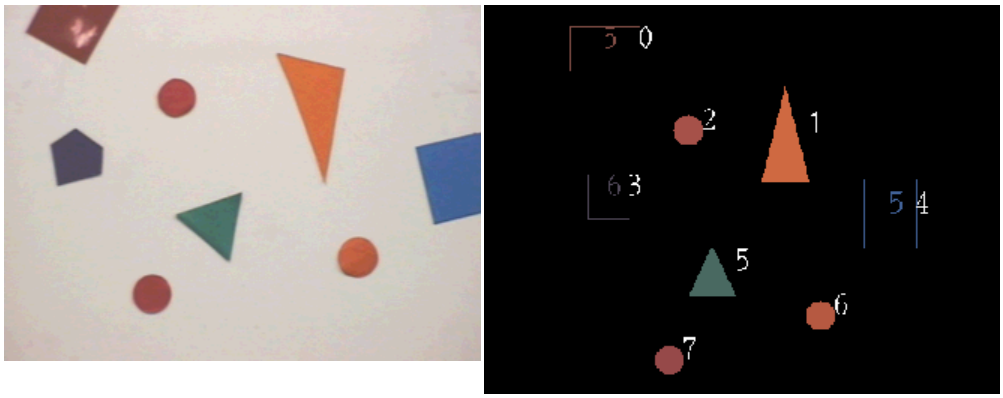



図 6.7: B1: B's default image(left) and database(right)

左上の赤い物体 (0), 右手中央の青い物体 (4) は視野からはみ出しており, 正確に形状認識できていない. 左手中央の 5 角形は 6 角形として認識している (3).

2. ***look orange object***

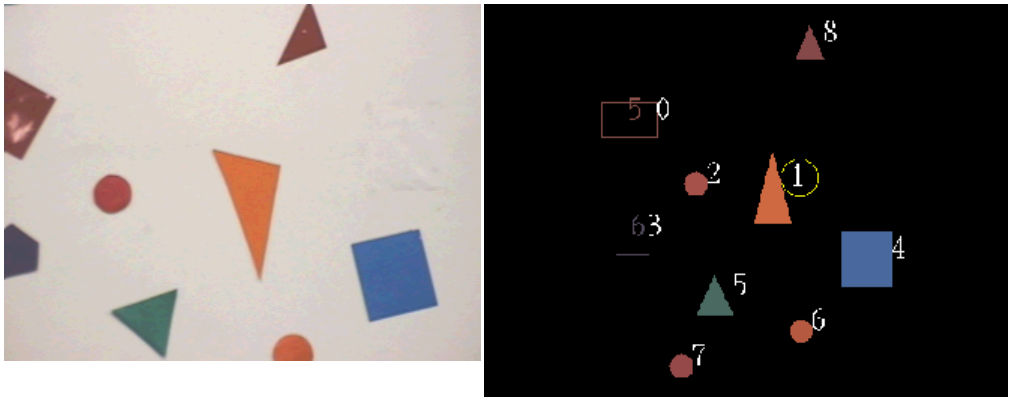



図 6.8: B2:*look orange object*

彼女の期待通りに (1) を見た. *object* と形状を指定しなかった場合だが, 色だけの情報で正確に把握できた. また情報に新たに赤い物体を検出し, 3 角形として認識した (8).

3. ***look red triangle above of orange triangle***

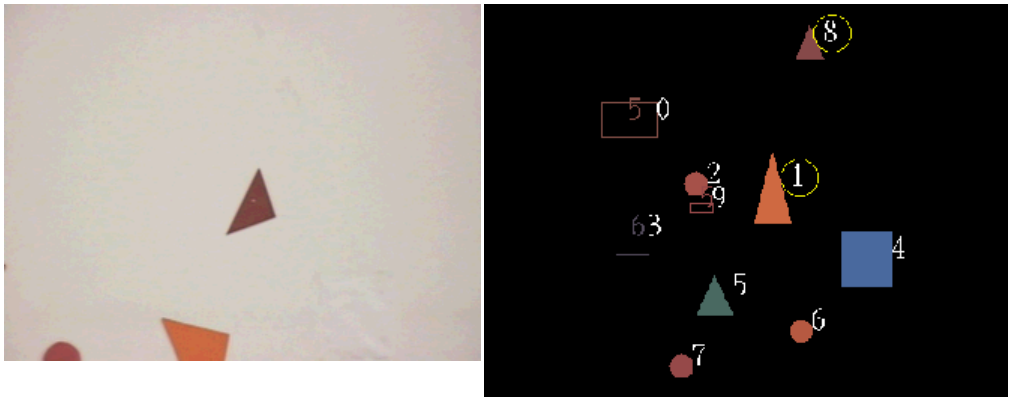



図 6.9: B3:*look red triangle above of orange triangle*

彼女の期待通り (8) を見た. (1) の *above* の位置にある物体として判断しており, 完全な理解と言えよう.

4. ***look blue object near orange object***

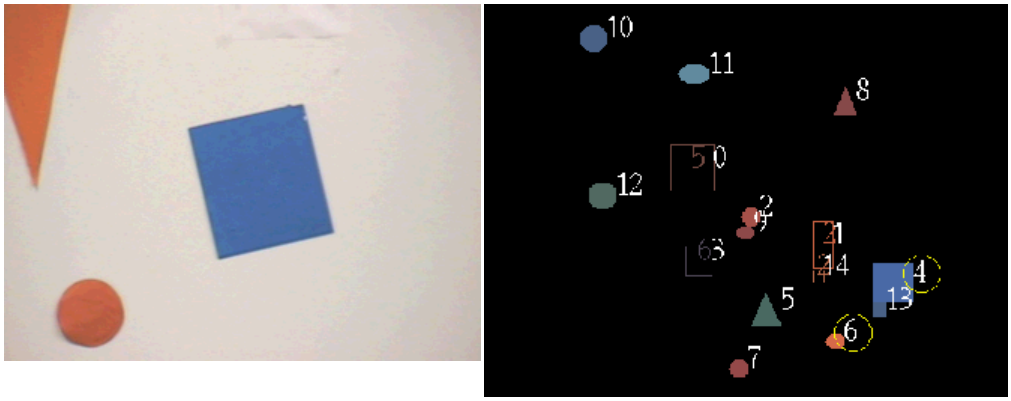



図 6.10: B4:*look blue object near orange object*

しばらく動作させた後の状態. いくつかの新しい物体がデータベースに登録されている. しかし, (9),(13),(14) は既知の物体の一部が別なものとして検出, 認識してしまったものである.
この命令では彼女は (1) の近くにあるとして (4) を見ることを望んだが, システムは (6) を *orange object* として参照し, それの近くとして (4) を選んだ. 過程に間違いはあっても結果は期待通りになりやすいということの好例.

5. ***look blue object near cyan object***

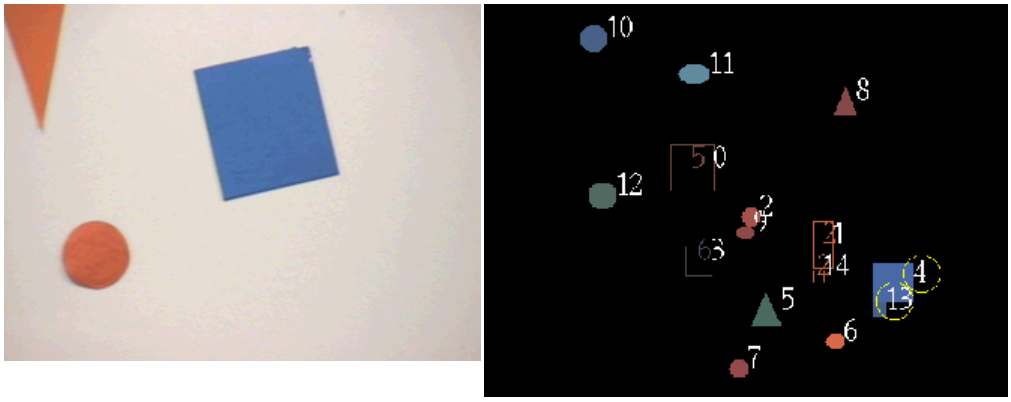



図 6.11: B5:*look blue object near cyan object*

彼女は視野外の左上にある (11) の近くの (10) を期待したが, システムは視野にあった (4) のそばに (13) としてノイズを認識してしまっていたため, (13) を見てしまった.

### 6.2.3 両氏の感想

被験者の両氏に頂いたコメントをまとめると以下のようであった.

- 位置関係を用いると期待通りに動作することが多い.
- 形状は 5 角形以外ほぼ期待通りに見てくれる.
- 大きさ, 近さの判断がよくわからない.
- 重なっている物体を認識してくれない.
- 使っているうちにノイズを拾い, 期待通りに動作しにくくなる.

# 第7章　おわりに

本研究は, 人間と計算機との空間情報の共有と円滑なコミュニケーションを目指し, 定性的な表現による位置関係の理解と, その位置関係を用い, 空間情報からユーザの望む物体を選出する手法について検討した. 物体の特徴を記述する定性的な表現として, 位置関係と, 色, 形状, 大きさ, 距離を用いた.

また, パンチルトカメラを用いて動的に実空間から画像を取り込み, 本手法を用いた空間情報の理解, 推論を行うシステムを構築した. それによってユーザの期待する動作と, 物体と物体の位置関係の認識をすることにおいて妥当性を検証した.

以下, 現状での問題点と今後の課題を述べる.

本システムは, 3.1 章で提案した位置関係を用いた表現に威力を発揮することがわかったが, 物体の形状認識で, ごく基本的な形状しか認識できない, 重なっている物体を認識できない, などと言った不十分な部分が感じられた. 形状の認識にパターンマッチなどを取り入れることで, 物体の特徴をより正確に捉えることが期待できる.

ユーザの要求する物体への指示が多いほど, 特に位置関係を用いた表現を使うほど物体の候補選出の正確さが増した. その一方で, 不完全な表現から期待通りに動作しなかった後も, 学習することなく, 同じ過ちを繰り返してしまう. 2 度目は違う答えを返すといった方法が考えられる.

また, **??**章で述べている各所に用いられたパラメータは現段階で筆者が経験的に数値を与えているが, 対象とするシーンに依存するため, 動的に値を変えることが望ましい. 教師あり学習を用いたニューラルネットなどの利用が考えられる.

今後, 人間が計算機とコミュニケーションをとる機会が増えてくることと思われる. その際には計算機が人間同等の定性的な表現を受け入れる必要が出てくるだろう. 本研究が計算機をより使いやすく, より賢くなることの一助となることを願って本文の結びとする.

# 謝辞

本研究を進めるにあたって, 多くの御指導, 御助言, および御厚意を賜わりました名古屋大学情報メディア教育センターの渡辺崇教授に心より深く感謝致します. また, 有益な御助言を賜りました人間情報学研究科の横澤肇教授, 北栄輔助教授をはじめとする情報処理論講座の先生方にも深く感謝いたします. 最後に, 情報処理論講座において苦楽を共にし, 有益な御助言を頂いた同輩諸氏にも感謝の意を述べさせていただきます.

# 参考文献


[1] 新山祐介, 秋山英久, 鈴木泰山, 徳永健信, 田中穂積, 自然言語を理解するアニメーテッドエージェントのための 3 次元仮想空間における位置の表現と処理, 1999 年度人工知能学会全国大会論文集, 217-220

[2] 戸田真志, 川嶋稔夫, 青木由直, 画像情報分布に基づく視覚ズームの適応的制御, 信学論, Vol.J81-D-II(1998),84-92

[3] 宮崎英明, 亀田能成, 美濃導彦, 複数のカメラを用いた複数ユーザに対する講義の実時間映像化法, 信学論, Vol.J82-D-II(1999),1598-1605

[4] 宮本圭, 上野敦志, 武田英明, オフィス環境における文字情報の検出と利用に関する研究, 人工知能学会研究会資料, SIG-KBS-9904-2(3/27)

[5] 加藤祥史, 劉 詠梅, 大西 昇, 移動ロボットによる文字情報を利用した環境地図の作成, 計測自動制御学会論文集, 37-1(2001-1),95-96

[6] 和田俊和, 浮田宗伯, 松山隆司, 視点固定型パンチルトズームカメラとその応用, 信学論 D-II, Vol.J81-D-II.6(1998-6),1182-1193

[7] 出口光一郎, 鏡 慎吾, 嵯峨 智, 本谷秀堅, 能動カメラによる運動物体追跡と実時間形状復元システム, 計測自動制御学会論文集, 35-5(1999-5),675-683

[8] R.O.Duda and P.E.Hart. Use of the hough transformation to detect lines and curves in pictures. CACM, 1972.

[9] Sony EVI-D30/D31 製品情報 Home Page, http://www.world.sony.com/JP/Electronics/ISP/products/color/EVID30.html

[10] Argo Craft 社 Home Page, http://www.argocraft.co.jp/

[11] SONY EVI D30/D31 VISCA/RS232C Interface Library, http://www.cs.ubc.ca/spider/vk/sony.html

[12] 菅野 裕, 複数物体間の位置関係の定性的推論, 名古屋大学情報文化学部情報処理基礎学講座,1999

[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　コマンドマニュアル

ユーザの利用可能なコマンドは以下の通りである.

***Command***

| | | |
|---|---|---|
| look, see | *direction* | *direction* の方を向け |
| | *object* | *object* を見よ |
| focus | *object* | *object* を注目せよ |
| zoom | out | ズームひけ |
| | in | ズームせよ |
| home | | 元の位置に戻れ |
| back | | 前の位置に戻れ |
| more, again | | もう一回 ~ して |
| show | map | データベース図を見せて |
| quit, exit | | 終了せよ |

look, see で用いられる *direction* には以下のものがある.

***direction***

| | |
|---|---|
| right | 右 |
| left | 左 |
| above | 上 |
| below | 下 |

look, see, focus で用いられる *object* は次の表現で記述される.

*object*: [*direction*] [*color*] [*size*] *shape*

*color*, *size*, *shape* の語順は任意である. ここで用いられる *color*, *size*, *shape* は以下のものがある. *direction* は上と同じである.

***shape***

| | |
|---|---|
| circle | 円 |
| triangle | 三角形 |
| rectangle, square | 四角形 |
| one, something, object | なにか |

***color***

| | |
|---|---|
| red | 赤 |
| green | 緑 |
| blue | 青 |
| yellow | 黄 |
| orange | オレンジ |
| purple | 紫 |
| cyan | シアン |

***size***

| | |
|---|---|
| big | 大きい |
| small | 小さい |

*object* を修飾する *direction, color, size* はどれも省略可能である．ただし，*shape* は必須である．

また *object* は他物体を参照して表現することができる．

$$[color_1]\ [size_1]\ shape_1\ \{\ direction_1\ of,\ distance_1\ \}\ [direction_2]\ [color_2]\ [size_2]\ shape_2$$

$direction_2$, $color_2$, $size_2$, $shape_2$ で表される物体の $direction_1$ の向き，または $distance_1$ にある $color_1$, $size_1$, $shape_1$ で表される物体を指し示す．

ここで用いた *distance* は以下のものがある．

***distance***

| | |
|---|---|
| near, next | 近い |

**使用例:**

focus big green object
「大きな緑色の物体を注目せよ」
look red circle left of right big triangle
「右手にある大きな三角形の左にある赤い円を見よ」

# 付 録B　ハードウェアマニュアル

## B.1　SONY EVI-D30/31

SONY 製パンチルトカメラ EVI-D30/31 の最新の情報は, 以下を参照.

SONY EVI-D30/31 製品情報 Home Page

http://www.world.sony.com/JP/Electronics/ISP/products/color/EVID30.html

ここでダウンロードできるマニュアル

- インストラクションマニュアル (98BIM7J.pdf)
- コマンドリスト (EVID30CLJ.pdf)

のコピーを, 以下の場所に置いておく.

[解凍先 Directry]

```
└── man/
    ├── 98BIM7J.pdf
    └── EVID30CLJ.pdf
```

C++で書かれた, EVI-D30/31 に特化したコマンドライブラリは以下を参照.

SONY EVI D30/D31 VISCA/RS232C Interface Library

http://www.cs.ubc.ca/spider/vk/sony.html

ここでダウンロードできるファイル

- カメラ制御プログラムサンプル (cam.c)
- コマンドライブラリ (evicamera.c)

のコピーを, 以下の場所に置いておく.

[解凍先 Directry]

```
└── evi/
    ├── cam.c
    └── evicamera.c
```

## B.2 ArgoCraft ビデオキャプチャボード

ArgoCraft 製ビデオキャプチャボードの最新の情報は, 以下を参照.
ArgoCraft ビデオキャプチャボード説明書
http://www.argocraft.co.jp/vcb/doc/acvc.html
この中の Linux 向けの情報を抽出して以下に書く.

### B.2.1 ソフトウェアのインストール

1. アルゴクラフトの WWW ページ
   http://www.argocraft.co.jp/vcb/info.html から Linux および FreeBSD 用ソフトウェアのアーカイブファイルを入手します. 同様のものが,
   [解凍先 Directry]
   └ acvc/
   　└ acvc_132.tgz
   にある.
2. アーカイブファイルを適当なディレクトリに展開します.

```
% tar xvzf acvc-1.32.tar.gz
acvc-1.32/
acvc-1.32/common/
acvc-1.32/common/vctest.c
（中略）
acvc-1.32/FreeBSD-2.2.x/xvctest2
acvc-1.32/FreeBSD-2.2.x/xvctest3.o
acvc-1.32/FreeBSD-2.2.x/xvctest3
```

3. カレントディレクトリを, 展開したファイル中の Linux 用ディレクトリに移動します.

```
% cd acvc-1.32
% cd Linux-2.0.x
```

4. xmkmf コマンドを実行して Makefile を生成します.

```
% xmkmf -a
mv -f Makefile Makefile.bak
imake -DUseInstalled -I/usr/X11R6/lib/X11/config
```

```
make Makefiles
make: Nothing to be done for ‘Makefiles’.
make includes
make: Nothing to be done for ‘includes’.
make depend
gccmakedep  --   -I/usr/X11R6/include  -Dlinux -D__i386__
-D_POSIX_SOURCE -D_BSD_SOURCE -D_GNU_SOURCE -DX_LOCALE  -DF
UNCPROTO=15 -DNARROWPROTO    -- xvctest.c xvctest2.c
gccmakedep  -a -- -- acvc.c
gccmakedep  -a -- -- vctest.c
gccmakedep  -a -- -- vctest2_.c
gccmakedep  -a -- -- stereo.c
gccmakedep  -a -- -- svctest.c
gccmakedep  -a -- -- svctest2.c
```

5. はじめから make 済みのバイナリが存在しますが, 念のため再 make を行います. 再 make にあたっては svgalib が必要ですので, すでにインストールされていない場合はあらかじめインストールしてください.

```
% make clean
rm -f acvc.o vctest vctest2_ stereo svctest svctest2 xvctest xvct
est2
rm -f *.CKP *.ln *.BAK *.bak *.o core errs ,* *~ *.a .emacs_*tags
TAGS make.log MakeOut  "#"*
% make all
rm -f acvc.o
gcc -O2 -Wall -c acvc.c -o acvc.o
rm -f vctest
gcc -O2 -Wall vctest.c -o vctest
rm -f vctest2_
gcc -O2 -Wall vctest2_.c -o vctest2_
rm -f stereo
gcc -O2 -Wall stereo.c -o stereo
rm -f svctest
gcc -O2 -Wall svctest.c -lvgagl -lvga -o svctest
rm -f svctest2
gcc -O2 -Wall svctest2.c -lvgagl -lvga -o svctest2
gcc -O2 -Wall -ansi    -I/usr/X11R6/include  -Dlinux -D__i386__
```

```
-D_POSIX_SOURCE -D_BSD_SOURCE -D_GNU_SOURCE -DX_LOCALE  -DFUNCP
ROTO=15 -DNARROWPROTO     -c xvctest.c -o xvctest.o
rm -f xvctest
gcc -o xvctest -O2 -Wall -ansi     -L/usr/X11R6/lib xvctest.o
-lXExExt -lXext -lX11       -Wl,-rpath,/usr/X11R6/lib
gcc -O2 -Wall -ansi    -I/usr/X11R6/include  -Dlinux -D__i386__
 -D_POSIX_SOURCE -D_BSD_SOURCE -D_GNU_SOURCE -DX_LOCALE  -DFUNC
PROTO=15 -DNARROWPROTO     -c xvctest2.c -o xvctest2.o
rm -f xvctest2
gcc -o xvctest2 -O2 -Wall -ansi     -L/usr/X11R6/lib xvctest2.o
 -lXaw -lXmu -lXt -lSM -lICE -lXExExt -lXext -lX11 -Wl,-rpath,
/usr/X11R6/lib
```

6. スーパーユーザになって, インストールとデバイスファイルの作成を行います.

```
% su
Password:
# make install
install -c -s  xvctest /usr/X11R6/bin/xvctest
install -c -s  xvctest2 /usr/X11R6/bin/xvctest2
install -c -d /lib/modules/‘uname -r‘/misc
install -c -m 0644 acvc.o /lib/modules/‘uname -r‘/misc/acvc.o
install -c -d /usr/local/include
install -c -m 0444 acvc.h /usr/local/include/acvc.h
install -c -d /usr/local/bin
install -c -s vctest /usr/local/bin/vctest
install -c vctest2 /usr/local/bin/vctest2
install -c -s vctest2_ /usr/local/bin/vctest2_
install -c -s stereo /usr/local/bin/stereo
install -c -s -m 4755 svctest /usr/local/bin/svctest
install -c -s -m 4755 svctest2 /usr/local/bin/svctest2
install in . done
# make dev
rm -f /dev/acvc /dev/acvc0 /dev/acvc1 /dev/acvci
mknod -m 444 /dev/acvc u 60 0
mknod -m 444 /dev/acvc0 u 60 1
mknod -m 444 /dev/acvc1 u 60 2
```

```
mknod -m 444 /dev/acvci u 60 3
# exit
exit
```

7. 毎回 Linux の起動直後に, スーパーユーザとなり insmod コマンドでドライバを組み込みます.

   ```
   % su
   Password:
   # insmod acvc
   # exit
   exit
   ```

   多くのシステムでは初期化スクリプトの記述によって, 自動的に組み込みを行うことができます. 当 Vine 環境では /etc/rc.d/rc.sysinit のはじめのほうに「/sbin/insmod acvc」と書き加えます.

### B.2.2 デバイスドライバの使用法

Linux および FreeBSD 用ドライバの使用法は, vctest のソースファイル vctest.c の注釈で詳しく説明されていますので, そちらを参照してください. vctest.c は Linux および FreeBSD 用ソフトウェアのアーカイブに含まれています.

同様のものを以下に置いておく.

[解凍先 Directry]

└ acvc/

　　└ vctest.c

にある.

# 付 録C　ソフトウェアマニュアル

本システムのアーカイブ, マニュアル等は以下に置いてある.
joker の /home/graduates/sugano/

| | |
|---|---|
| cam.tgz | 本システムのアーカイバ |
| thesis.pdf | 本論文全体 |
| software.pdf | 本紙 |
| hardware.pdf | ハードウェアインストールマニュアル |

本システムは C, C++コンパイラ, Perl インタプリタが必要である. gcc 2.8.1, g++ 2.8.1, perl 5.004_04 で動作確認している.

## C.1　解凍, コンパイル

1. アーカイブファイルを適当なディレクトリに展開する.

```
% tar xvzf cam.tgz
acvc.h
border.c
...
```

2. メインモジュールをコンパイルする.

```
% make cam
g++    -c hough.c -o hough.o
g++    -c fourier.c -o fourier.o
...
```

3. 動画表示モジュールをコンパイルする.

```
% make xvc
gcc -o xvc xvc.c -L/usr/X11R6/lib       -lX11 -lXpm
```

4. Perl 受渡しモジュールをコンパイルする.

```
% make ctop
gcc -o ctop ctop.c `perl -MExtUtils::Embed -e ccopts -e ldopts`
```

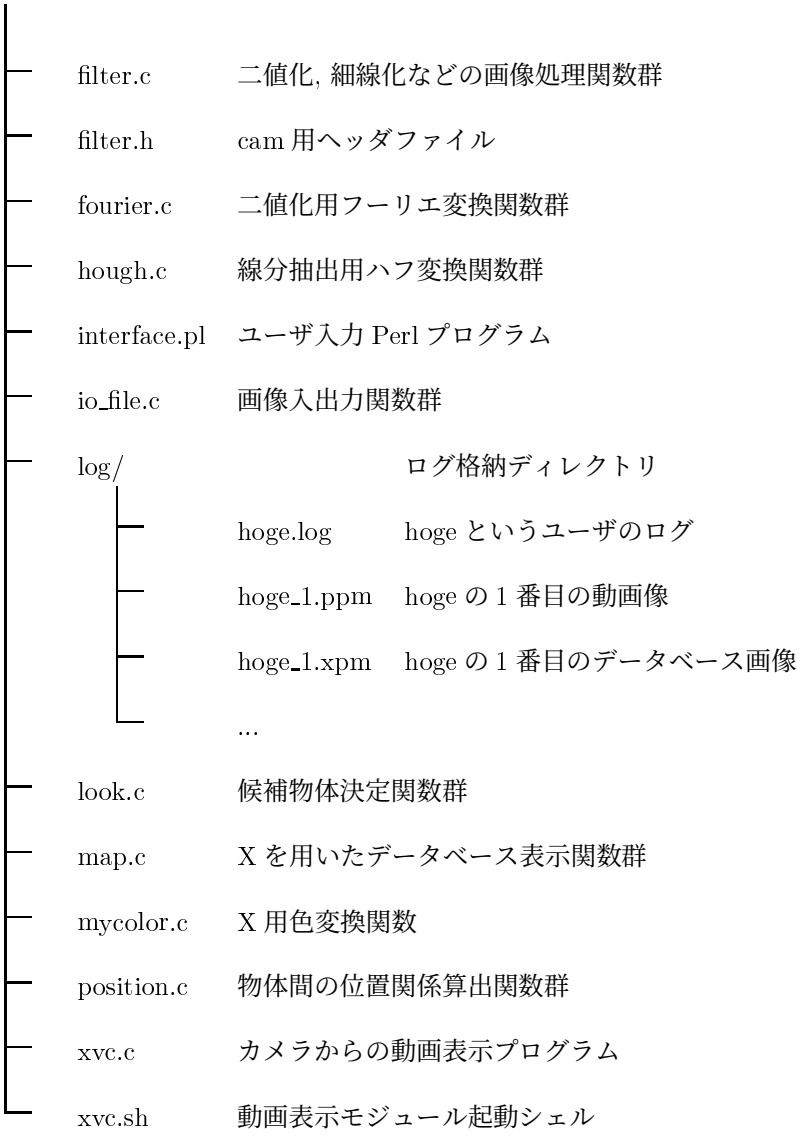
エラーを出さず, cam, xvc, ctop の三つのファイルができたらコンパイル終了.

## C.2 ファイル構成

アーカイブを解凍して作られるファイル, ディレクトリは以下の通りである.

[解凍先 Directry]

| | |
|---|---|
| acvc/ | |
| acvc_ 121.tgz | キャプチャボードデバイス, ソフトウェアアーカイブ |
| vctest.c | ArgoCraft キャプチャボード用キャプチャソフト |
| evi/ | |
| cam.c | カメラ制御プログラムサンプル |
| evicamera.c | EVI-D30 コマンドライブラリ |
| man/ | |
| 98BIM7J.pdf | EVI-D30 インストラクションマニュアル |
| EVID30CLJ.pdf | EVI-D30 コマンドリスト |
| acvc.html | ArgoCraft キャプチャボード説明書 Index |
| Makefile | Makefike |
| acvc.h | キャプチャボード用ヘッダファイル |
| border.c | 画像からの物体の抽出, 成分分析を行う関数群 |
| cam.c | メインプログラム |
| cam_func.c | 物体の角数計算, crop, ズーム, 近傍度計算など雑多な関数群 |
| color.c | 色識別関数群 |
| ctop.c | C での Perl インタプリタ呼び出しプログラム |
| evi_func.c | 制御前処理, スキャン, 移動などのカメラコントロールを含む関数群 |
| evicamera.c | EVI-D30 コマンドライブラリ |




## C.3 起動方法

1. 動画表示モジュールを起動する.

   ```
   % ./xvc.sh &
   ```

   ウィンドウが一つ開き, カメラの Power が On になっていれば, 入力動画像が表示される. Off であれば黒いままである.
2. メインモジュールを起動する.

   ```
   % ./cam hoge
   ```

```
Log Output > ./log/hoge.log
なんなりとご命令をお申しつけください
BANANA >
```

この際, 第一引数を名前にしてログが残る. 今回の例では hoge が名前になる. メインモジュール起動時にカメラの Power が On になり, 初期動作を経て動画表示モジュールのウィンドウに動画像を表示する.

3. 命令を入力する.

```
BANANA > look above circle
perl は正常に動作してるにゃ
...
```

## C.4 制御の流れ

図 C.1 にシステムの制御の流れを摸式的に表す.

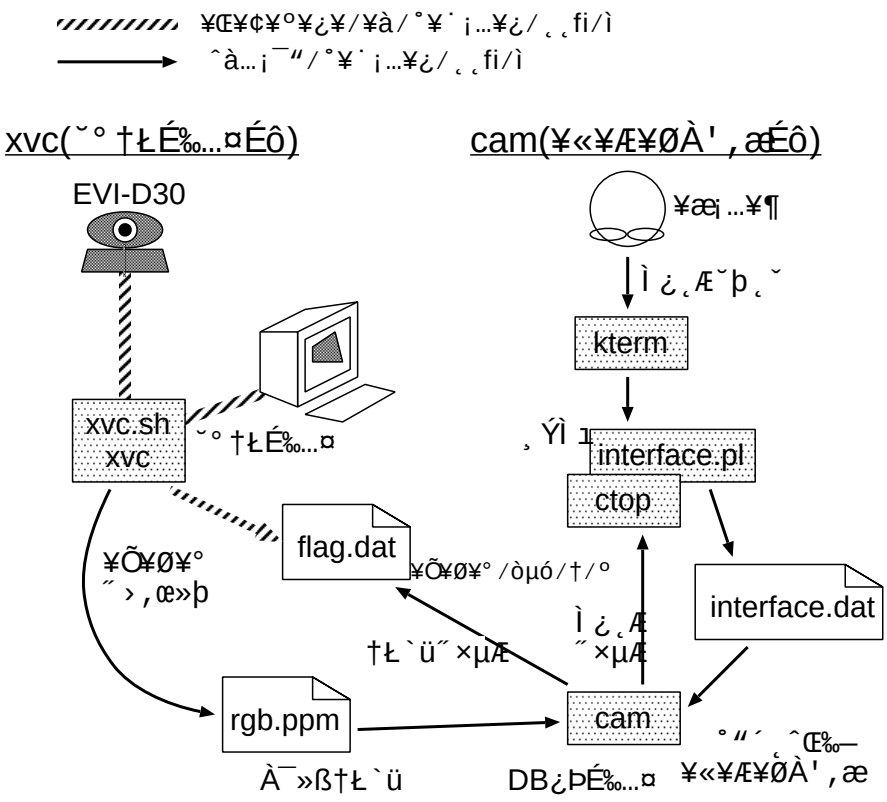



図 C.1: whole process and data-flow

xvc.c をコンパイルしてできる xvc は, キャプチャボードが取得したビデオ信号をリアルタイムに動画として表示するアプリケーションである. また同時に, flag.dat を監視し, cam からの静止画像の要求があると, すなわち, 1 が立っていると, ビデオ信号から静止画像を作り, rgb.ppm として出力する.

Warning:
ビデオボードを替えるなどして, xvc を使えない場合が考えられる. その際は, 動画表示は使用のビデオボードに対応したアプリケーションを用い, そのプリケーションの中から, ないしは別のアプリケーションで静止画像を作成する必要がある. 静止画像を作るタイミングは, flag.dat の中身が 1 になった時である. これは, 常に監視する必要がある. また, 静止画像 rgb.ppm を更新した後は flag.dat の中身を 0 に戻さなければならない.

静止画像は ppm 形式に限定する. ppm 形式の仕様を以下に示す.

P6
width height
depth
(0,0) の RGB 値, (0,1) の RGB 値, ...
(1,0) の RGB 値, ...
...
(width-2, height-1) の RGB 値, (width-1,height-1) の RGB 値

width, height はそれぞれキャプチャ画像の幅, 高さである. depth は色の階調である. 32 階調なら 31, 256 階調なら 255 である. その後はバイナリデータであり, depth を納めるデータ型で書かれる. 左上の画素の赤成分, 青成分, 緑成分の順で, 右下の画素まで続く.

静止画像が ppm 形式であれば, カメラ制御部分はビデオボードと独立しているため, 影響を受けない.