修士論文

観察によるロボット動作スキルの獲得

平成12年2月4日提出

指導教官　池 内　克 史　　教授

電子情報工学専攻

８６４５０　　冨長　裕久

内容梗概

ロボットの特長は物理的な構造を大きく変えることなく様々な作業を行なうことができる汎用性にある。しかし、ロボットに作業をさせるためには適切な動作の指示が必要であり、指示が簡単に行なえなければロボットの汎用性は生かされない。

作業動作の教示法としては、teach-by-showing、テレオペレーション、ロボット用プログラミング言語による記述などが提案されている [1]。teach-by-showing はオペレータが教示モードにおいてロボットの軌道を教示し、実行モードでその軌道に従ってロボットが動く方法である。この方法は、工業ロボットで多く用いられており、単純な繰り返し作業を行なわせるには適している。しかし、複雑な動きを行なわせるためにはオペレータの熟練が必要となり、必ずしも簡便な教示法とは言えない。テレオペレーションシステムでは、二つのロボットを用意して、作業者はその一方をロボットとは別の場所で操作する。その動きに従って、遠隔地にある他方のロボットが実際の作業を行ない、その様子はカメラを通して作業者に送られる。この方法の欠点は、軌道の情報しかロボットに教示することができないので、力の具合いなどを自動的に調節したり、誤った軌道を修正することが難しい。プログラミング言語を使った動作記述では、ロボットが行なうことのできるすべての機能を使用することができるという点が利点となるが、開発に時間がかかることとプログラミングに特別な知識を必要とする点が欠点となる。

そこで、視覚システムにより、対象物体および人間が行なう作業動作を観察することにより、どのような作業が行なわれたのかを認識し、それをロボット用に展開することにより、人間と同じ作業を実行するシステムが提案されている。本論文では視覚システムを利用するもののうち、物体の面接触関係の変化を元に、作業の内容を定義して実行を行う APO システム [1][2] と、物体の移動した軌跡を元に作業を実行するシステム [3][4] に注目した。

APO システムは人間の作業の前後において扱う物体の面接触関係を解析し、その面接触状態の変化を元に必要な動作を選びだす。この方法によって、作業中の補正動作などを適切に選びだすことが可能である。しかしながら、人間が何を行っていたかということは認識しても、どのように行ったかという情報を得ることができないため、複雑な作業にたいしては実装が難しいという欠点がある。それに対して、軌跡を元にするシステムは、人間の動作をそのまま模倣するシステムであるので、実装は簡単になるが、作業の目的を認識していないために、実行時に誤差があってもそれを修正することが出来ない。

そこで本論文では、得られた人間動作の軌道を、作業の途中状態における物体の接触状態を元に分割し、分割された状態に対して適切な動作を割り当てるという手法を提案した。これによって、教示されたお手本動作から動作の目的と方法を得ることができ、より信頼性の高い作業を行わせることができた。また、途中状態を分割し記述するための手法についても提案した。

目 次

第1章　序論

1.1　研究の背景

ロボットの特長は物理的な構造を大きく変えることなく様々な作業を行なうことができる汎用性にある。しかし、ロボットに作業をさせるためには適切な動作の指示が必要であり、指示が簡単に行なえなければロボットの汎用性は生かされない。

作業動作の教示法としては、teach-by-showing、テレオペレーション、ロボット用プログラミング言語による記述などが提案されている [1]。teach-by-showing はオペレータが教示モードにおいてロボットの軌道を教示し、実行モードでその軌道に従ってロボットが動く方法である。この方法は、工業ロボットで多く用いられており、単純な繰り返し作業を行なわせるには適している。しかし、複雑な動きを行なわせるためにはオペレータの熟練が必要となり、必ずしも簡便な教示法とは言えない。テレオペレーションシステムでは、二つのロボットを用意して、作業者はその一方をロボットとは別の場所で操作する。その動きに従って、遠隔地にある他方のロボットが実際の作業を行ない、その様子はカメラを通して作業者に送られる。この方法の欠点は、軌道の情報しかロボットに教示することができないので、力の具合いなどを自動的に調節したり、誤った軌道を修正することが難しい。プログラミング言語を使った動作記述では、ロボットが行なうことのできるすべての機能を使用することができるという点が利点となるが、開発に時間がかかることとプログラミングに特別な知識を必要とする点が欠点となる。

そこで、視覚システムにより、対象物体および人間が行なう作業動作を観察することにより、どのような作業が行なわれたのかを認識し、それをロボット用に展開することにより作業を実行するシステムが提案されている。國吉 [5] は、一連の作業を意味のあるいくつかの部分に分けて階層化し、大局的な観察からより詳しい観察へ階層毎に理解することで、作業方法を得る手法を提案した。Takahashi[7] は、仮想環境を利用してユーザの動作を入力し、動作中の状態遷移をオートマトンによって解析し、作業内容を理解するシステムを構築した。また、作業中のロボットの動作を視覚や力覚センサーで観察し、エラーリカバリに役立てる研究もなされている [8][10]。

しかしながら、これらは作業の目的あるいは作業の中間状態における目標状態を定めることはできるが、その目的状態を実現するための戦略は作業ごとに、あらかじめプログラムして与えなければならなかった。そのため、ある特定の作業においては機能するが、一般的な作業を行わせることを考えた場合、プログラミングが非常に複雑になるという欠点があった。そこで、作業の戦略も観察から得るために、人間の動作をそのままロボットに

模倣させるシステムが提案された [3] が、一度得られた動作は変更することが出来ないため、実行時のエラーリカバリができないという問題があった。

そのため、作業の目的と戦略をともに観測から得られるシステムが必要と考えた。

1.2 研究の目的

本論文では視覚システムを利用するもののうち、物体の面接触関係の変化を元に、作業の内容を定義して実行を行う APO システム [1][2] と、物体の移動した軌跡を元に作業を実行するシステム [3][4] に注目した。

APO システムは人間の作業の前後において扱う物体の面接触関係を解析し、その面接触状態の変化を元に必要な動作を選びだす。この方法によって、作業中の補正動作などを適切に選びだすことが可能である。しかしながら、人間が何を行っていたかということは認識しても、どのように行ったかという情報を得ることができないため、複雑な作業にたいしては実装が難しいという欠点がある。それに対して、軌跡を元にするシステムは、人間の動作をそのまま模倣するシステムであるので、実装は簡単になるが、作業の目的を認識していないために、実行時に誤差があってもそれを修正することが出来ない。

そこで本論文では、得られた人間動作の軌道を、作業の途中状態における物体の接触状態を元に分割し、分割された状態に対して適切な動作を割り当てるという手法を提案した。これによって、教示されたお手本動作から動作の目的と方法を得ることができ、より信頼性の高い作業を行わせることを目的とする。これを実現するために、途中状態を記述する手法を定義し、それに基づいて得られた人間動作を分割する一般的な方法を提案する。そして、その分割された中間状態に割り当てらるべき作業を sub-skill として定義し、その割り付け方法についても述べる。これらの sub-skill の組合わせによって、全体として信頼性の高い作業スキルを獲得し、実行させることを目的とする。

本論文の構成は以下の通りである。第 2 章では観察を使った作業システムを概観する。第 3 章では作業の途中状態を記述する方法を述べ、またその記述を使用して、いかにして一つの作業をよりプリミティブな作業に分割するかを述べる。第 4 章では動作プリミティブであるサブスキルを定義し、第 5 章でサブスキルの実装法を検討した後、本論文で提案した手法を使って実際に作業を行った結果を示す。第 6 章で結論を述べる。

第2章　観察を利用した教示法

前章 1.1 節で述べたように、ロボットに作業を行わせるための教示法にはさまざまの方法が提案されているが、その中でも観察を利用した教示やエラーリカバリの手法が有用である。本章では、今までに提案されている観察を利用したロボットへの教示、作業計画の生成法を概観し、その利点と問題点を考察すると共に、本研究の位置付けを明らかにする。

2.1　作業目的の認識

人間の動作を観察することによって、ロボットの動作を生成するためには、まず作業内容の理解が必要となる。本節では観察によって動作の目的を理解する方法について紹介する。[7][1][2]

Takahashi[7] は、オートマトンと仮想現実空間を利用して人間の作業を理解する手法を提案している。

この手法では教示者は仮想現実感を利用したインターフェースで動作を教示する。これにより、手の形状等を測定する必要がなくなるという利点がある。

仮想空間内の教示者の動きを「物をつかむ」などの意味的なレベルで認識するためには、手の形や動きに加え、周りの状態を考慮する必要がある。例えば、部品を把持する動作と空中で手を握る動作とでは、同一の手の動きであっても異なった目的を持った操作と解釈されなければならない。そのため [7] は教示者の動作における手の形や手振りに加えて、環境を含めた状態を以下のように定義した。

1. 手の形 (Hs) は、指の関節の曲げ角や手の向きで決まる。手の姿勢 (Hp) は、手の形 Hs と手の 3 次元位置で決まる。
2. 環境 (W) は、教示者の手の周りの環境を規定する。組み立て作業においては、仮想作業空間を構成する部品を多面体近似した頂点座標や部品間の幾何学的位置関係、拘束条件などが環境 W の構成要素となる。
3. 手振り (Hg) は、環境 W における手の姿勢 Hp の解釈である。

手振り (組み立て動作) の解釈モデルを有限オートマトンモデル

$$M =< Q, \Sigma, \delta, q_0, F >$$

で表現する。Qは手振りのなされる環境の状態、Σはオートマトンへの入力で手の姿勢Hpが入力となる。δは状態遷移関数で、q_0とFはそれぞれオートマトンの初期状態と最終状態の集合である。例として、パラレルジョーのマニピュレータにおける動作を教示するときのオートマトンを以下に示す。

- $Q = \{q_0, q_1, q_2, q_3\}$

  q_0:部品を把持していず、近くに部品がない。(初期状態)

  q_1:部品を把持していず、近くに部品がある。

  q_2:部品を把持し、近くに部品がない。

  q_3:部品を把持し、近くに部品がある。

- $\Sigma = \{a_1, a_2, a_3\}$

  a_1:手を握る動作。

  a_2:手を開く動作。

  a_3:動く動作。

- δ:$Q \times \Sigma \to Q$で定義される。状態図をFig. 2.1に示す。

- $F = \{q_0, q_1\}$

手振りとしては、３種類の組み立て作業を規定している。

- $Hg = \{op_1, op_2, op_3\}$

  op_1:部品を持ち上げる。(pick up an object)

  op_2:把持している部品を作業台に置く。(put the object)

  op_3:把持している部品を他の部品の上におく。(put the object on an object(s))

組み立て動作は、手の開閉動作(a_1, a_2)と組み立て部品の状態に依存する$\delta(q_1, a_1) = q_2$、$\delta(q_2, a_2) = q_1$、$\delta(q_3, a_2) = q1$の状態遷移に対応する。Fig.2.2に、仮想環境においてネジをブロックに挿入する操作の教示シーケンスと解釈された手振りHgを示す。

以上によって手振りが認識されるが、この手法では、仮想空間を使っているために、実際に動作を行う際の力の加減や、エラーリカバリの方法を観察から得ることができない。また、扱う物体の形状と用途をあらかじめ定義し、それに基づいて手振りの意味を理解しなくてはならないため、作業毎に状態を定義しなくてはならず、認識できる手振りが限られる。そこで次に、より一般的な動作の目的を認識する手法について紹介する。

Assembly Plan from Observation (APO) システム [1][2] は、作業前と後の面接触関係を解析することによって、作業内容を理解する。

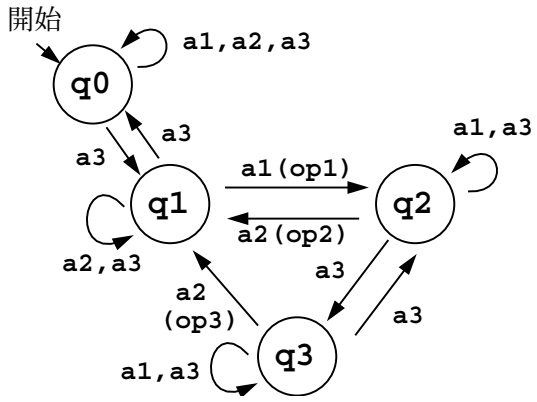


Fig. 2.1: 手振り理解モデルの状態図

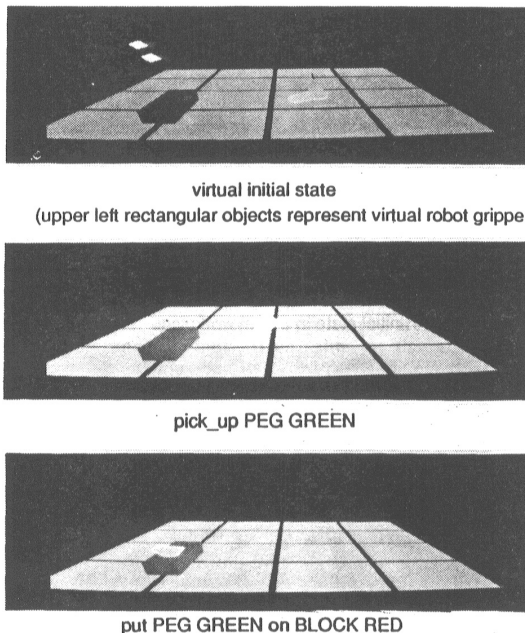


Fig. 2.2: 手振り認識の例

まず、任意の二つの多面体における、可能な面接触を抽出する。例として、操作物体と環境物体が一つの面で接している場合を考える。この時、操作物体の可能な動かし方は、式 2.1 で制限される。

$$\mathbf{N} \cdot \Delta\mathbf{T} \geq 0 \tag{2.1}$$

ここに $\Delta\mathbf{T}$ は操作物体の動き、$\mathbf{N}$ は環境物体の法線ベクトルである。

操作物体の可能な動きベクトルはガウス球面上の点として図解できる。球の中心をベクトルの起点、球面上の点をベクトルの先として、動きベクトルを表現する。この例においては、Fig.2.3 における上半球が操作物体が動ける範囲を示し、同時に環境物体との接触がなくなる動きを示している。また、下半球は環境物体があるために動けないことを表わし、赤道にあたるベクトルは面接触を維持した動きを示す。

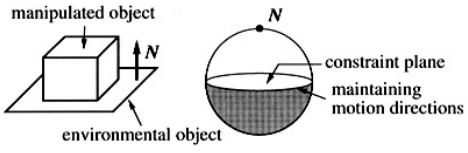


Fig. 2.3: ガウス球面による拘束の表現

多面接触が起こっている場合も、同様に考えることができる。例えば、二つの面で接触が起こっている場合は、次の二つの式で拘束が表現される。

$$\mathbf{N_1} \cdot \Delta\mathbf{T} \geq 0 \tag{2.2}$$

$$\mathbf{N_2} \cdot \Delta\mathbf{T} \geq 0 \tag{2.3}$$

$\mathbf{N_1}$,$\mathbf{N_2}$ はそれぞれの面の法線ベクトルを表わす。

このようにして、組み立て作業中に起こり得る面接触関係をすべて数えると、Fig. 2.4 のような 10 種類の面接触関係が起こり得ることが分かる。

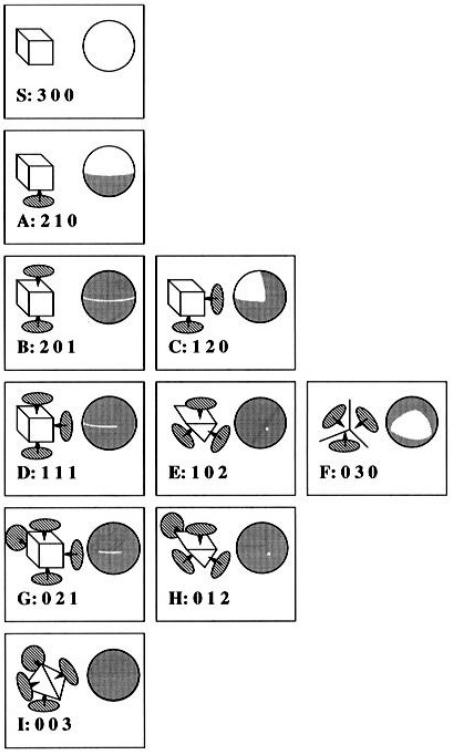


Fig. 2.4: 面接触関係状態図; 3 つの数字はそれぞれ maintaining DOF, detaching DOF, constraining DOF を表わす。

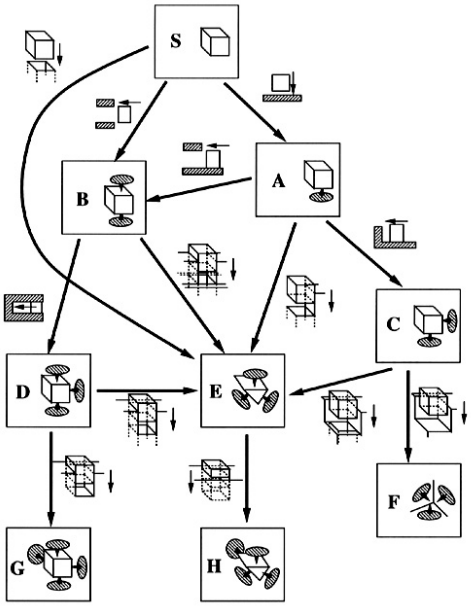


Fig. 2.5: 面接触関係遷移図

次に、これらの状態の可能な遷移を考える。組み立て作業は拘束を増やす方向に状態が遷移する作業と考えられるから、これら 10 種類の状態の組み立て作業における遷移は 13 の場合だけを考えれば良いことが分かる。この状態遷移図を Fig. 2.5 に示す。

この状態遷移図を使って、作業前後の面接触関係から状態遷移を認識することにより、どのような作業が行なわれたかを理解する。

作業の開始と終了は画像の明るさの変化から検出される。作業が終了すると距離画像が取得され、作業前の画像との差分から操作物体を認識する。認識された操作物体に、あらかじめ与えられている CAD モデルを当てはめ、面接触関係を解析する。解析された作業前後の面接触関係と前述の状態遷移図から面接触遷移が求められる。

解析された面接触遷移にはあらかじめ motion macro と呼ばれる、作業実行のための手順が対応しており、これを呼び出すことにより、実際に作業を行なう。

APO システムの利点は面接触関係とその遷移という、シンボリックな表現で作業を認識することにある。そのため、複雑な作業も有限個の状態遷移に分類され、それに応じて作業を割り当てることも可能になっている。また、作業の目的が分かっていることから、実行時にエラーを検出しリカバリできる。

しかしながら、motion macro には二つの欠点が考えられる。一つは、人間動作がどのように行われたかという、軌道情報を持たないために、CAD モデルと環境の情報から実際に実行される動作の軌道を生成しなければならないことである。二つめの欠点は、motion macro は作業の最初と最後をつなぐ全ての動作を生成しなければならないことである。そのため motion macro は複雑で巨大なものとなり、実装が難しくなる。

Miura[12] は組み立て作業における状態遷移図を、より一般的な作業を記述できるように拡張している。これは並進運動に加えて、一つの回転軸に対する回転運動を加えたもので、状態は 54 個の状態が定義され、その遷移は 85 種類が考えられる。この事からも類推されるように、並進 3 自由度と回転 3 自由度を許した一般的な作業の状態を記述しようとすると、組み合わせ数が膨大になってしまう。そこで、扱う物体を多面体に限り、スクリュー理論 [13][14] を使ってシンボリックに状態を記述する方法を 3 章で述べる。ただし、Miura はこの状態遷移図を作業計画に利用するのではなく、視覚センサーを使わなくては実行が困難な遷移の検出に使用している。そのような遷移は maintaining DOF が constraining DOF に変化するような遷移であると述べられている。本研究でも実装段階においてどのようなセンサを使うべきであるかに関して、この基準を指針とする。

2.2 作業方法の決定

前節では作業内容を理解する方法を述べたが、実際にどのように作業を行なうかということについて、さらに考える必要がある。

前節の手法では、得られた作業の目的に対して、実行ライブラリを呼び出して実行している。しかし、この方法ではライブラリが複雑で大きなものになり過ぎるという欠点を持つ。

そこで、作業をどのように行なうかも観察から取得するため、操作物体のパスを利用する [5]。また、得られたパスが実行可能かどうかを解析する手法も提案されている [3][4]。さらに、操作物体をどのように把持するかということを観察から得る手法も提案されている [15][16][17]。

國吉 [5] は、一連の作業を意味のあるいくつかの部分に分けて階層化し、大局的な観察か

らより詳しい観察へ階層毎に理解することで、作業方法を得る手法を提案している。

この手法では動作を、ある時刻に開始され、別の時刻に終了し、その間 (時区間) に、環境および動作主の体の位置姿勢に対して意味のある変化を生じさせる事象であると定義している。この定義にしたがって、開始時点と終了時点における環境および動作主の状態の差に着目することによって動作を分類し、モデル化している。

一つの動作の開始と終了は、手と操作対象物体との接触状態の変化や、手や操作物体の運動特徴の変化によって抽出される。抽出された動作のうち、情報が多く含まれると思われる部分に関してはさらに細かく分割し、より詳しい情報を得る。このようにして分割された動作モデルは、分割の細かさに応じて階層構造に表わすことができる。

また、動作の分割に合わせて、その動作を特徴づける環境情報も階層化されて用意する。例えば、接触という動作に対して、面やエッジ等の特徴が対応づけられる。対応づけされた階層モデルの図を Fig.2.6 に示す。

このような構造を用意し、まず動作モデルの最上位ノードである “Task” が起動される。Task は環境モデルの最上位レベル “World” すなわちシルエット画像によって、シーンを大局的に観察する。

大局的な観察が行なわれている際に、より細かい下位の階層での観察が必要な場合は、これを呼び出す。実時間で動かすために、下位の階層による観察は上位の観察が終了すると打ち切られる。この様子を「物体を持ち上げて移動する」という動作を例にとって Fig.2.7 に示す。

このようにして、人間の動きの大局的な面と局所的な面から観察し、その動きをロボットに教示する。

この手法では得られた動作軌跡をそのまま用いているため、観測誤差の補正手法を持た

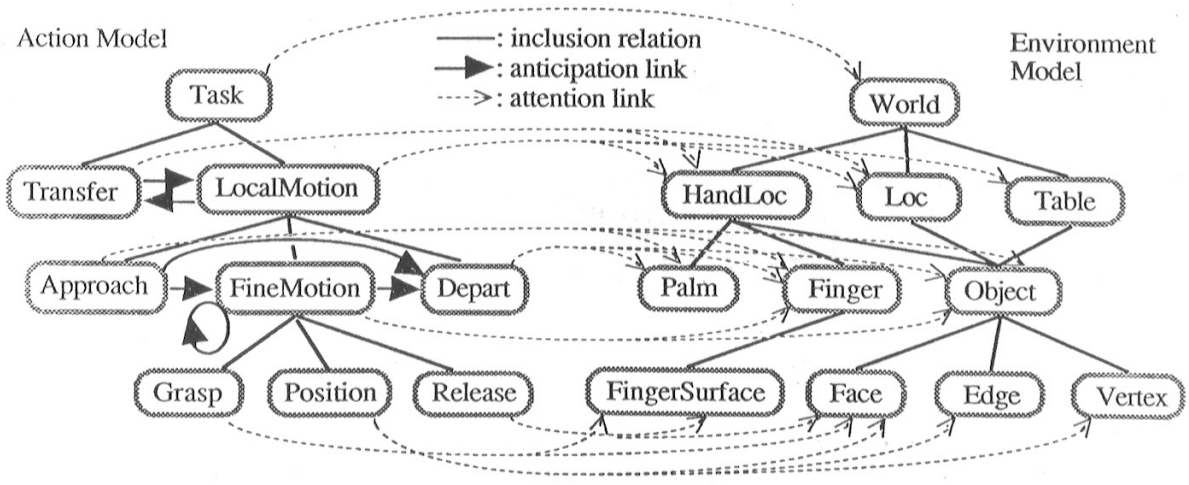


Fig. 2.6: 動作と物体の階層構造

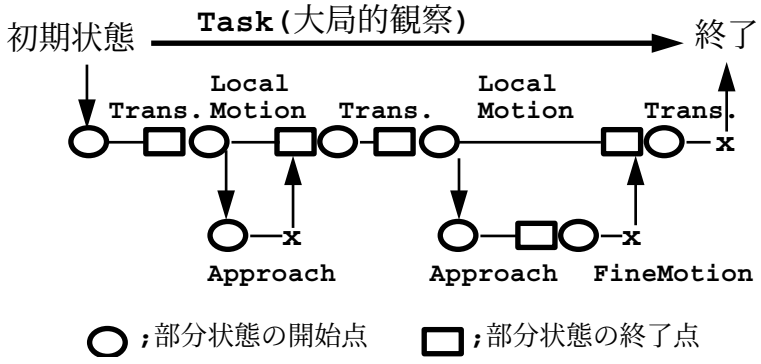


Fig. 2.7: タスク実行の流れ

ない。そこで次に、観測された軌跡を補正する手法について紹介する。

Paul[4] は得られた操作物体の軌跡を、実行可能なものに補正する手法を提案している。

システムはあらかじめ操作物体及び環境物体のモデルを持っている。観測によって物体の姿勢が与えられると、このモデルを使って以下のように接触関係を見つける。以下では平面内の運動を例にとる。平面内の運動では以下の 3 種類の接触を考えれば良い。

頂点とエッジの接触　頂点座標を (v_x, v_y) とし、エッジの方程式を $e_a x + e_b y + e_c = 0$ とする。このとき接触を表す条件は、式 2.4 で表される。

$$e_a v_x + e_b v_y + e_c < \delta_{ve} \tag{2.4}$$

ただし、(e_a, e_b) は 2 次元の単位ベクトルである。さらに、頂点のエッジへの投影がエッジの端点の間にあるかどうかを調べる。

頂点と頂点の接触　二つの頂点を (v_{x1}, v_{y1}) と (v_{x2}, v_{y2}) とする。このとき二つの頂点が接触している条件は、式 2.5 で表される。

$$\sqrt{(v_{x1} - v_{x2})^2 + (v_{y1} - v_{y2})^2} < \delta_{vv} \tag{2.5}$$

エッジとエッジの接触　二つのエッジの方程式を $e_{a1}x + e_{b1} + e_c = 0$ と $e_{a2}x + e_{b2} + e_c = 0$ とする。このとき、二つのエッジが接触している条件は、式 2.6 で表される。

$$(1 - (e_{a1}e_{a2} + e_{b1}e_{b2}) < \delta_{ee1}) \wedge (|e_{c1} - e_{c2}| < \delta_{ee2}) \tag{2.6}$$

これらの接触を満たす条件は、C-Space 中で C-Obstacle と呼ばれるマニフォールドによって表現される。

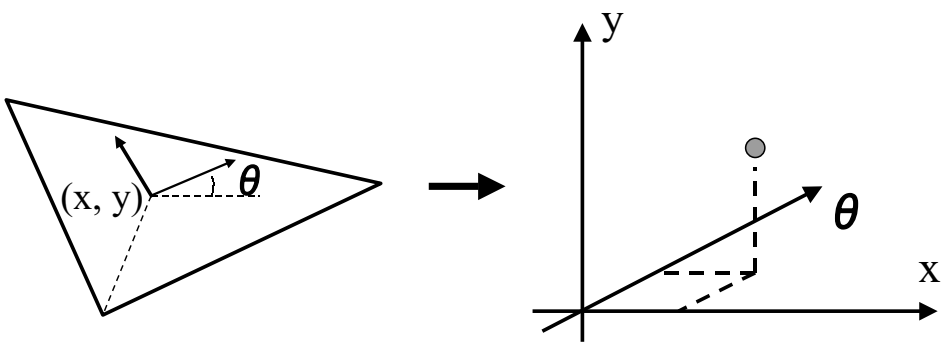


Fig. 2.8: 二次元物体の C-Space 中での表現

C-Space(Configuration Space) とは物体の位置と姿勢を同時に表現する多次元空間である。たとえば、２次元平面内の物体の位置・姿勢は Fig.2.8 のように、(x, y, θ) の３次元空間内の点で表現される。ここに、x, y は物体の位置を、θ は物体の姿勢を表す。

次に C-Space を使ってどのように拘束条件が表されるかを説明する。

例として、Fig.2.9 で示されるような、二つの物体の接触を考える。Fig.2.9(a) は操作物体が姿勢を一定に保ち、環境物体に対して接触しながら環境物体の周囲を一周したときの軌跡を示している。この時、点線で示された軌跡の内側は、操作物体が存在することのできない位置を示していると考えることができる。

そこで、全ての姿勢に対して、同様な解析を繰り返し、それら全ての禁止領域を C-space 中に表現する。すると、Fig.2.9(b) に示されるように、これらの禁止領域をつないだマニフォールドが形成される。これが C-obstacle であり、C-space 中の拘束の表現と解釈される。

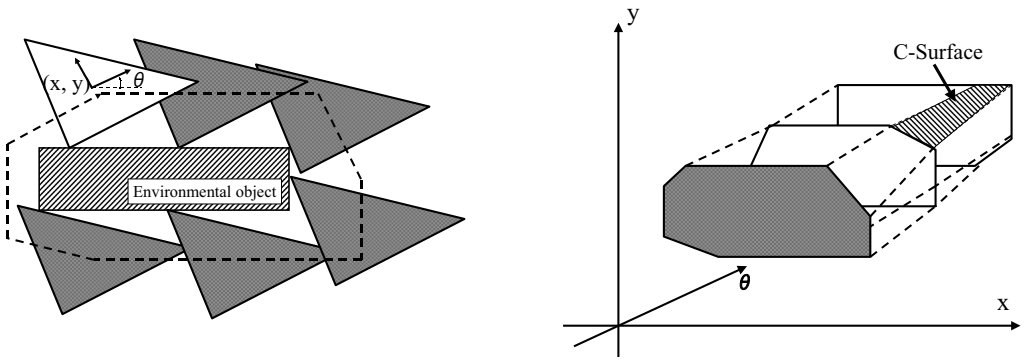
このとき、ある頂点とあるエッジが接触を保つための条件は C-obstacle を構成する曲面の一つとなる。これは C-surface と呼ばれ、Fig.2.9(b) に示したような C-obstacle の側面である。

接触が起きているとき、観測された接触点の C-Space 内の点 $o_i(x, y, \theta)$ と、C-Surface C_j が与えられると、o_i に最も近い C_j 上の点を計算することができる。これより真の値の推定値 $t_i(x, y, \theta)$ を計算することができる。このとき観測誤差は

$$E(o_i, t_i) = \|t_i - o_i\| \tag{2.7}$$

であると考えられる。

これによって、観察により得られた操作物体の軌道を実現可能な軌道に修正することができる。



(a) 接触を維持する動きの軌跡　　(b)C-space 内での表現

Fig. 2.9: C-obstacle

2.3 実行時の誤り検出

前節までの手法により、観測による誤差を補正することができる。しかし、計算機の持つ物体の内部モデルと実際のモデルには誤差があり、また多関節多指ハンドのような自由度の大きい把持システムを使う際には実行時に誤差を生じる。そのため実行時にこれらを補正する必要がある。

このため斎藤 [10] は分節化した状態遷移図を作成して、力センサによって状態を把握し、それに基づいて動作の補正を行う手法を提案している。しかしながら、この手法では状態の定義の方法が曖昧であり、また状態を把握するために必要な、物体への力の変化をあらかじめ知っておかなければならず、汎用性に乏しい。

松岡 [8] は多関節多指ハンドのみによる作業において、作業途中の状態の遷移を定義し、遷移に動作プリミティブを割り当てることで作業を実行し、実行時は状態の変化を観察してエラーリカバリを行う手法を提案している。この手法も [10] と同様に、状態の切り分けが自動化されておらず、そのため人があらかじめ定義した作業のみにしか適用できないものである。

2.4 問題提起

前節までで、観察を利用した、ロボットへの教示やエラーリカバリの手法の幾つかを概観したが、それぞれ一長一短である。そこで、より良いロボットスキルの獲得に必要な要素を整理したい。

まず第一に、作業状態がシンボリックに表現されていなければならない。このことは作業の目的状態を定義し、また、目的状態に達するための動作を定義することを可能にする。

次に、人間動作を手本として、作業方法を取得できなければならない。このことは、障

害物回避のための計算や操作物体の扱い方の解析を行う必要をなくし、実装を容易にする。

最後に、実行時にエラーが発生した場合の修正を行えなければならない。これは、状態がシンボリックに表現されていれば、目的状態を達成するための動作記述の中に組み込めるものである。

まとめると、

1. 状態のシンボリックな表現と目的状態の取得

2. 人間動作の模倣による作業方法の取得

を同時に満たすようなシステムが望ましいロボットスキルの獲得システムと言うことができる。

そこで、本研究ではこの条件をそれぞれ満たしている、[1] と [4] にとくに注目し、二つの手法を組み合わせることで二つの条件を満たす。すなわち、[4] の手法によって得られた軌道情報を [1] に類似の状態分類によって分割し、途中状態を生成する。そして、その途中状態を逐次達成するように動作記述を割り当てることによって、全体としてロボットスキルを獲得するようにするのである。これによって、先の二つの条件を同時に満たすシステムを構築する。

第3章　接触状態解析

本研究でのスキルの獲得は、

1. 手本動作軌跡の獲得
2. 接触状態に基づく軌跡の分割
3. セグメントへのサブスキルの割り当て

の3段階を経て行われる。本章ではサブスキルを割り当てる前処理としての、動作軌跡の獲得手法と接触状態解析について述べる。

3.1　人間動作軌跡の獲得

人間動作軌跡の獲得は、2章で紹介した[4]の手法に基づいて行われる。

オペレーターがステレオカメラシステムの前で組立作業を実行すると、システムはその作業を距離画像のシーケンスとして記録する。撮像システムは3眼のステレオカメラを利用しており、それぞれのカメラで得られた動画から距離画像のシーケンスを生成する。距離画像の精度を向上させるため、撮像の際には物体に縞のパターンを投影している。システムはあらかじめ扱う物体のCADモデルを与えられており、得られた距離画像中の操作物体に対して、このCADモデルを使ってテンプレートマッチングを行い、その動きの軌跡を記録する。

しかし、ステレオシステムによって得られた軌跡のデータは、観測誤差を含んでおり、そのままロボットに与えても動作を実行することは難しい。そのため、何らかの手法で誤差を含んだ軌道データを実行可能なものに修正する必要がある。そこで、configuration space(C-space)を使った軌道修正を行う。

得られた動きの軌跡はまずC-spaceを使って表現される。C-spaceは2章で説明したように、位置と姿勢を同時に表すことの出来る多次元空間であり、3次元空間の物体の位置姿勢は6次元のC-spaceによって表現される。すなわち、位置を表現するのに3次元、姿勢を表現するのに3次元を割り当てられた6次元空間である。実際には距離画像の軌跡は、ある一定の時間間隔をおいた距離データの並びで与えられているから、システムはその各観測点における位置と姿勢をC-space中に点として表現する。

次に、得られたC-space中の動作軌跡を実行可能なものに補正する。C-space中には操作物体の存在し得ない領域がマニフォールドとして表現されている(C-obstacle)。この領

域は環境物体と操作物体が重なって存在することはあり得ないことに起因する。したがって、このマニフォールドを構成する境界の曲面 (C-surface) は操作物体と環境物体が接触を持つために、操作物体の位置・姿勢の値が満たすべき条件を表していると考えることが出来る。そこで、このことを利用して、接触を持つと考えられる観測点を C-surface 上に投影することによって軌道を修正することが出来る。

具体的には、C-space 中の観測点の中で、C-surface からの距離がある閾値より小さいものと、C-Obstacle の中に含まれている観測点に関しては、操作物体が環境物体に対して接触を持っているとみなして、C-surface 上に投影する。投影は、観測された接触点の C-Space 内の点 $o_i(x, y, \theta)$ と、C-Surface C_j に対して、o_i に最も近い C_j 上の点を計算することで行う。これより真の値の推定値 $t_i(x, y, \theta)$ を計算することができる。このとき観測誤差は

$$E(o_i, t_i) = \|t_i - o_i\| \tag{3.1}$$

であると考えられる。

基本的には以上の手法で実行可能な動作軌跡を得ることが出来るが、たとえば観測点が Fig.3.1 のようになっているとき、生成される軌道は Fig.3.2 の実線で示されたような、環境物体の中を通るものになってしまい、正しい軌道が生成されない。そこで、Fig.3.2 の点線部で示されるような軌道を描くために、C-surface の境界を求めなくてはならない。

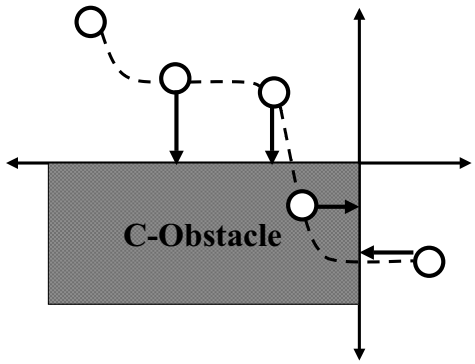


Fig. 3.1: 観測される軌道

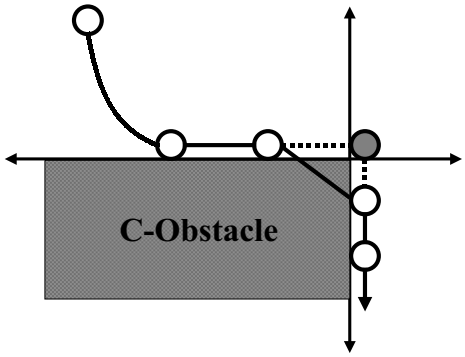


Fig. 3.2: 生成される軌道

そこで Fig.3.3 のようにして、C-Surface の交点を見つける。

すなわち、C-Surface C_i とそれに続く C_{i+1} があり、その交点近傍に観測点 o_i があったとすると、以下のようにして交点を計算する。

- o_i を C_i に投影する。この点を q_i とする。
- q_i を C_{i+1} に投影する。この点を q_{i+1} とする。
- $q_i - q_{i+1} < \delta$(閾値) なら q_{i+1} を交点とする。そうでなければ繰り返し。

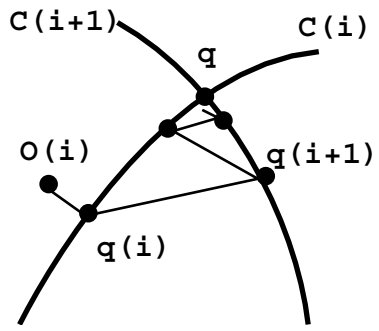


Fig. 3.3: C-Surface の交点

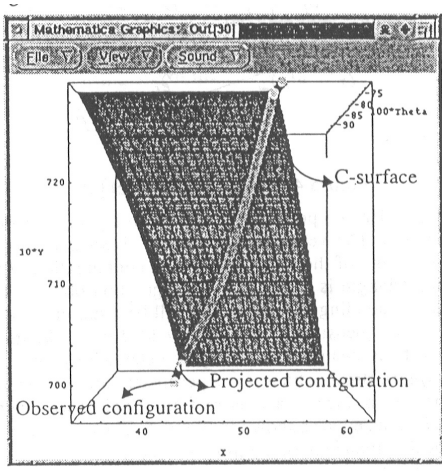


Fig. 3.4: C-Surface と補正された軌道

このようにして生成された C-Surface と補正された軌道の様子を Fig.3.4 に示す。

以上の手法によって、人間動作を元にした解析的に実行可能な動作軌跡を取得することが出来た。

3.2 接触状態の解析

前節で得られた軌道は、理想的な環境、すなわち CAD モデルと実際の物体との間に誤差がなく、与えられた位置・姿勢を正確に再現できる環境でのみ実行が可能である。しかし、現実にはそのような環境を作り出すことは難しく、特に多関節多指ハンドを使ったマニピュレーションシステムにおいては、物体をロボットに対して固定することは困難であるという実情があり、実行時の軌道修正が必要となってくる。

そこで、得られた軌跡を物体の接触状態によって分割し、分割されたセグメント毎に達成しなくてはならない目標状態を解析する。そして、セグメントに対してサブスキルと呼ぶ動作記述を割り当てることによって中間状態を達成していき、全体としてロボット動作スキルを獲得する手法を提案する。

本節ではこのために必要な、軌道の分割方法を説明する。

3.2.1 接触方程式

2 章でも紹介したように、作業状態を記述するための特徴量としてはさまざまなものが提案されてきた。その中でも、最も汎用性があり、一般的な作業の記述に適していると思われるものは、作業物体の拘束条件がどのようにな状態であるかを、各自由度に対して調

べる方法である。本研究でも一般的な作業を扱うために、この作業物体の拘束を元に状態を記述する。

操作物体と環境物体の接触による拘束条件は、複数の接触方程式によって表現される。[1] では式 2.1 のような接触方程式を使っている。しかし、この接触方程式は面接触を前提にしており、点と面の接触や回転の拘束を表現するには不充分である。そこでスクリュー表現を利用した接触方程式の形を採用する。

スクリュー理論

スクリューを使った接触方程式の表現について論じるためには、まずスクリュー理論について知っておく必要があるので、ここで簡単に説明する。[13][14]

スクリューとは物体の平行移動と回転移動を表す表記法の一つで、二つの 3 次元ベクトル $[\mathbf{S_0}, \mathbf{S_1}]$ で構成される。$\mathbf{S_0}$ は screw axis と呼ばれ、この軸に沿って並進運動が行われる。$\mathbf{S_1} = \mathbf{P} \times \mathbf{S_0} + p\mathbf{S_0}$ であり、$\mathbf{P}$ は原点から screw axis へのベクトル、p はピッチ、すなわちねじ山の幅で、並進量に対する回転の量を決定する。

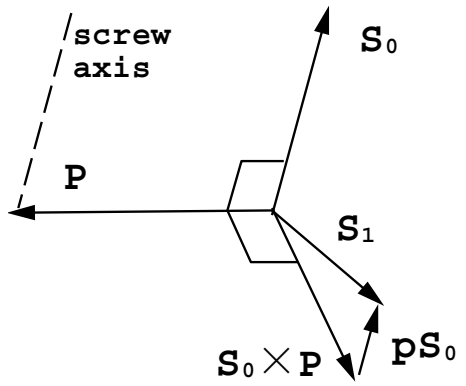


Fig. 3.5: スクリュー

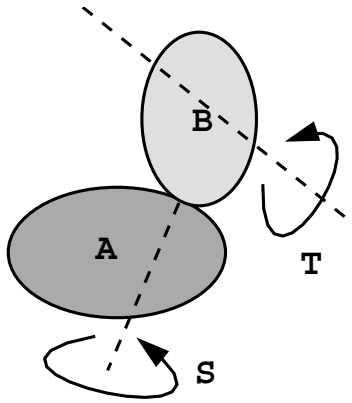


Fig. 3.6: スクリューによる接触表現

並進を伴わない回転運動に対しては $p = 0$ であり、$\mathbf{S} = [\mathbf{S_0}, \mathbf{S_0} \times \mathbf{P}]$ で表される。また、回転を伴わない並進運動に対しては、p は無限大となり、そのため、$\mathbf{S} = [0, \mathbf{S_0}]$ となる。

接触方程式のスクリュー表現

スクリューを使うと Fig.3.6 のような接触があった場合、line of contact を $\mathbf{S} = [s_1, s_2, s_3, s_4, s_5, s_6]$ とし、可能な動きを $\mathbf{T} = [t_1, t_2, t_3, t_4, t_5, t_6]$ で表すと、これらの関係は、式 3.2 を満たす必要がある。これが、本研究で使用する接触方程式である。この方程式は回転を伴わない並

進運動に対しては式 2.1 と等しくなる。

$$s_1t_4 + s_2t_5 + s_3t_6 + s_4t_1 + s_5t_2 + s_6t_3 \geq 0 \tag{3.2}$$

ここで、扱う物体を多面体に限るとすると、3 次元の物体では Fig.3.7 に示されるような 9 種類の接触を考えることができる。すなわち、face-face, face-edge, face-vertex, edge-face, edge-edge, edge-vertex, vertex-face, vertex-edge, vertex-vertex の 9 種類である。これらの

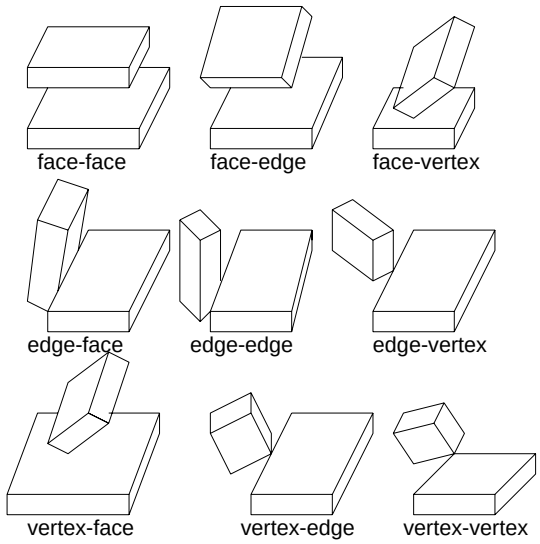


Fig. 3.7: 3 次元物体の接触

命名法は、例えば face-vertex contact であれば、操作物体の一つの面が、環境物体の一つの頂点に対して接触を持っている状態を表している。

これら 9 種類の接触は、vertex-face, face-vertex, edge-edge の 3 種類の基本的な接触の組合わせで表現される。例えば、face-dege contact は二つの face-vertex contact によって表現でき、vertex-vertex contact は三つの vertex-face contact で表現できるなどである。従って、接触方程式のスクリュー表現がどのようになるかは、この 3 種類の基本接触に対してのみ、解析を行えばよい。以下、順に解析を行う。

vertex-face contact は、Fig.3.8 に示されるような接触のことをいう。二つの物体が点 $p(p_x, p_y, p_z)$ で接触し、その位置での放線ベクトルが $\mathbf{n}(n_x, n_y, n_z)$ だとすると、この場合の接触方程式はベクトル $(\mathbf{n}, \mathbf{p} \times \mathbf{n})$ を接触方程式 3.2 の $(s_1, s_2, s_3, s_4, s_5, s_6)$ に代入することで得られる。ゆえに、接触方程式は、

$$n_xt_4 + n_yt_5 + n_zt_6 + (p_yn_z - p_zn_y)t_1 + (p_zn_x - p_xn_z)t_2 + (p_xn_y - p_yn_x)t_3 \geq 0 \tag{3.3}$$

で表される。

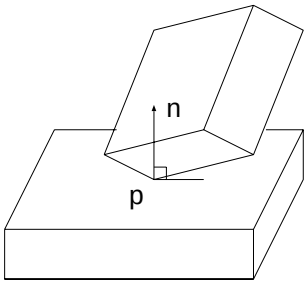


Fig. 3.8: vertex-face contact

face-vertex contact は、Fig.3.9 に示されるような接触のことをいう。同様にして、二つの物体が点 $p(p_x, p_y, p_z)$ で接触し、その位置での放線ベクトルが $\mathbf{n}(n_x, n_y, n_z)$ だとすると、この場合の接触方程式は、

$$-n_x t_4 - n_y t_5 - n_z t_6 - (p_y n_z - p_z n_y)t_1 - (p_z n_x - p_x n_z)t_2 - (p_x n_y - p_y n_x)t_3 \geq 0 \qquad (3.4)$$

で表される。

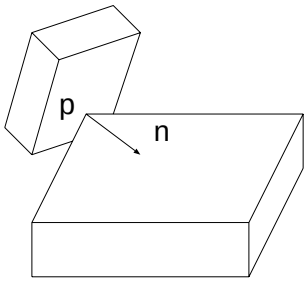


Fig. 3.9: face-vertex contact

最後に **edge-edge contact** を考える。これは Fig.3.10 に示されるような接触をいう。二つの物体の接触点は前と同様に点 $p(p_x, p_y, p_z)$ にとり、エッジ e_1 の方向をベクトル (e_{1x}, e_{1y}, e_{1z})、エッジ e_2 の方向をベクトル (e_{2x}, e_{2y}, e_{2z}) で表すことにすると、放線ベクトル $\mathbf{n}(n_x, n_y, n_z)$ は二つのベクトル (e_{1x}, e_{1y}, e_{1z}) と (e_{2x}, e_{2y}, e_{2z}) に垂直な方向で、環境物体の外側を向くベクトルとして定義される。これを利用するとこの場合の接触方程式は式 3.5 のようになる。

$$n_x t_4 + n_y t_5 + n_z t_6 + (p_y n_z - p_z n_y)t_1 + (p_z n_x - p_x n_z)t_2 + (p_x n_y - p_y n_x)t_3 \geq 0 \qquad (3.5)$$

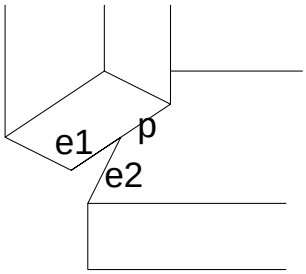


Fig. 3.10: edge-edge contact

3.2.2 接触状態の分類

接触状態を分類するためには、先の接触方程式を扱う物体の全ての接触に対してたて、それらを連立させて解空間を求める。そして、解空間の形によって接触状態を分類する。

Polyhedral convex cones

解空間の形は凸多面錐 (polyhedral convex cones:PCC) の形で表現される。例として、m 個の線形な連立不等式 $AX \geq 0$ を考える（式3.6)。ここに、X は n 次元のベクトルである。この連立不等式の解空間は A^* で表され、これを polyhedral convex cones と呼ぶ。[9]

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ & \ddots & \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \geq \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \tag{3.6}$$

PCC を扱うために、まず face という概念について説明する。PCC $A^* = \{X | A_1X \geq 0, \cdots, A_pX \geq 0\}$ に対して、$1, \cdots, p$ のサブセット H を考える。H は A^* のサブセット F_H に対応する。face F_H は H に含まれる h に対しての $A_hX > 0$ という式と、H に含まれない h によって作られる $A_hX = 0$ によって定義される。$A_hX > 0$ を満たす全てのベクトル X によって作られる解空間を O_H、$A_hX = 0$ を満たす全てのベクトル X からなる解空間を L_H とすれば、$F_H = O_H \cap L_H$ である。

H は 2^p 通りの選び方があり、空でない face は重なりを持たないので、A^* からは空である face を含むと 2^p 個の face が作れることがわかる。

また L_H の次元は、$d_H = n - r_H$ で与えられる。ここに r_H は L_H に対応する式のうち、線形で独立な方程式の最大数である。このとき d_H を F_H における face の次元と定義する。

接触方程式への応用

接触方程式は式 3.2 で表されるような不等式であるので、全ての接触を考えると、式 3.7 のような形で表現することができる。

$$\begin{pmatrix} s_{11} & \cdots & s_{16} \\ & \ddots & \\ s_{m1} & \cdots & s_{m6} \end{pmatrix} \begin{pmatrix} t_1 \\ \vdots \\ t_6 \end{pmatrix} \geq \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \tag{3.7}$$

ここに 6 次元ベクトル $(t_1, \cdots, t_6)$ は、操作物体の動きを表すスクリューベクトルであるので、$(t_1, \cdots, t_6)$ の解空間が可能な動きベクトルを表す。そこで、行列 S_{mn} のランクと face の次元を使って解空間の形を分類する。ここで、行列のランクは独立した境界の数を、face の次元は解空間の形状の次元を表すと考えて良い。

ここで Face の次元の範囲の求め方を簡単に記す。まず、式 3.7 を式 3.8 の形に置き換える。

$$\begin{pmatrix} s_{11} & \cdots & s_{16} \\ & \ddots & \\ s_{m1} & \cdots & s_{m6} \end{pmatrix} \begin{pmatrix} t_1 \\ \vdots \\ t_6 \end{pmatrix} = \begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix} \quad (\forall i, u_i \geq 0) \tag{3.8}$$

次に、各行の式の中で独立ではない式の組を探し出す。もしそのような組が存在したとすると、$c_1 u_{i1} + \cdots + c_l u_{il} = u_j$ のような式を立てることができる。そして、この時 $\forall k, c_k < 0$ または $u_{ik} = 0$ ならば、$\forall k, u_{ik} = 0$ かつ $u_j = 0$ であると言うことができる。

これを $u_i = 0$ になるような式が増えなくなるまで繰り返す。すると、式 3.8 は $u_i = 0$ になる式で作られる部分と $u_i \geq 0$ になる式で作られる部分に分けられる。このとき前者に対応する行列のランクを r_{eq} とすれば、face の次元は最大で $6 - r_{eq}$ となる。また、行列 S_{mn} のランクを r_{all} とすれば face の次元の最小は $6 - r_{all}$ である。

このことを使って、まず、並進運動に関して状態を分類する。並進運動の際には $t_1 = t_2 = t_3 = 0$ であるから、式 3.9 について考えれば可能な並進運動が求まることになる。

$$\begin{pmatrix} s_{14} & s_{15} & s_{16} \\ & \vdots & \\ s_{m4} & s_{m5} & s_{m6} \end{pmatrix} \begin{pmatrix} t_4 \\ t_5 \\ t_6 \end{pmatrix} \geq \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \tag{3.9}$$

この式 3.9 に関して、ランクと face の次元に対する解空間の形を Table3.1 にまとめる。

ここに「四半空間」は平行でない二つの平面によって区切られる、四つの部分のうちの一つをさし、1/8 空間は平行でない三つの平面によって区切られる八つの部分のうちの一つ、などとする。

この時、$3-(rank)$ はその状態を維持するような並進運動の自由度数 (maintaining DOF) であり、(face の次元の最大)−(face の次元の最小) はその状態から他の状態へ変化する並進運動の自由度 (detaching DOF) の数、3−(maintaining DOF の数)−(detaching DOF の数)

Table 3.1: 並進運動の分類表

rank	face の次元	解空間の形状
0	3	全空間
1	2	平面
	2,3	半空間
2	1	直線
	1,2	半平面
	1,2,3	四半空間
3	0	点
	0,1	半直線
	0,1,2	四半平面
	0,1,2,3	1/8 空間

が、その方向には動くことのできない自由度 (constraining DOF) の数を表す。例えば、解空間が半平面を取る場合、rank は 2 であるから、maintaining DOF の数は 1、face の次元の範囲は 1 から 2 までが取り得るから、detaching DOF の数は 1、そして constraining DOF の数は 1 として求まる。これは直感的な理解とも一致する。

次に、回転運動に関しての分類を行う。回転運動はまず、その軸を回転軸として左右両回転ができる軸の本数と片側回転だけができる回転軸の本数を求める。

まず、式 3.8 を変形して式 3.10 の形にする。

$$\begin{pmatrix} s_{14} & s_{15} & s_{16} \\ & \vdots & \\ s_{m4} & s_{m5} & s_{m6} \end{pmatrix} \begin{pmatrix} t_4 \\ t_5 \\ t_6 \end{pmatrix} = \begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix} - \begin{pmatrix} s_{11} & s_{12} & s_{13} \\ & \vdots & \\ s_{m1} & s_{m2} & s_{m3} \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} \tag{3.10}$$

ここで、$\begin{pmatrix} s_{14} & s_{15} & s_{16} \\ & \vdots & \\ s_{m4} & s_{m5} & s_{m6} \end{pmatrix}$ を S で表すことにする。

ここで S の行に対して独立でない組を探し出す。すると、u_i, t_1, t_2 および t_3 だけを含む方程式を立てることができる。ここで、そのような方程式が

$$d_1 t_1 + d_2 t_2 + d_3 t_3 = u_j - c_1 u_{k1} - \cdots - c_l u_{kl} \tag{3.11}$$

のように表されているとき、もしも $\forall m, c_m < 0 \quad$ または $\quad u_{km} = 0$ ならば、不等式 $d_1 t_1 + d_2 t_2 + d_3 t_3 \geq 0$ を得ることができ、またもし、$u_j = 0 \quad$ かつ $\quad \forall m, c_m > 0 \quad$ または $\quad u_{km} = 0$ ならば、不等式 $d_1 t_1 + d_2 t_2 + d_3 t_3 \leq 0$ を得ることができる。

これを繰り返すことによって、式の形は式 3.9 と同じ形となり、回転軸に関して、ランクと face の次元の範囲を求めることができるようになる。

これによって Table.3.2 に示されるように、回転軸の本数を計算することができる。

Table 3.2: 回転運動の分類表

rank	face の次元	両回転できる軸の数	片回転できる軸の数
0	3	3	0
1	2	2	0
	2,3	2	1
2	1	1	0
	1,2	1	1
	1,2,3	1	2
3	0	0	0
	0,1	0	1
	0,1,2	0	2
	0,1,2,3	0	3

この結果を用いて、回転の constraining DOF の数を 3−(両回転できる軸の数) −(片回転できる軸の数) で定義し、回転の maintaining DOF の数を 6−(rank)−(並進の maintaining DOF の数)、回転の detaching DOF の数を 3−(maintaining DOF の数)−(constraining DOF の数) と定義する。このように一見複雑な定義の仕方をしたのは maintaining DOF の回転の軸は、その位置によっては操作物体を環境物体から離脱させる運動の回転軸になり得るためである。このような現象は、無限遠に回転軸の位置をとれば並進運動も回転運動の一種となることに原因があるものである。

3.2.3 特異点の扱い

ここまでの分類で、作業中の物体の接触状態の基本的な部分は網羅できた。しかし、作業の中には物体の可動範囲が polyhedral convex cones の形にならない接触が起こり得る。例えば、Fig.3.11 では線 a の上側と線 b の右側が動ける範囲になるが、これは明らかに PCC ではない。このような接触点は解析上の特異点であり、これを singular contact と呼ぶことにする。

Singular contact は、大域的 singular contact と局所的 singular contact に分類される。局所的とは、個々の接触の状態だけで見ることを、大域的とは、個々ではなく全体の接触を考えた上で見ることを意味する。大域的 singular contact と局所的 singular contact は、そ

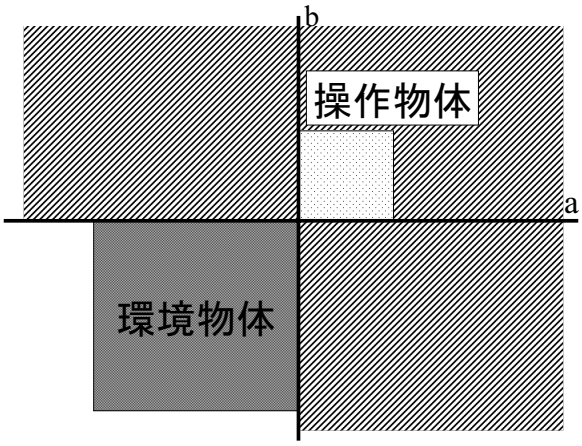


Fig. 3.11: 可動範囲が PCC にならない場合

れぞれ、大域的、局所的に見て、運動物体の可動範囲が PCC の形にならない場合の接触の事をさす。

まず、局所的 singular contact について詳しく説明する。

局所的 singular contact には、Fig.3.12 から Fig.3.14 に示すような、凸型の頂点同士の接触、凸型の頂点と凸型の辺の接触、凸型の辺同士で且つそれぞれの辺が平行である接触が考えられる。

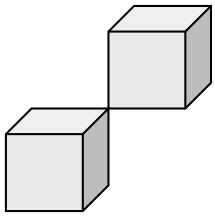


Fig. 3.12: 凸型頂点同士の接触

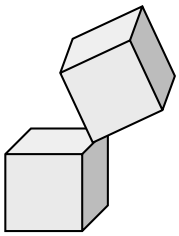


Fig. 3.13: 凸型頂点と凸型の辺の接触

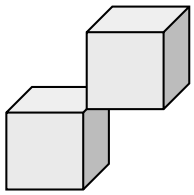


Fig. 3.14: 平行な凸型辺同士の接触

これら局所的 singular contact における拘束不等式は、

$$\begin{aligned} s_{11}T_1 + s_{12}T_2 + \cdots + s_{16}T_6 \geq 0 \\ \vdots \\ s_{n1}T_1 + s_{n2}T_2 + \cdots + s_{n6}T_6 \geq 0 \end{aligned} \tag{3.12}$$

というPCCの形にはならず、

$$\begin{aligned} s_{11}T_1 + s_{12}T_2 + \cdots + s_{16}T_6 \geq 0 \quad or \\ s_{21}T_1 + s_{22}T_2 + \cdots + s_{26}T_6 \geq 0 \quad or \\ \vdots \\ s_{n1}T_1 + s_{n2}T_2 + \cdots + s_{n6}T_6 \geq 0 \end{aligned} \tag{3.13}$$

の形で表現される。

次に、大域的singular contactについて述べる。大域的singular contactは全ての接触方程式を考慮したときになお、その解空間がPCCの形にならないsingular contactのことをさす。これは解析上の特異点として作業の実行の際にも特別な扱いをする必要がある。

一方、局所的にはsingular contactを持つが、全ての接触方程式を考慮するとその解空間の形状がPCCで表すことのできるものが存在する。例としてFig.3.15で示したような接触状態を考える。この場合は、図中の矢印で示した接触が局所的なsingular contactになるが、その他の接触によって、物体の可動範囲が制限されているために、最終的な可動範囲はPCCの形をとる。

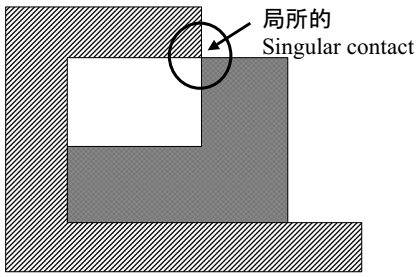


Fig. 3.15: 大域的にはsingular contactにならない接触例

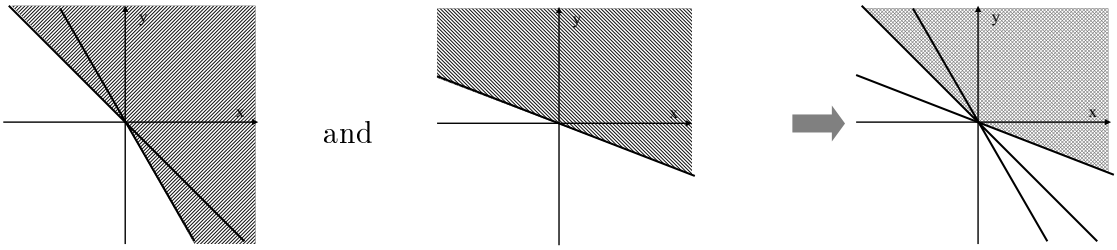
このように大域的にはsingular contactにならない接触は、作業で特別な扱いをする必要はないので、ここではこのように局所的singular contactを持っても大域的なsingular contactではないような状態をどのようにして発見するかについて述べる。

簡単のため、2次元で考える。今,物体の拘束方程式が式3.14および3.15で与えられているとする。

$$x + y \geq 0 \quad or \quad 2x + y \geq 0 \tag{3.14}$$

$$x + 2y \geq 0 \tag{3.15}$$

式 3.14 のみで見たときの解空間の形は PCC の形をとらないので、見た目上、解領域の形は PCC にならないように思われるが、式 3.15 までを考えると Fig.3.16 のように PCC の形になることがわかる。



$x+y \geq 0$ or $2x+y \geq 0$ $\qquad$ $x+2y \geq 0$

Fig. 3.16: 大域的には singular contact でない場合の解空間

式 3.14 における解領域は式 3.17 に示す、3 つの領域に分割することができる。

$$\begin{aligned} &x+y \geq 0 \ \ and \ \ 2x+y \geq 0 \\ &x+y \geq 0 \ \ and \ \ 2x+y < 0 \\ &x+y < 0 \ \ and \ \ 2x+y \geq 0 \end{aligned} \tag{3.16}$$

解領域が PCC にならないためには、ほかの不等式の解領域も考慮したときに、下 2 つの領域が空でないことが必要十分である。すなわち、解領域が PCC であることを見つけるためには下の 2 つの解領域が空であることを示せばよい。

これは face の次元を求めた際に、解空間が空であるかどうかを調べた方法と同様にして求めることができる。例えば、この場合だと $x+y<0 \ \ and \ \ 2x+y \geq 0$ の領域が、空でないとする、すなわち解を持つとすると、

$$x+y = \frac{1}{3}(2x+y) + \frac{1}{3}(x+2y) \geq 0 \tag{3.17}$$

となり、矛盾する。よって、解領域は空となり、解領域は polyhedral convex cones となる。この方法を用いて、局所的 singular contact で大域的 singular contact でないものを発見することができる。

このようにして、大域的に singular contact になる場合を見つけることができた。以後、単に singular contact といった場合には、大域的に singular contact である場合をさすものとする。

次に、singular contactである場合の扱いについて考える。

Fig.3.17に示した2つのsingular contactから、矢印の方向へ挿入を行う場合を考えてみる。Fig.3.17左の場合は、操作物体の位置を穴の位置に正確に持ってくることは、ビジュアルフィードバックを使うなどしなければ困難である。しかし、Fig.3.17左の場合であれば、右側の環境物体に操作物体を突き当てるだけで、操作物体は環境物体の穴の位置に持ってくることが可能である。そのため、この2つの状態は区別して考えなければならない。

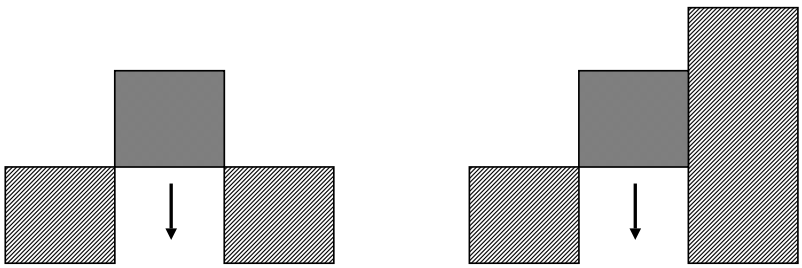


Fig. 3.17: 異なった扱いの必要なsingular contact

このようなsingular contactの中での接触の区別を行うために、局所的singular contactの接触を除いた接触方程式で考えたときのmaintaining DOF, detaching DOF, constraining DOFを調べて区別することにする。これらの自由度のことを以後、それぞれsingular maintaining DOF, singular detaching DOF, singular constraining DOFと呼ぶことにする。

例えば、Fig.3.17左の場合、並進運動におけるsingular maintaining DOF=2、singular detaching DOF=0、singular constraining DOF=0となり、右の場合、並進運動におけるsingular maintaining DOF=1、singular detaching DOF=1、singular constraining DOF=0ということになる。

最後に、束縛自由度という指標について説明する。束縛自由度とは、接触方程式3.7において、faceの次元の範囲を求める際に$A_h X = 0$となる部分の行列のランクを求めることで求まるもので、拘束のある自由度の数である。これは、通常の接触点でも特異点でも同様に定義できるので、特異点が関係する遷移で、並進運動と回転運動のどちらの動作を選ぶべきかを選択するときの指標となる。

以上の議論によって、作業中の物体の接触状態を分類するための、特徴量が定義できた。すなわち、並進運動のmaintaining, detaching, constrainingの3種類の自由度、回転運動のmaintaining, detaching, constraining DOF、singular contactの際の並進、回転運動の自由度、および、並進、回転に関する束縛自由度である。

3.3 接触状態に基づく軌跡の分割

3.2 節で状態を表現するための道具立てが揃ったので、3.1 節で得られた、人間の動作軌道を物体の接触状態を元にして分割し、状態遷移図を作ることができる。

例として、Fig.3.18 のような、2 次元平面内の作業における挿入動作について考えてみる。この動作をカメラの前で実演したあと、C-Space 内での解析によって、実行可能な軌跡を計算する。

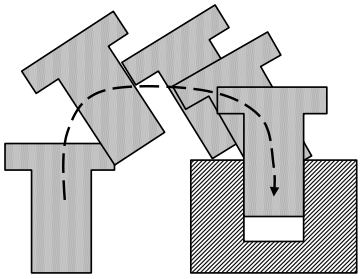


Fig. 3.18: 二次元の挿入動作の軌跡

この得られた軌跡に対して、全ての接触の接触方程式を考慮し、前述の手法で中間状態の接触状態を記述して、軌跡を分割した結果、えられた状態遷移図が Fig.3.19 である。

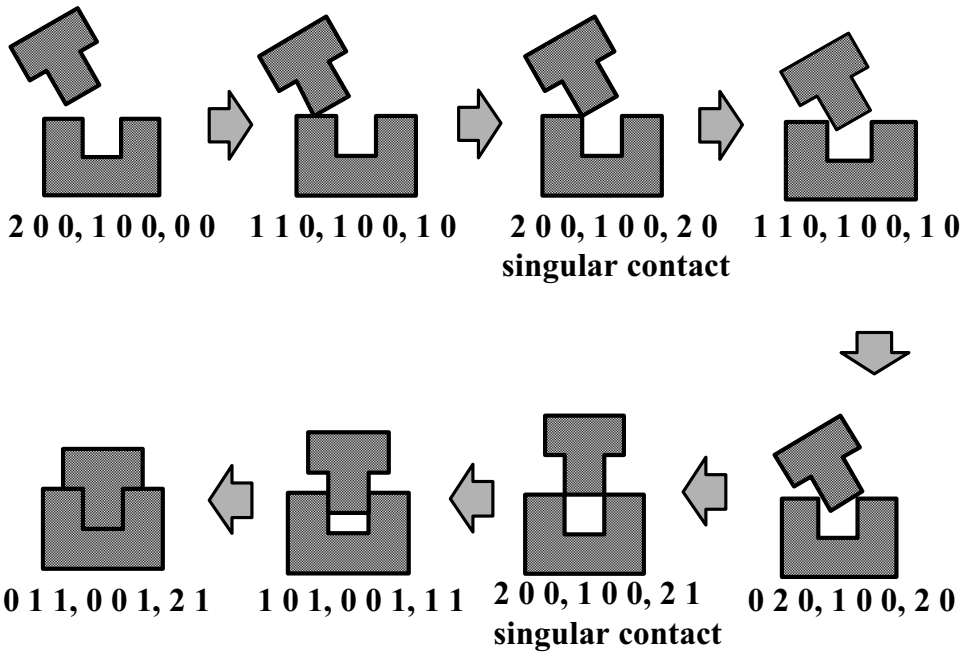


Fig. 3.19: 状態遷移図

図中の数字は、左から並進運動における maintaining DOF,detaching DOF, constraining DOF、回転運動における maintaining DOF,detaching DOF, constraining DOF、並進運動における束縛自由度、回転運動における束縛自由度を表している。

次章では、このような状態変化が起こるためには、実際にどのような動作を行えば良いのかを述べ、必要となるサブスキルを定義する。

第4章　サブスキル

3 章では、作業の際に物体の接触状態がどのように変化するかを解析する手法について述べた。本章ではその状態変化が起きるためには、どのような動作が必要となるのかを検討し、状態遷移図に割り付けるべき動作記述である、サブスキルを定義する。

4.1　サブスキル

サブスキルとは、作業全体を行う「スキル」に対して、作業の途中状態を実現していくために使われる動作要素のことをさす。従って、スキルはサブスキルの適切な組合わせによって獲得される。

そこでまず、作業においてどのような状態変化が起こり得るかを論じ、そこからどのようなサブスキルが必要となるかを考察する。

作業途中の操作物体の分類は、各方向の自由度が maintaining, detaching, constraining のいずれであるかで行った。また、singular contact を除いては作業は拘束の増える方向、すなわち maintaining → detaching → constraining の方向に作業が進むという前提をおく。

このことから、並進運動、回転運動のそれぞれに対して

1. maintaining DOF から detaching DOF への変化
2. detaching DOF から constraining DOF への変化
3. maintaining DOF から constraining DOF への変化

がありえることがわかる。

次に singular contact を含んだ場合を考える。

singular maintaining DOF を持つ状態は、Fig.4.1 のように物体の頂点と頂点が接触している場合の状態に代表される。

この状態は大域的にも singular contact であるから、他の部分の接触の方程式によってこの状態を実現させることができない。そのため、一般的な maintaining DOF とは別に扱う必要がある。

singular detaching DOF を持つ状態は、Fig.4.2 のような状態である。これは局所的 singular contact であっても、大域的 singular contact ではないので、singular contact 以外の接触による接触方程式によってこの状態を実現することができる。したがって、singular detaching DOF は通常の detaching DOF と同じ扱い方で扱うことができる。

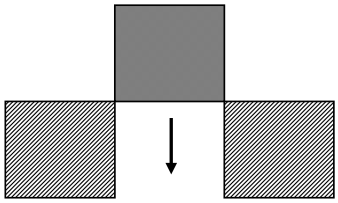


Fig. 4.1: singular maintaining DOF

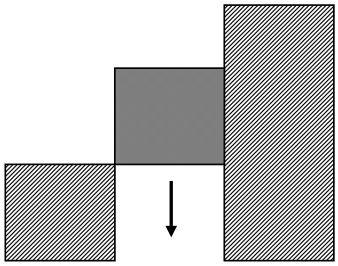


Fig. 4.2: singular detaching DOF

同様にして singular constraining DOF に関しても大域的 singular contact にはならないので、これも constraining DOF と同様に扱う。

従って、singular contact に関して、新たに起こり得る変化は

4. maintaining DOF と singular maintaining DOF 間の変化
5. singular maintaining DOF から detaching DOF への変化
6. singular maintaining DOF から constraining DOF への変化

と考えられる。

ここで上記の 3 の変化を再考してみると、maintaining DOF から constraining DOF への変化は必ず singular contact を経て行われる。すなわち maintaining DOF から constrainig DOF への変化は maintaing DOF $\rightarrow$ singular maintaining DOF $\rightarrow$ constrainingd DOF という変化で行われる。そのため、singular contact を考慮に入れると、3 の変化を考える必要はない。

同様に 2 の動作についても必ず singular detaching DOF を持つ状態を経るが、detaching に関しては singular と non-singular では区別の必要がないので、削除の必要はない。

これら 5 つの状態変化の中でも、直接動作呼び出しに関係する動作は 2 つだけである。その理由は次のように説明できる。組立作業では拘束が増す方向に作業が進むという仮定をおいているので、物体を動かすことを考えたとき、動ける方向は maintaining DOF の方向に限られる。なぜなら、detaching DOF の方向に動かすと拘束が減るので仮定に反し、coustraining DOF の方向に物体を動かすことはできないからである。また、そのとき動作の方向である maintaining DOF の方向は、singular maintaining DOF か detaching DOF にしか変化しない。

このことから、サブスキルを導き出すのに直接関係する変化は上記の変化のうち、1,4 の 2 つであり、残りの変化は 1,4 の動作を実行したときの副作用的なものであることがわかる。しかしながら、2,5,6 の変化が起こる動作が行われたことを知ることは、サブスキルの実行を確実に行うために必要なものであるから、サブスキルの設計を行う際にこれらの副次的な変化も考慮に入れなければならない。

まとめると、サブスキルを設計するのに基本となる動作は、

- maintaining DOF を detaching DOF に変化させる動作
- maintaining DOF と singular maintaining DOF 間の変化を起こす動作

であり、それと同時に達成されるべき変化が

- detaching DOF から constraining DOF への変化
- singular maintaining DOF から detaching DOF への変化
- singular maintaining DOF から constraining DOF への変化

となる。

以下にこのような変化を起こすサブスキルとはどのようなものが必要か、具体的に示す。

4.2 並進突き当て動作

並進運動に関して、maintaining DOF が detaching DOF に変化する遷移を考える。このような変化が起こる動作は Fig.4.3 で示されるように、ある方向に並進運動を行い、新たな接触が生じたところで停止する動作で、これを並進突き当て動作と呼ぶ。

突き当て動作の際に、動作方向ではない方向の接触条件は変わらない。そのため、動作方向でない方向の自由度の種類は maintaining, detaching, constraining の 3 種類が考えられる。maintaining であるときは、Fig.4.3 で示した。detaching, constraining の時は、それぞれ以下の Fig.4.4 および Fig.4.5 で示したような動作となる。

突き当て動作では、動作方向の自由度の種類は maintaining DOF から detaching DOF に変化するから、突き当ての動作方向は終了状態の detaching DOF の向きによって求められる。

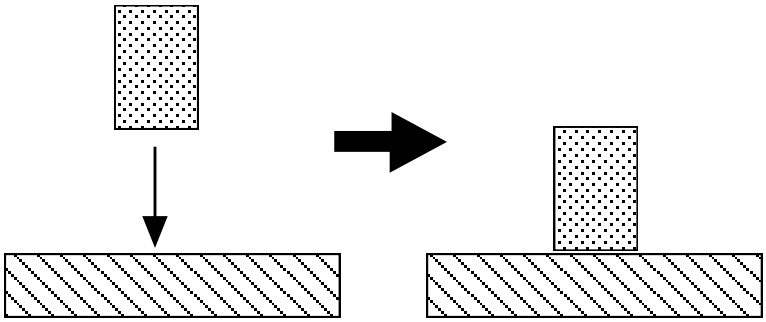


Fig. 4.3: 突き当て動作

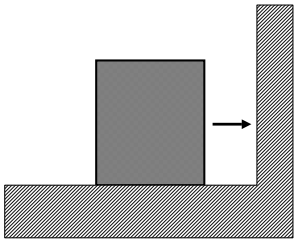


Fig. 4.4: detaching DOF を維持した突き当て

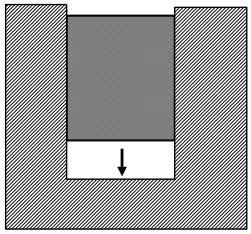


Fig. 4.5: constraining DOF を維持した突き当て

この動作で問題になるのは、動作方向でない方向の自由度の種類が maintaining DOF の場合である。なぜなら、この場合は障害物を回避して、目的の接触面まで操作物体を移動させるという要素が含まれるからである。一般に障害物の回避は CAD モデルの解析などで対応するが、本研究では人間動作を元に軌道が生成されているので、それを模倣することで、接触面まで操作物体を移動させることができる。

4.3 回転突き当て動作

回転動作について、maintaining DOF が detaching DOF に変化する動作が、回転突き当て動作である。

この動作は Fig.4.6 のように、接触状態を維持しながら（無接触を含む）、回転運動を行い、新たな接触が生じた時点で停止する動作である。

この動作の際、接触点を回転中心として回転運動が起る場合には問題がないが、無接触の状態から接触が起った場合には、解析上、並進と回転それぞれで maintaining DOF から detaching DOF への変化が同時に生じる。

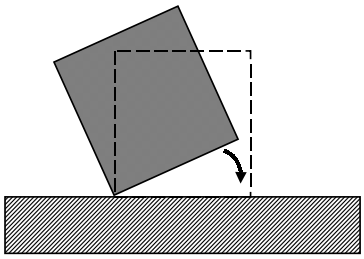


Fig. 4.6: 回転突き当て動作

そこで、このように同時に変化が起った場合には、並進突き当てを行ってから回転突き当てを行う、または、動作終了時の姿勢にまで回転運動を行ってから突き当てを行う、というように、動作を分解して作業を行う必要がある。(Fig.4.7) どちらの実装が適しているかは実装系によって異なる。

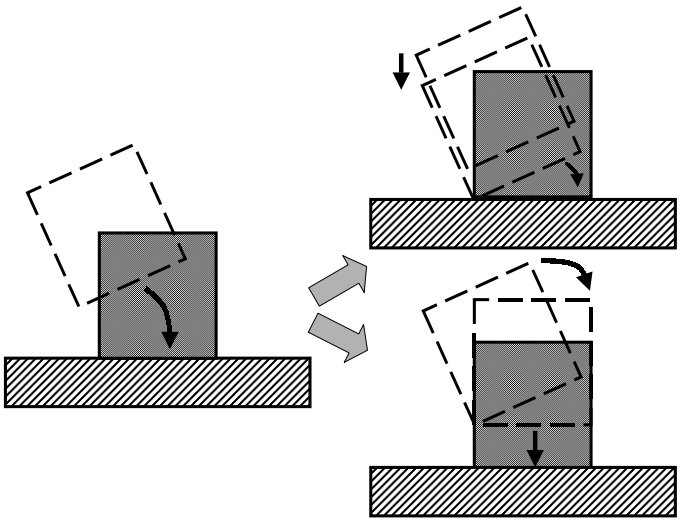


Fig. 4.7: 並進と回転の突き当て動作が同時に起る場合

4.4 並進滑らし動作

次に、並進運動において、maintaining DOF が singular maintaining DOF に変化する動作を考える。このような変化は、Fig.4.8 のような動作の際に起こると考えられる。これは、接触を保ったまま物体を滑らし、singular contact を達成した時点で動作を停止させる動作で、並進滑らし動作と呼ぶ。

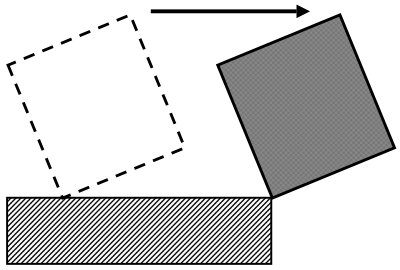


Fig. 4.8: 並進滑らし動作

また、singular maintaining DOF から maintaining DOF へ変化する場合もやはり並進滑らし動作となる。この動作の場合は前述した副次的な変化、すなわち

- detaching DOF から constraining DOF への変化
- singular maintaining DOF から detaching DOF への変化
- singular maintaining DOF から constraining DOF への変化

が起る。

まず、detaching DOF から constraining DOF へ変化するような滑らし動作は、Fig.4.9 のような動作を行うときで、これは singular contct を達成することが特別な意味を持つ動作ではなく、自動的に変化が達成される動作である。

しかし、Fig.4.10 のように singular maintaining DOF が detaching DOF に変化する動作や Fig.4.11 のような singular maintaining DOF が constraining DOF に変化する動作を行う際には、singular contact が達成されていないと、環境物体の縁への突き当てが起ってしまい、動作が妨げられる。特に、constraining DOF に変化する場合には、2 つの singular contact が確実に達成されていないと、次の動作に入れない。[12] ではこのように maintaining DOF が constraining DOF に変化する場合には、カメラやレーザーなどを使った位置の測定が必要になるとされているが、それは singular contact を達成することが、他の動作に比べて極めて難しいことに起因すると考えられる。

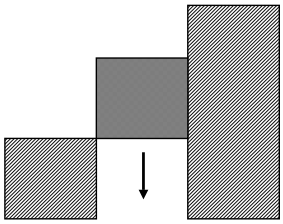


Fig. 4.9:

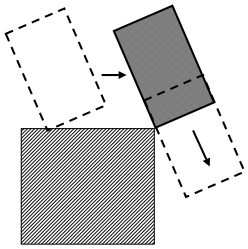


Fig. 4.10:

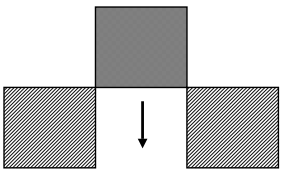


Fig. 4.11:

副次的な変化が起こる滑らし動作

4.5 回転滑らし動作

回転運動で maintaining DOF が singular maintaining DOF に変化する場合を考える。これは現在の接触を保ちながら、操作物体を滑らし、回転運動を行うもので、回転滑らし動作と呼ぶ。

まず、一点接触の回転滑らし動作を考える。一点接触での回転滑らし運動は本質的に並進滑らし運動を伴うものである。なぜなら、本研究では扱う物体を多面体と仮定しているため、回転突き当ての時のように接触点に回転中心がある場合を除いて、純粋な回転運動では接触を保つことができないからである。

そのため、このような変化は Fig.4.8 で示した、並進滑らし動作の際に同時に起こる。こ

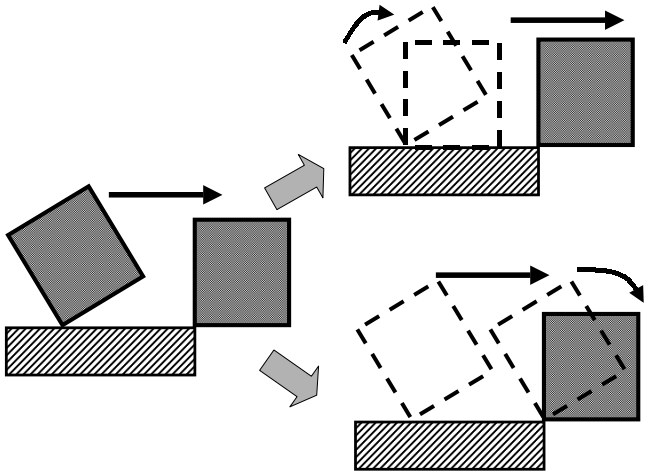


Fig. 4.12: 並進と回転の滑らし動作が同時に起こる場合

の場合は、突き当て動作で並進と回転運動が同時に起こるときのように、動作を分解し、まず回転運動を行ってから並進滑らし運動をする、または、並進運動をしてから、回転運動を行うという方法をとれば良い (Fig.4.12)。これが、一点で接触している場合の回転滑らし運動である。

次に 2 点で接触している場合の回転滑らし運動を考える。2 点で接触している場合には、接触点での放線の交わりが回転中心となる回転運動が起こる。

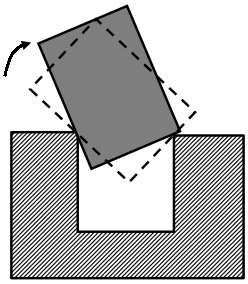


Fig. 4.13: 2 点接触の回転滑らし運動

この運動は、回転するにつれて回転中心が移動していくので、実装系によって制御の仕方は異なるが、回転中心を解析的に求めるか、接触を維持する動作と回転を行う動作に分解して考えることが必要となるであろう。この場合も、並進運動が同時に起こるが、束縛自由度の変化を使って、回転の束縛自由度が増えているときにこの動作を呼び出せば良い。

3 点以上の接触では基本的には、接触点からの放線が一点では交わらないので、回転滑らし運動は起こらない。接触放線が一点で交わる特殊な場合には回転滑らし運動が起こるが、これは 2 点の接触を保つ動作が行えれば、残りの接触点に関しては自動的に接触が成り立つものであるので、2 点接触の滑らし動作までを考えればよいことになる。

4.6 サブスキルと状態遷移図

前節までで、状態の変化とそれに対して割り付けるべきサブスキルの内容について述べた。そこで、3 章の最後に例として挙げた状態遷移図を使って、実際にサブスキルの割り付けを行ってみる。

1 番目の変化は、並進運動の maintaining DOF が detaching DOF に変化しているので、並進突き当てが割り当てられる。

2 番目の変化は、並進、回転運動の maintaining DOF が singular maintaining DOF に変化し、かつ、並進運動の束縛自由度が増え、回転運動の束縛自由度が変化していないので、並進滑らしが割り当てられる。

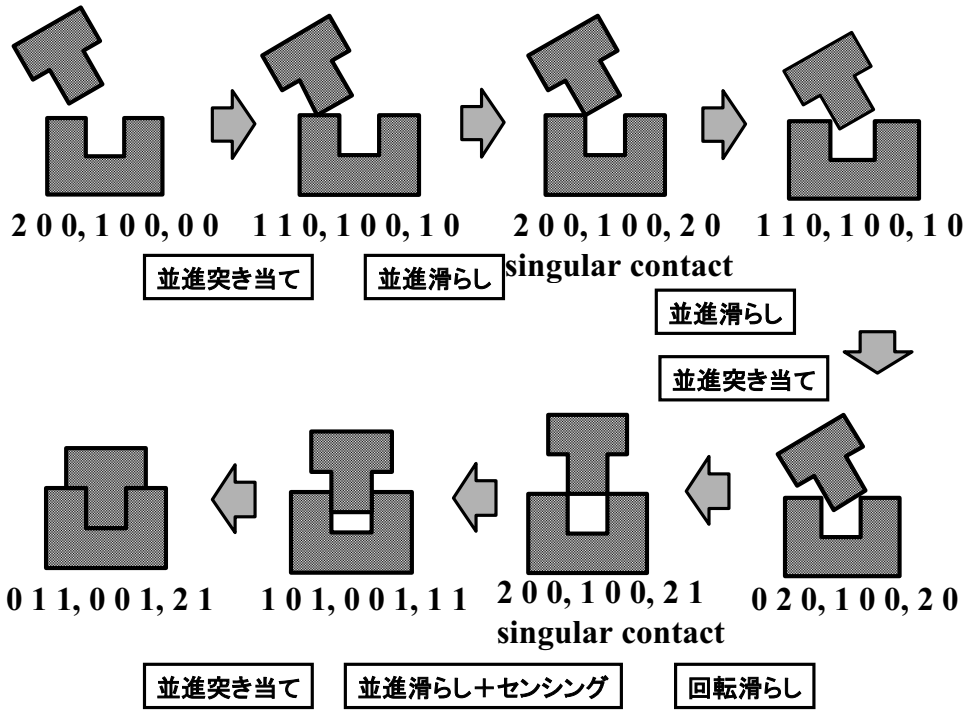


Fig. 4.14: 状態遷移図

3 番目の変化は、並進、回転運動の singular maintaining DOF が maintaining DOF に変化し、束縛自由度の変化は並進に関して変化しているので、並進滑らし動作が割り当てられる。また、singular maintaining DOF が detaching DOF に変化しているので、singular contact が確実に達成されていることが要求される。

4 番目の変化は、並進運動の maintaining DOF が detaching DOF に変化しているので、並進突き当て動作。

5 番目の変化は、並進運動の回転運動の maintaining DOF が singular maintaining DOF に変化していて、かつ回転運動の束縛自由度数が増えているので、回転滑らし動作。

6 番目の変化は、並進運動の singular maintaining DOF が maintaining DOF に変化し、束縛自由度が並進運動に関して変化しているので、並進滑らし。また、横方向の並進及び、回転運動において singular maintaining DOF が constraining DOF に変化しているので、その方向の位置合わせに正確さが要求される。

7 番目の変化は、並進運動の maintaining DOF が detaching DOF に変化しているので、並進突き当て動作が割り当てられる。

以上のようにサブスキルを割り当てることができたが、singular maintaining DOF が maintaining DOF に変化する際の滑らし動作は、微少距離を動かすことですぐに次の状態へ

と遷移していることがわかる。そのため、実装ではsingular maintaining DOFがmaintaining DOFに変化する場合、副次的な要素による補正動作のみを行い、次に呼び出されるサブスキルによって以降の動作を決定するなどの工夫が必要である。

次章では、これらのサブスキルをロボットへ実装し、作業が可能であるかどうかを確認する。

第5章　サブスキルの実装と実験

5.1　プラットフォーム

Fig.5.1 に実験に使用したロボットの写真を、Fig.5.2 にそのシステム構成を示す。

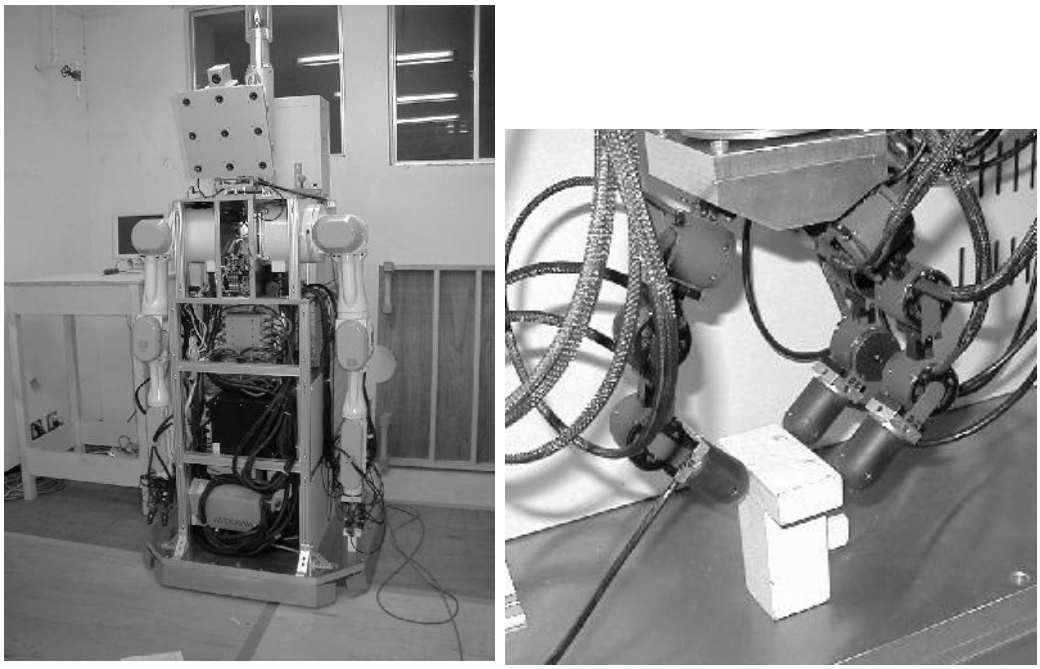


Fig. 5.1: 実験に使用したロボット

ロボットは物体を操作するために、7 自由度の二本の腕を持ち、左腕、右腕それぞれに 3 本と 4 本の指を持つ。指は 3 自由度であり、それぞれ指先に力センサーがついている。これによって、物体を把持するための制御や、操作物体と環境物体の接触の検出などが可能になる。

物体を認識するために、9 眼のステレオシステムを持つ。また、ズームカメラをひとつ持つ。今回の実装における物体の認識にはこのズームカメラを 1 台使って、テンプレートマッチングを行った。テンプレートマッチングは 2DTM というソフトウェアを使った。これは得られた画像のエッジ情報とあらかじめ与えられてある CAD モデルとのマッチング

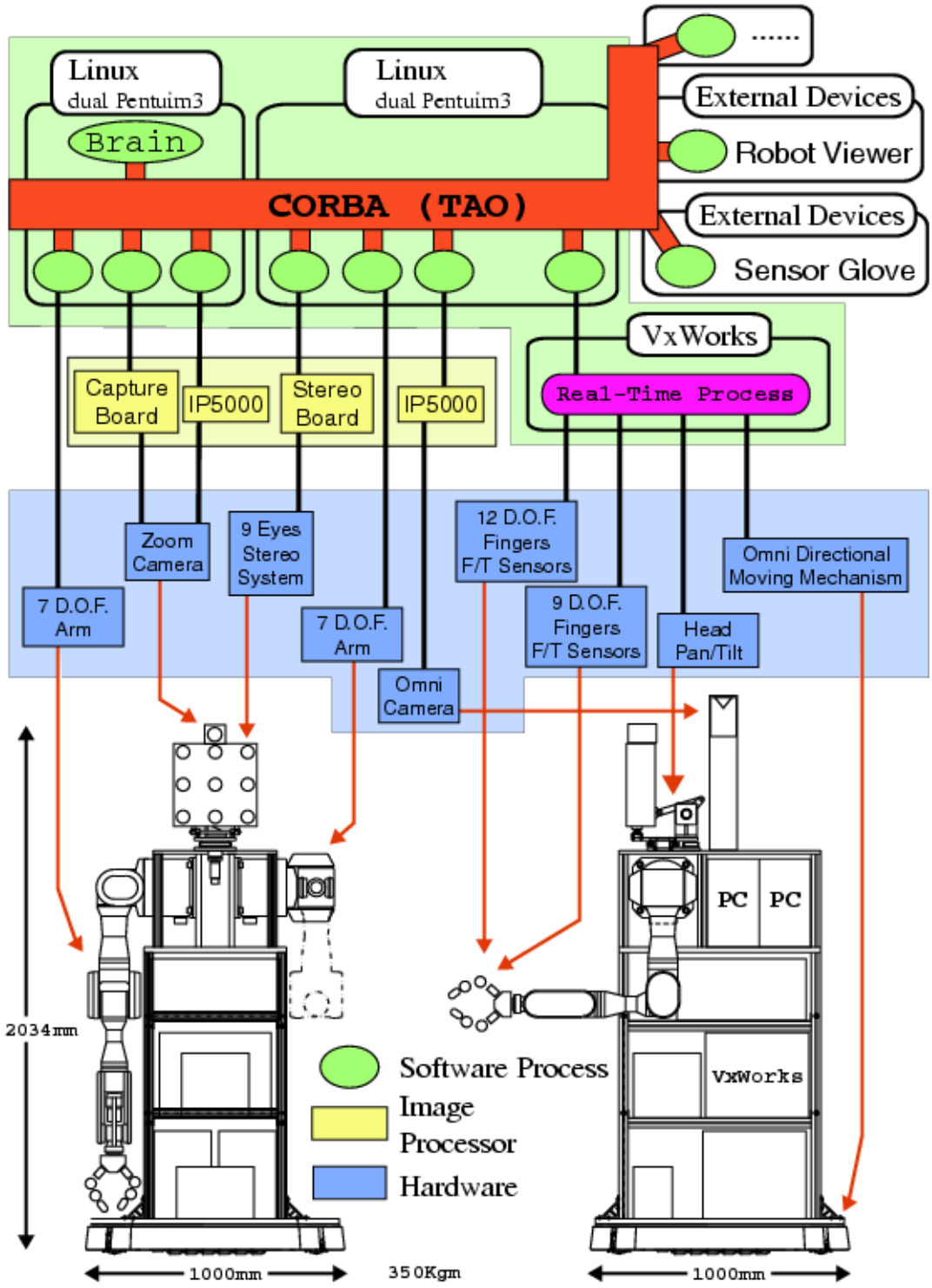


Fig. 5.2: 実験に使用したロボットの構成

をとるソフトウェアである。キャリブレーションを行うことで、ある程度まで奥行き情報も得ることができるが、一つのカメラから得られた情報であるから、信頼性は低い。そのため、今回の実験では 2 次元平面内での作業を仮定した。

その他、移動機構としてロボット全体が平面内の 4 方向の並進運動と、回転運動が行える機構を持っている。これによって、腕を動かすだけでは届かない範囲にあるものに対しても操作が行えるようになる。

これらはそれぞれの部位を動かすサーバに CORBA を使って通信を行うことによって、統一的に制御することが可能である。

5.2 実装

本節では 4 章で述べた、並進突き当て動作、回転突き当て動作、並進滑らし動作、並進滑らし動作を実際にどのように実装するかについて述べる。理論的には 4 種類のサブスキルで作業は行えるが、作業を確実に実行するために、適宜、補助的な動作を行う。

また、サブスキルを実現するための要素としては、

- ロボットの腕の位置制御
- 力センサーによるセンシング
- 単眼カメラと 2DTM によるセンシング

を利用する。

5.2.1 並進突き当て動作

並進突き当て動作は、現在の接触を満たしながら並進させて、新たな接触が起ったら停止する動作であった。

本研究で使用するロボットでは力センサーを使うことができるので、接触の判定には力センサーを用いる。

まず、並進突き当て動作を呼び出すのは、並進運動の maintaining DOF が detaching DOF に変化するときであるので、突き当て方向を最終状態の detaching の方向から得る。

つぎに、得られた突き当て方向へ物体を移動させながら、力センサーの値を監視する。センサーの変化がある閾値を越えたら、その位置で接触したと見なし、動作を停止する。この様子を Fig.5.3 に示す。

閾値を設ける必要があるのは、力センサーの値は、扱う物体の自重と動作の揺れによる力のかかり方の変化や、摩擦力の大きさ、ノイズなどによって常に変動しているためである。この閾値は実験によって最適と思われる閾値を設定した。

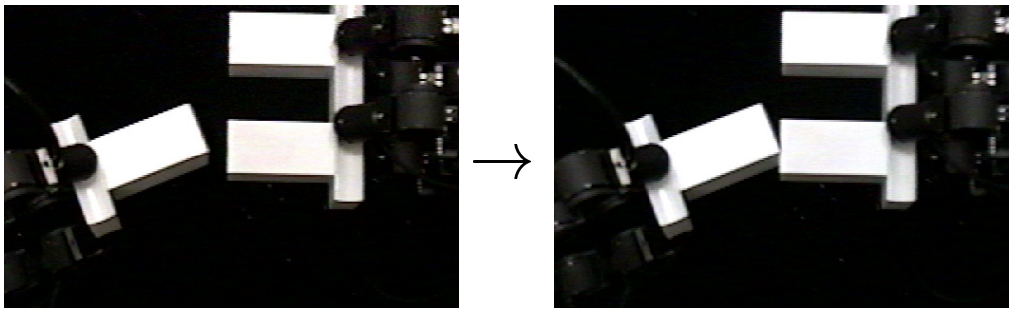


Fig. 5.3: 突き当て動作の実装

接触が全くない場合は障害物を回避しなければならない。これは人間の動作を模倣することによって実現する。これは後に補助動作として述べる。

基本的には、接触がない場合は模倣によって接触が達成されるが、実際には教示動作の模倣後、誤差によって接触が達成されない場合がある。この場合は、並進突き当てによって接触状態を達成させる。この場合、模倣によって回転運動の変化は達成されているので、回転運動に関する変化を考える必要はなく、また接触後の maintaining DOF の方向に関する誤差は、後の動作に大きな影響を与える可能性が低い。しかも、このような誤差が大きく影響する、singular contact を達成しなければならない場合には、適宜補正動作を呼び出すことができる。そのため、単純に突き当て動作を行うことで作業を行ってよい。

接触をもった状態からの突き当て動作の場合には、Fig.5.4 のように、最終状態の detaching DOF が 2 以上の場合がある。

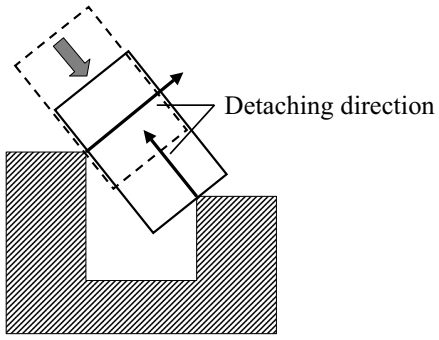


Fig. 5.4: 接触のある突き当て動作

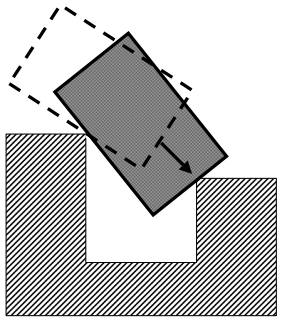


Fig. 5.5: 姿勢の変化する突き当て動作

この場合、突き当て方向を選択する必要があるが、これには動作のはじめの位置から終了位置へのベクトルを作り、各 detaching の方向との内積が最も大きいものを突き当て方向として採用する。

並進突き当てには、Fig.5.5 のようなものも起こり得る。これは並進突き当て動作の前後で姿勢が変化するものである。

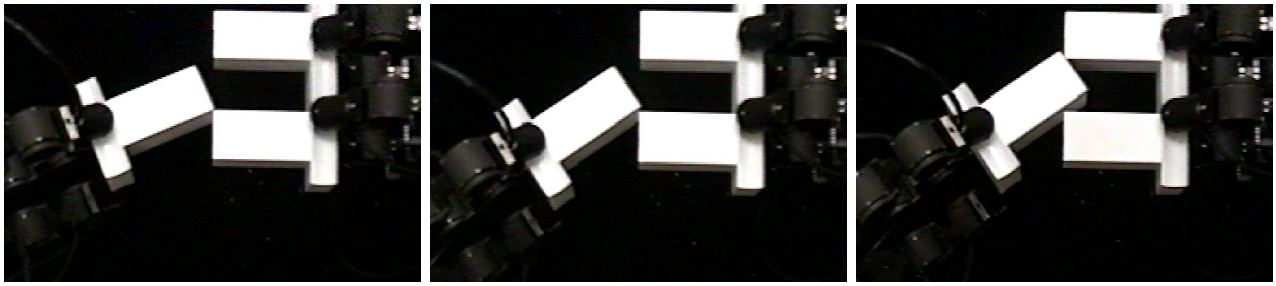


Fig. 5.6: 姿勢の変化する突き当て動作の実装

本来この接触によってもたらされる変化は、回転運動に関しての動作を呼び出すような変化ではない。しかしながら、姿勢の変化が大きい場合に、姿勢の修正なしに動作を行うと、衝突が起こって物体の移動が妨げられたり、目的とする接触が得られない可能性がある。そのため突き当て動作に入る前に、操作物体を突き当ての最終状態の姿勢に修正してから動作を行う。

最後に接触を維持すべき方向が、constraining DOF にあたる場合を考える。この場合、接触の維持は自動的に達成されるので、最終状態だけを検出すればよい。

5.2.2 回転突き当て動作

この動作はある接触を保ちながら、回転中心の周りを回転させ、環境物体へ突き当てる動作であった。

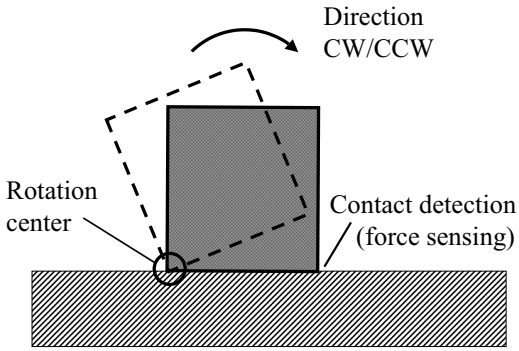


Fig. 5.7: 回転突き当て動作

回転中心を求めるためには、オブジェクトの構造に加えて、解析の際にどの頂点が接触しているのかを知る必要がある。システムは CAD モデルを使って軌道の解析を行っているので、この解析結果から接触点を知ることができる。

次に回転方向を知らなければならないが、人間の動作から、最終状態の物体の姿勢は得られている。そこで、現在の姿勢と最終状態の姿勢の差分を考えることによって、どちらに回転すべきかを求める。

最後に、力の変化を測定しながら、求めた回転中心と回転方向に従って回転運動を行い、力の変化がある閾値を越えた場合に動作を停止させる。(Fig.5.8)

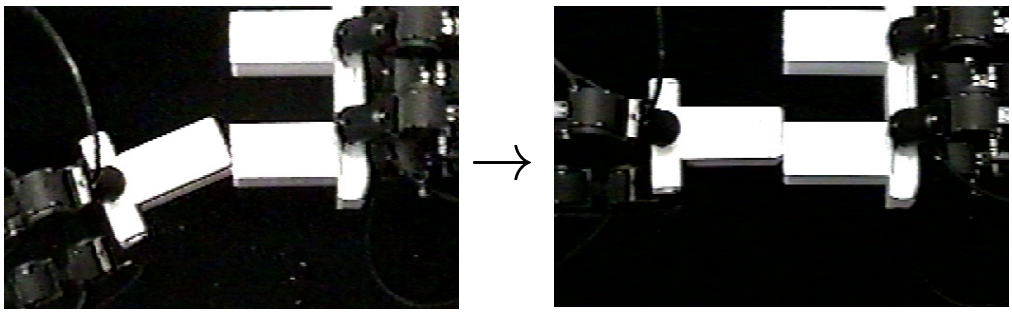


Fig. 5.8: 回転突き当て動作の実装

ここまでの実装によって、並進運動と回転運動に関する突き当て動作が実装できる。基本的には、並進運動に関する自由度に変化が起った場合には並進突き当て動作を、回転運動に関する自由度に変化が起った場合は回転突き当て動作を呼び出せば良いが、現実の作業では並進、回転の双方が同時に突き当てを起こすような変化があり得る。すなわち、Fig.5.9 のような場合がそうである。

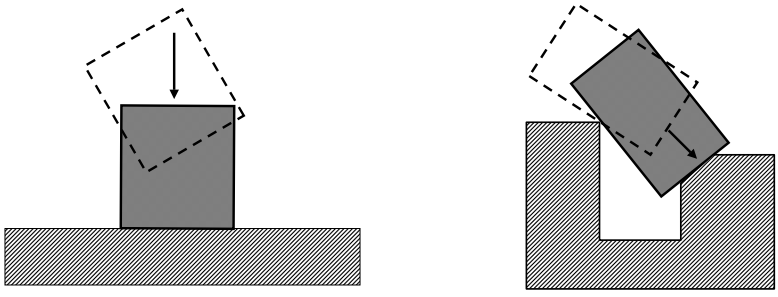


Fig. 5.9: 並進と回転が同時に起る場合

Fig.5.9 左の場合は接触の無い場合であるから、人間動作の模倣によって自動的に達成される。また、Fig.??右の場合は並進滑らし動作のうち、姿勢が変化する並進突き当て運動

に当てはまるものであるのでる。従って、並進、回転の双方で突き当て動作を要求するような接触の変化が起った場合には、並進突き当て運動を優先して呼び出すことで結果として同時に変化が起ったのと同等な運動を行うことができる。

5.2.3 並進滑らし動作

並進滑らし動作は、現在ある接触は保ちながら、ある方向に物体を滑らせる動作のことをさした。

まず、接触が失われた位置が滑らしの最終状態になる場合を考える。このとき操作物体は singular maintaining DOF を持つことになる。接触が失われた位置を見つけるには、力センサーの値の変化を利用する。

操作物体は、滑らし運動に入る前の突き当て動作によって、ある程度の力で環境物体に押し付けられている。この状態で滑らし方向に操作物体を移動させ、接触がなくなったときの力センサーでの変化を読み取って動作を停止させる。

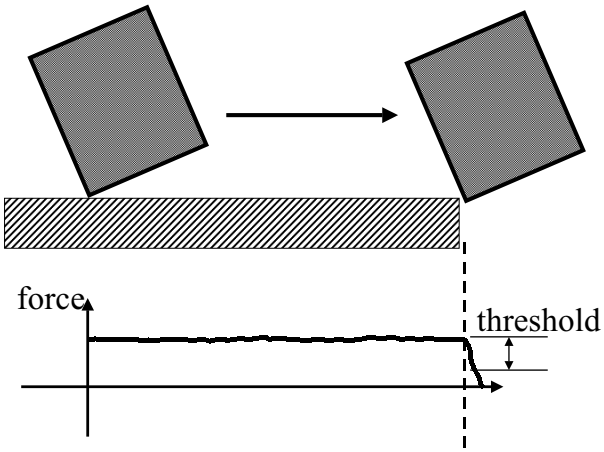


Fig. 5.10: 並進突き当て動作

この動作を行うためには、突き当ての判定および接触が失われたことを判定するための閾値をどの程度に設定するかが重要になる。大きすぎる力で押し付けると、把持が崩れて作業の続行ができなくなり、また、押し付けが弱すぎても、接触がなくなったときの変化がノイズに埋もれてしまい、動作の終了点を検出することができなくなってしまう。一般的に、押し付けた力以上の力の変化が、動作終了点で起こる可能性は低いので、閾値は押し付けの力の変化幅以下に設定すると良い。最適な値は、扱う物体によって異なるので、実験によって求めなくてはならない。

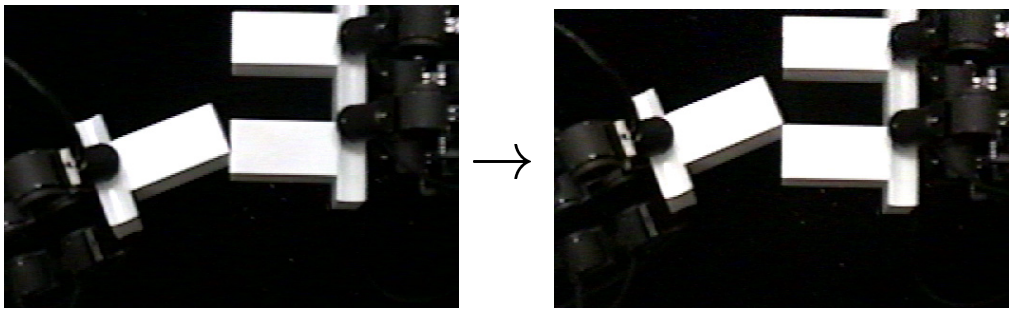


Fig. 5.11: 並進滑らし運動の実装

Singular maintaining DOF が maintaining DOF に変化する場合にも、並進滑らし動作が呼び出されるが、この場合は微少な距離を移動するだけで次の状態に遷移する。そのため、singular contact からの遷移に対しては、実際の移動は行わず、必要ならば補正動作のみを行うことにする。

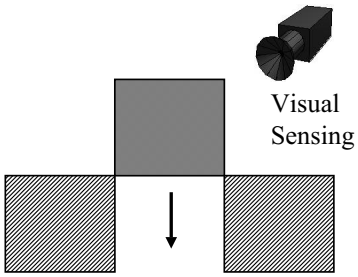


Fig. 5.12: 補助動作のみの呼び出し

例えば、Fig.5.12 の場合は、singular maintainind DOF から maintaining DOF への変化は矢印方向に微少距離動かせば達成されるので、実際に動作を行う必要はないが、横方向に関しては singular maintaining DOF が constraining DOF に変化するので、位置を正確に測定する必要があり、ビジュアルセンシングが行われる。

5.2.4 回転滑らし動作

回転滑らし動作は多点接触を保ちながら、回転を行う動作であった。これは回転中心がずれながらの回転運動なので、実装が難しい。Fig.5.13 のような挿入動作前のはめ合い動作に典型的に見られる動作であるので、この動作を例にとり検討を行う。

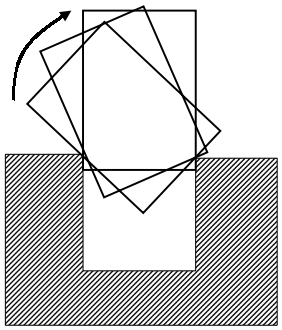


Fig. 5.13: はめ合い動作

まず、操作物体を環境物体に押し付けるようにしながら、回転させるという方法が考えられる。これを実現するには衝突を検知して操作物体を環境物体から離すような制御が必要となるが、本実験で使ったロボットでは処理の時間的遅れがあり、そのようなコントロールをすることが難しい。また、そのようなロボットを使用したとしても、衝突および摩擦による把持の崩れは避けられないと思われる。そのため、別の方法を検討することにした。

まず、回転滑らし動作は、滑らしによって接触を維持して動く動作と、回転運動が同時に起っている動作であるから、これを並進滑らしと回転運動に分けて考え、二つの動作の組合わせで実現することを考えた。すなわち、操作物体をある微少距離だけ一方の detaching DOF の方向に戻し、回転運動を突き当てが起るまで実行して、再び 2 点接触を達成する。そして、また微少距離だけ戻して、作業を繰り返す。

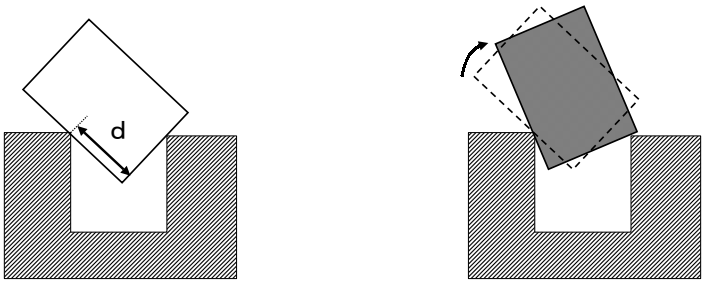


Fig. 5.14: 回転滑らし動作の実装方法

この動作を行うためには、回転中心と回転方向を知らなくてはならない。これには、回転突き当てと同様に、モデルからの解析によって接触点を得て、最終状態との差分から，回転方向を得る。

しかし、この場合は接触している頂点と辺が特定できるだけでは不十分で、辺のどの位置が回転中心になっているかを知らなくてはならない。また、操作物体の総移動距離はFig.5.14 左の d に等しくなるが、総移動距離がこれより大きくなると、物体が目的とする角度より大きく傾いてしまうことになる。

これら二つの理由から d を求める必要があるが、これには 2 通りの方法が考えられる。

まず、2 点接触が起こるためには、その前に突き当て動作が起こっているはずであるが、この動作が singular contact から始まっていたときには、どれだけの距離を移動させて接触が起こったかということを記録しておくことができる。

また、singular contact からの突き当て動作でない場合には、前の方法よりも誤差が大きくなる可能性があるが、ビジョンを使って測定を行う。

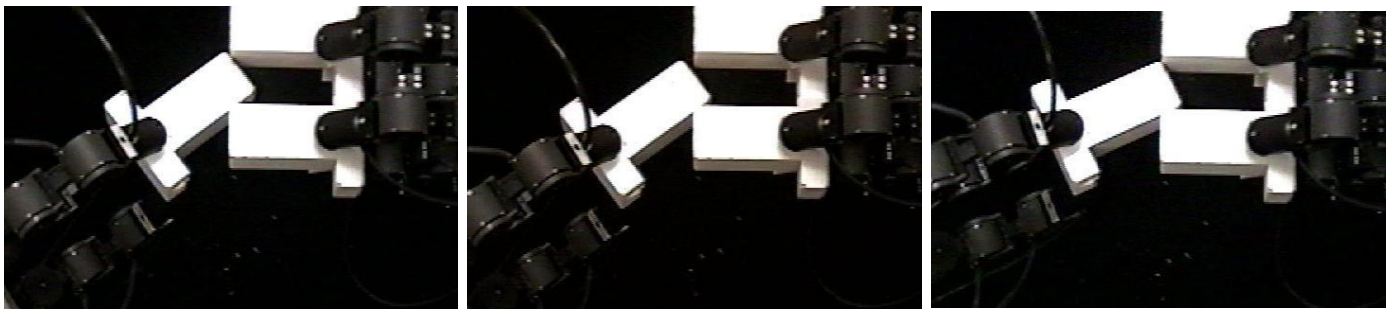


Fig. 5.15: 回転滑らし動作の実装

5.2.5 補助的動作

人間動作を模倣

接触のない状態で物体を動かす場合、人間動作を模倣させる。この状態は必ず接触をもって終了するので、模倣させながら力の変化を監視し、接触を持ったら終了する。最後まで接触を持たなかった場合は、誤差修正として並進突き当て動作を行う。この場合、姿勢の変化に関しては模倣動作終了時点で達成されているので、単純な並進突き当て動作を行えば良い。

この動作で、意図した面と異なる面で接触した場合は別の誤差修正アルゴリズムが必要であるが、そこまで大きな誤りは起きないものと仮定する。

ビジュアルフィードバックによる誤差修正

singular maintaining DOF が constraining DOF に変化するような、挿入動作の singular contact を達成するための手段として、ビジュアルフィードバックによる誤差修正がある。

実装に使ったロボットは、ズームカメラを備えているので、このカメラと 2dtm というテンプレートマッチングを行うソフトウェアを使って、位置を測定する。

ロボットはあらかじめ、カメラ座標とロボット座標のキャリブレーションを行っておき、扱う物体の CAD モデルも与えられている。この条件で、作業中に扱う物体の画像を取得し、そのエッジ情報を使って、CAD モデルとのマッチングを行う。

5.3 実験

3 章や 4 章の最後に例に挙げた、peg を hole に挿入する動作の画像をシステムに与え、C-space を使った解析を経て修正された軌道に対して、接触状態を解析して状態遷移図を得た後、サブスキルを呼び出して動作を実行した。

まず、オペレータはステレオカメラの前で目的とする作業を実行する。実行の様子を Fig.5.16 に示す。

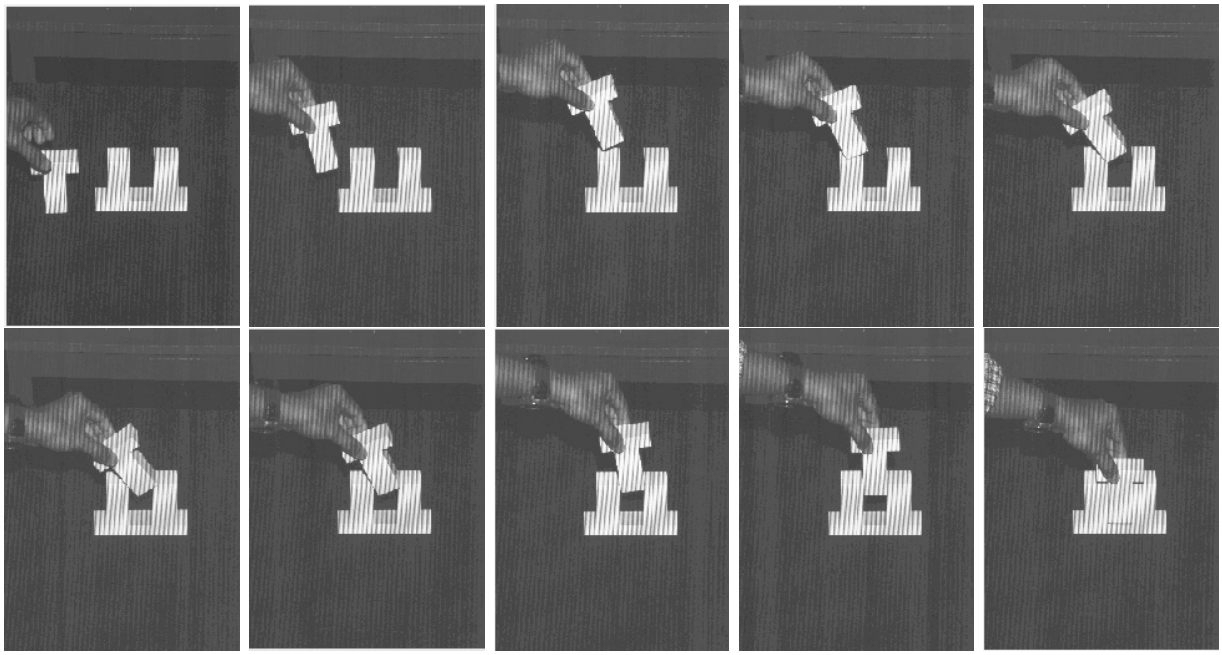


Fig. 5.16: オペレータによる動作の実行

この動作をステレオカメラを使って撮影し、距離画像を得る。この距離画像から得られた軌跡の誤差を C-space を使って修正する。ここまでは [4] の手法と同様である。

得られた実行可能な軌跡を、接触状態を元に分割し、状態遷移図を得る。この状態遷移図を Fig.5.17 に示す。カメラが単眼であることや [4] の手法が平面内の作業を仮定していることもあり、今回の実験では平面内の作業を仮定した。

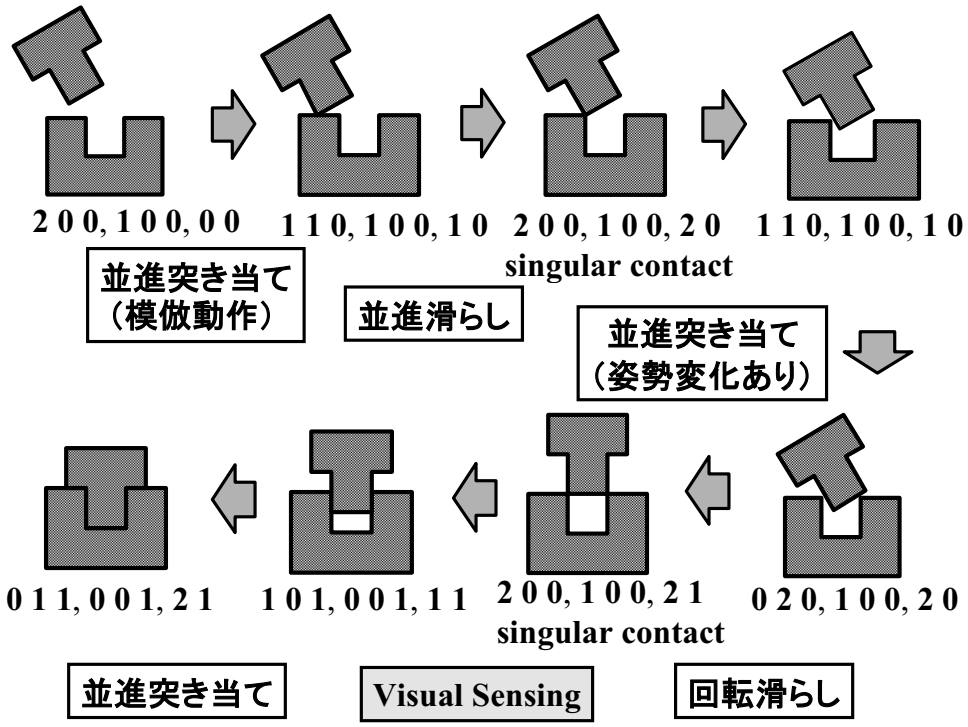


Fig. 5.17: 実験で得られた状態遷移図

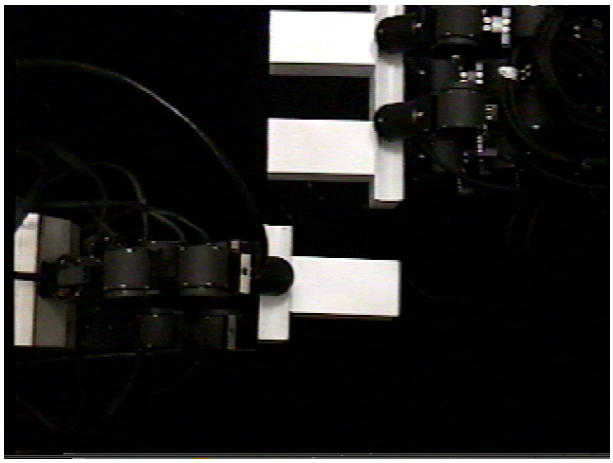
このような状態遷移図は実際には以下に示すようなデータファイルとして保持する。状態を表す各自由度の数、操作物体の環境物体に対する相対位置、接触法線ベクトルの向きなどの情報を保持している。

状態遷移データファイルの一部

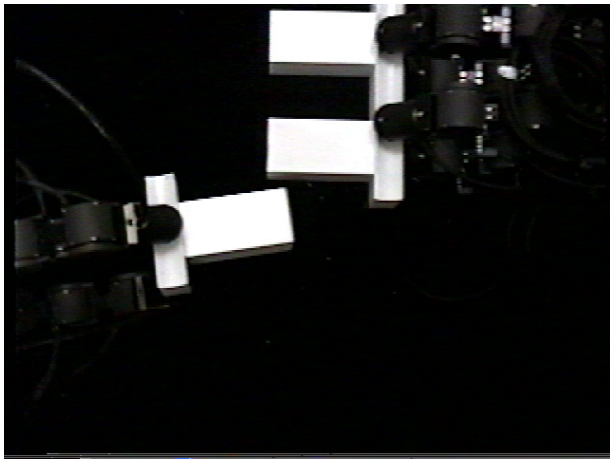
```
****** State 2 ******
Contact_class: 2 1 0 2 1 0 1 1
Configuration_number: 5
Sequence: 1
Position: 41.189999 74.080002 0.000000 -23.749998 -0.000000 -0.000000
Translate:
Detaching->
0.000000 1.000000 0.000000
Rotate:
Detaching->
1.000000 -0.000000 0.000000
```

```
Sequence: 2
Position: 41.189999 74.080002 0.000000 -23.749998 -0.000000 -0.000000
Translate:
Detaching->
0.000000 1.000000 0.000000
Rotate:
Detaching->
1.000000 -0.000000 0.000000
Sequence: 3
Position: 34.500000 74.070000 0.000000 -23.330000 -0.000000 -0.000000
Translate:
Detaching->
0.000000 1.000000 0.000000
Rotate:
Detaching->
1.000000 -0.000000 0.000000
```

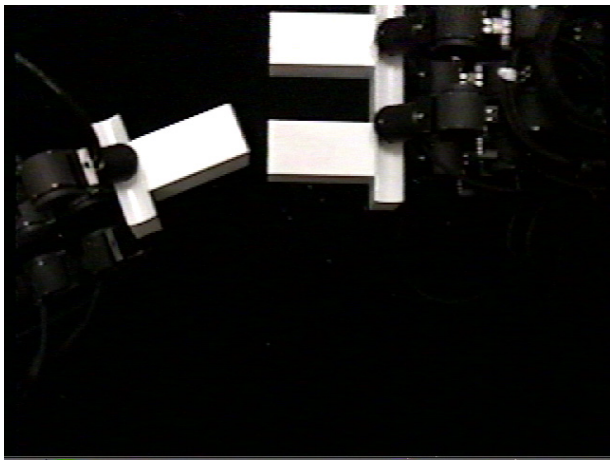
このサブスキルの呼び出し計画と、人間動作の軌道情報を使って、実装されたサブスキルを呼び出し、動作を実行する。以下に、獲得された peg-in-hole タスクのスキルが実行される様子を、一つ一つのサブスキルを説明しながら示す。



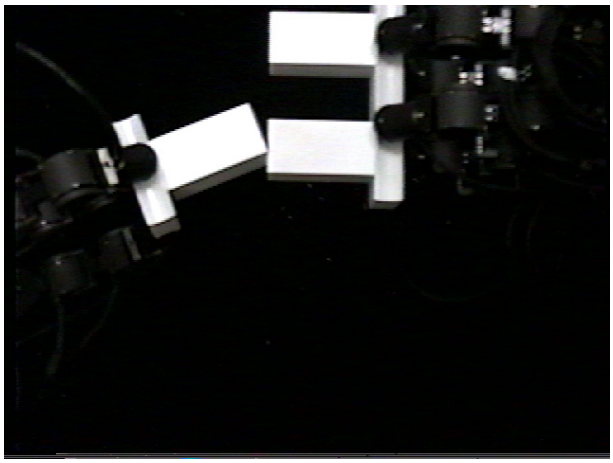
Sequence 1:
ロボットが環境物体および操作物体を、両腕を使ってそれぞれ把持し、各物体を観察によって得られた二物体の相対位置を実現するように移動させる。



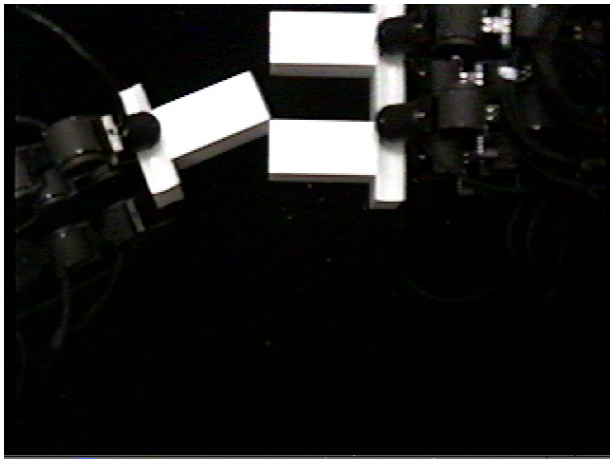
Sequence 2:
接触を持たない状態からの並進突き当て動作が呼び出されるので、人間動作を模倣して動作を開始する。



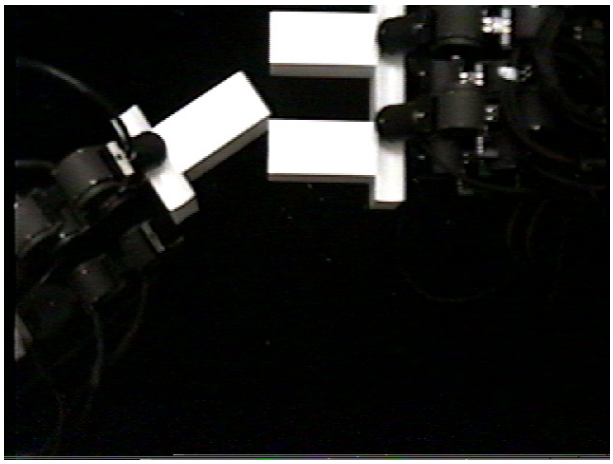
Sequence 3:
人間動作によって接触が達成されない場合は、この後並進突き当て動作を行う。接触が達成されたか否かは、力の変化を監視することによって行っている。



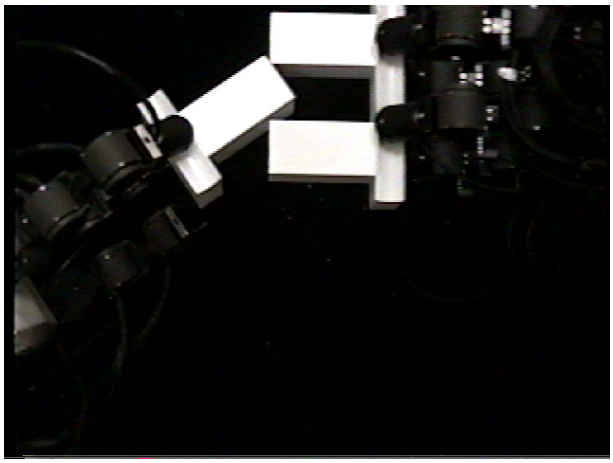
Sequence 4:
模倣動作を終了しても力センサーが接触を検知しなかったために、並進突き当て動作を行う。



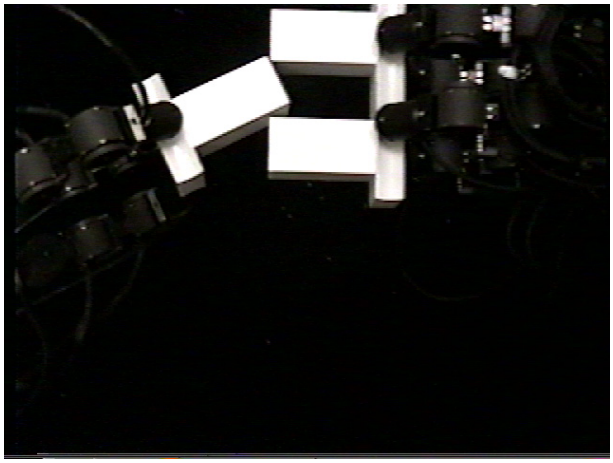
Sequence 5:
並進滑らし動作が呼び出され、singular contactが達成される。



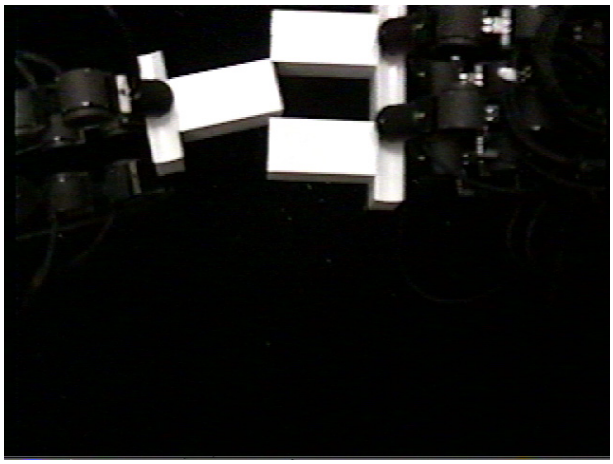
Sequence 6:
姿勢が変化する並進突き当て動作にはいる。まず、singular contactを保つようにして姿勢を変化させる。



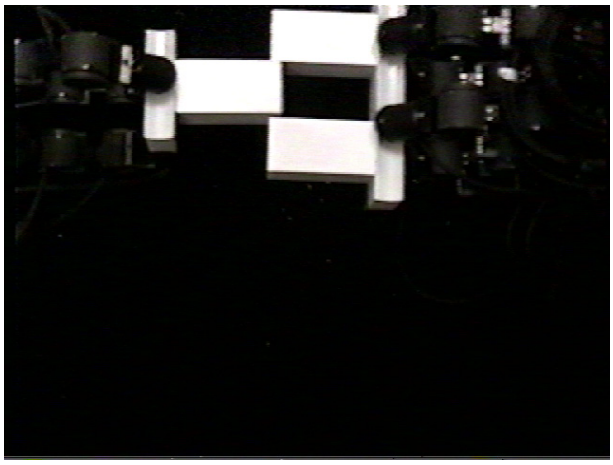
Sequence 7:
並進突き当て動作。あらかじめ姿勢を変化させているので、終了状態は目的とする位置・姿勢を達成している。



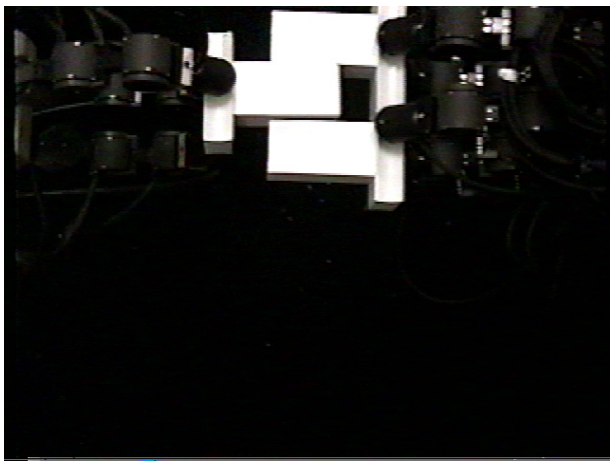
Sequence 8:
回転滑らし動作を行う。singular contact から二点接触を達成したので、カメラで挿入距離を測定する必要はない。画面下側の接触を維持するように、操作物体を滑らせる。



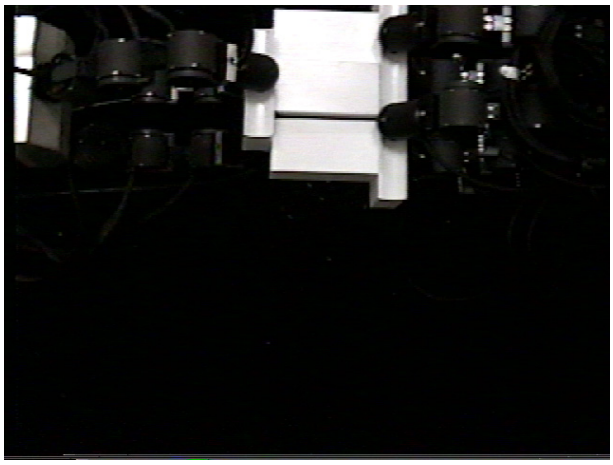
Sequence 9:
画面下側の接触を維持したまま、2 点接触を達成するように操作物体を回転させる。接触を検出したら、滑らし動作に戻り、目的とする姿勢が得られるまで繰り返す。



Sequence 10:
実験に使ったオブジェクトでは公差が約２ mm（左右各１ mm）あるため、回転滑らしによって操作物体は環境物体の間に挿入されるが、解析的には singular contact であり、singular maintaining DOF → maintaining DOF への滑らし動作が singular maintaining DOF → constraining DOF の変化を伴うものであるため、カメラによる位置の修正を行う。



Sequence 11:
並進突き当て動作を行う。



Sequence 12:
力の変化を検出し、作業を終了する。

以上のようにして、作業を達成することができた。

第6章　結論

6.1　本研究の成果

本研究では、操作物体と環境物体の接触状態の変化に着目し、その状態遷移に基本的な動作要素を割り当てることで、ロボットの動作スキルを獲得する手法を提案した。

本研究の主な成果を以下に挙げる。

まず、単純に人間の動作を模倣させる手法に、接触状態の遷移という要素を加えることによって、人間動作から得られた動作軌道を実行時に動的に修正することを可能にした。従来の人間動作による操作物体の軌道を再現する方法では、本研究で使用したロボットのように、把持状態を完全に把握することが難しい機構のロボットでは、完全な作業を行うことができなかった。また、作業の目的状態がわかっていないために、エラーを検出してもエラーリカバリを行う手法を適用することができなかった。本研究では、ロボットの動作一つ一つが目的とする状態を明確にしたことで、動作を確実に実行していくことを可能にした。

次に、作業の中間状態を記述するために、並進運動、回転運動に関して、接触によってもたらされる拘束状態を特徴量とした記述方法を提案した。さらに、それらを解析的に計算できるよう、接触状態方程式にスクリュー理論を使い、全ての拘束を行列で表現することで、行列のランクと face の次元を利用して、状態を分類する方法を提案した。

３番目に、状態遷移を実現するために必要な動作要素であるサブスキルを定義した。サブスキルは並進突き当て動作、並進滑らし動作、回転突き当て動作、回転滑らし動作を定義し、それらが状態遷移図から呼び出せることを示した。

４番目に、サブスキルを実際のロボットに実装した。実装に際しては、カメラによる visual sensing と力センサーによる力の変化の測定を利用し、これらのセンサーの使い方について述べた。また、サブスキルに加えて、作業の実行を容易にする補助動作を実装した。

最後に、実装されたサブスキルを用いて、実際に作業を行い、作業が効果的に実行できることを示した。

6.2　今後の課題

本研究の手法は、作業の内容や実装するロボットの種類に関わらず、一般的に適用できる手法である。しかし、扱う物体によって摩擦係数が違うために、接触や接触が失われた

ことを検出するための閾値をどの程度にすれば良いかは実験から求めなくてはならない。

人間が動作を行う際には、ロボットよりはるかに早いビジュアルフィードバックによって、物体の動きを観察し、それに合わせて力を加減しているものと思われる。また、経験による学習の効果も大きく影響していると思われる。

そのため、今後の課題として、作業が成功したか否かを何らかの評価関数を設けて判定し、力変化の閾値を最適なものに設定していく機構を構築することが求められる。

次に、把持の問題が考えられる。本研究は操作物体の動きを解析し、そこからサブスキルを設計し、動作スキルを獲得することを主眼に置いているため、物体をどのように把持するかに関しては、あらかじめロボットに把持手法を与えた。そのため、実行時に把持が崩れても、操作物体の運動を制御することでエラーをリカバリする方法しか取れない。

そこで、人間がどのように物体を把持して作業を行ったかを観察から求め、また作業中の効果的な把持手法を獲得できるようにしなければならないだろう。把持手法の分類には[15][16][17]などの研究がなされているが、腕の動きと連動した効果的な指の使い方を得る手法が今後必要であろう。

最後に、実行時間の問題が挙げられる。本研究ではリアルタイム性は考慮せず、作業が確実に実行されることに重きを置いた。積み木などの剛体を扱った組立作業ではリアルタイム性はさして重要にはならないが、溶接、接着などの要素が加わる、さらに複雑な人間の「技」をロボットで再現するためには、リアルタイム性を考慮する必要がある。そのためには、物体の動きに加えて、速度を考慮したサブスキルの設計や、物体の位置測定の高速化が求められるであろう。

謝辞

本研究を進めるにあたって、適切な御指導をいただいた池内克史教授に深く御礼申し上げます。また、研究全般に渡って貴重な助言を下さった佐藤洋一先生、電気通信大学の木村浩先生、ITESM の Santiago E. Conant Pablos 氏、九州電力の河村憲太郎氏、小松製作所の田貫富和氏ならびに池内研究室の小川原光一氏、高松淳氏、西野恒氏、大野一氏、電気通信大学の堀内智之氏、斉藤知隆氏、佐藤啓宏氏、三枝旭氏、さらに、研究の面だけでなく生活の面においてもお世話になりました池内研究室、佐藤研究室、坂内研究室の皆さんに感謝いたします。

参考文献

[1] K.Ikeuchi and T.Suehiro, "Toward an Assembly Plan from Observation Part I : Task Recognition With Polyhedral Objects," *IEEE Trans. Robotics and Automation*, vol.10, no.3, pp.368-384, June 1994.

[2] T.Suehiro and K.Ikeuchi, "Towards an Assembly Plan from Observation Part II : Correction of Motion Parameters based on Face Contact Constraints," in *Proc.of IEEE Int. Conf. on Intelligent Robots and Syst.*, Raleigh,NC, pp.2095-2102, July 1992.

[3] G.V.Paul and K.Ikeuchi, "Modeling Planar Assembly Task: Representation and Recognition," *Proc. of IEEE Int. Conf. on Intelligent Robots and Syst.*, IROS'95, Pittsburgh,Pa,USA, pp.17-22, 1995.

[4] G.V.Paul and K.Ikeuchi, "Modeling Planar Assembly Paths from Observation," *Proc. of IEEE Int. Conf. on Intelligent Robots and Syst.*, IROS'96, Osaka,Japan, pp.520-525, 1996.

[5] 國吉, 井上, 稲葉, "人間が実演して見せる作業の実時間視覚認識とそのロボット教示への応用", 日本ロボット学会誌, vol.9, no.3, pp.295-303, 1991.

[6] 比留川, 松井, 高瀬, "多面体間の接触による拘束条件を幾何モデルから導出する一般的なアルゴリズム", 日本ロボット学会誌, vol.9, no.4, pp.415-426, 1991.

[7] T.Takahashi, T.Sakai, "Teaching Robot's Movement in Virtual Reality," *Proc. of IEEE Int. Conf. on Intelligent Robots and Syst.*, IROS'91,Osaka,Japan, pp.1583-1588, 1991.

[8] 松岡，長谷川，本田，桐木，"作業状態観測と評価に基づく多関節多指ハンド物体操作システム"，日本ロボット学会誌，vol.17, no.5, pp.92-99, 1999.

[9] Takamatsu, Kimura and Ikeuchi, "Classifying Contact States for Recognizing Human Assembly Tasks," *Int.conf. on Multisensor Fusion and Integration for Intelligent Systems(MFI'99)*, Taipei,Taiwan,R.O.C., Aug., pp.177-182, 1999.

[10] 斎藤, 木村, 池内, “観察による組み立て計画生成と力覚による組み立て状態制御”, 日本ロボット学会学術講演会予稿集, vol.16, no.1, pp.61-62, 1998.

[11] 末広, “高技能マニピュレーションシステムに関する研究”，電子技術総合研究所研究報告, 第 912 号, 1990.

[12] J.Miura and K.Ikeuchi, “Task-Oriented Generation of Visual Sensing Strategies in Assembly Tasks,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.20, no.2, pp.126-138, February 1998.

[13] M.S.Ohwovoriole, et al., “An Extention of Screw Theory,” *Trans. ASME, Journal of Mechanical Design*, vol.103, pp.725-735, October 1981.

[14] B.Roth, “Screws, Motors, and Wrenches That Cannot Be Bought in a Hardware Store,” *Int. Symp. Robotics Research*, vol.1, pp.679-693.

[15] S.B.Kang and K.Ikeuchi, “Toward Automatic Robot Instruction from Perception - Recongnizing a Grasp from Observation,” *IEEE Trans. Robotics and Automation*, vol.9, no.4, pp.432-443, August 1993.

[16] S.B.Kang and K.Ikeuchi, “Toward Automatic Robot Instruction from Perception - Temporal Segmentation of Task from Human Hand Motion,” *IEEE Trans. Robotics and Automation*, vol.11, no.5, pp.670-681, October 1995.

[17] S.B.Kang and K.Ikeuchi, “Toward Automatic Robot Instruction from Perception - Mapping Human Grasps to Manipulator Grasps,” *IEEE Trans. Robotics and Automation*, vol.13, no.1, pp.81-95, February 1997.

発表文献

[1] H.Tominaga and K.Ikeuchi, "Acquireing Manipulation Skills through Observation," *Int.conf. on Multisensor Fusion and Integration for Intelligent Systems(MFI'99)*, Taipei,Taiwan,R.O.C., Aug., pp.7-12, 1999.

[2] 冨長，高松，木村，池内，"観察によるロボット動作スキルの獲得"，第*17*回日本ロボット学会学術講演会予稿集，vol.3，pp.875-876, 1999.

[3] Tominaga, Takamatsu, Kimura and Ikeuchi, "Symbolic Representation of Trajectories for Skill Generation," Int. Conf. on Robotics and Automation(ICRA2000), San Francisco, Apr., 2000, accepted.