修士学位論文

ロボット作業のための面・エッジを用いる位置・姿勢の教示に関する研究

平成 12 年度
（2000）

東京商船大学大学院

商船学研究科

交通電子機械工学専攻

田尻　大地

目次

Abstract 6

第 1 章　序章 7

第 2 章　物体の位置・姿勢の決定法 9

2.1　面ベースによる位置姿勢の決定 9

2.2　位置・姿勢を決定する面の組み合わせ 10

2.2.1　平面のみを利用する場合 11
2.2.2　円筒面を利用する場合 11
2.2.3　円錐体を用いる場合 13
2.2.4　球を利用する場合 14
2.2.5　楕円体を利用する場合 15

2.3　実環境における望ましくない条件 15

2.3.1　オクルージョン 15
2.3.2　鏡面反射 15
2.3.3　三角測量にともなう不可計測領域 16

2.4　エッジ要素の利用 16

2.5　処理の手順 16

2.6　対象物と作業内容 19

第 3 章　3 次元データの取得と前処理 20

3.1 3次元データの取得 20

3.2 面素 22

3.3 基礎領域の生成 23

3.4 エッジ検出 25

第4章 タスク指向型教示ツリーの　構築と位置・姿勢の教示 27

4.1 形状データベース 27

4.1.1 SOLVERによる形状データの作成 27

4.1.2 DXFフォーマットによるデータ表現 31

4.2 アスペクトの導出 32

4.2.1 測地ドームによるアスペクト法 33

4.2.2 位置・姿勢を決定に利用する面 35

4.3 ツリーの構築 38

4.4 エッジ要素 40

4.5 エッジベースツリーの構築 40

4.6 教示の流れ 42

第5章 実験結果 45

5.1 理想的な環境における場合 45

5.2 エッジツリーを用いる場合 47

5.3　オクルージョンが起こっている場合 49

第 6 章　まとめ 51

謝辞 53

参考文献 54

付録 1 DXF フォーマットのファイル 59

付録 2 プログラム 61

図目次

1 一つの面のみが一致する場合 10

2 円筒面を一つのみ利用する場合 11

3 円筒面が複数含まれる場合 13

4 円錐体を含む場合 14

5 処理の手順 18

6 バルブとロボットタスク 19

7 3次元計測器 VIVID 700 20

8 VIVID 700における 3次元計測の基本原理 21

9 計測されたバルブの 3次元データ 21

10 3点からの平面の構成 22

11 面素分割による法線の算出 23

12 伝播法による 2値画像のラベリング 24

13 ラベリングによる基礎領域 25

14 検出されたエッジ要素 26

15 バルブの設計図 28

16 素立体の生成 (1) 29

17 素立体の生成 (2) 29

18 合成した（和集合）物体 30

19 (3)から(2)を引いたもの 30

20 バルブの形状データ 31

21 DXFファイルの全体構造 32

22 測地ドーム 34

23 導出されたアスペクト群 35

24 教示面の選択 36

25 アスペクトグループ 37

26　タスクに関係する面 38
27　面ベースによるタスク指向型教示ツリー 39
28　教示エッジ 40
29　エッジベースの教示ツリー 41
30　教示の流れ 44
31　理想的な環境における教示データ 45
32　面ベースによる処理結果 46
33　エッジツリーを用いる場合 47
34　エッジを用いる処理結果 48
35　オクルージョンが起こっている場合 49
36　オクルージョンにおける処理結果 50

表目次

1　自由度の可決定性 12

Abstract

本論文では，極限作業ロボットを対象とした作業プランニングにおいて，その作業環境を教示する方法を提案する．ロボットの作業プランニングにおいて，作業対象の形状，位置，姿勢を記述する幾何モデルが必要となる．通常工業製品は物体の形状はデータベースとして保存することが可能であり，与えられた環境より，位置・姿勢を決定すればよい，極限作業ロボットが作業する実環境は，特殊な状況であり，また高度作業が伴い，失敗が許されない．不確実性を伴う完全自動による物体の検出よりむしろオペレータの介入により確実かつ効率的なモデリングを行うことが重要である．そこで，本研究では，レンジファインダを用いて環境の距離データを取得し，オペレータとの対話的処理により，環境内に存在する物体の位置・姿勢を決定する．あらかじめ，対象物体の形状データはCADモデル等で与えられているものとする．物体の形状データから位置・姿勢を決定する面の組み合わせを考慮した教示ツリーを作成しておく．個々の状況に応じた，物体の位置・姿勢の教示は次のように行う．対象とする物体を含む環境の距離データを取得したうえ，オペレータが教示ツリーの一部を選択することにより，距離データ内にある対応する面・エッジ等の部分を指示する．これによりシステムが物体の位置・姿勢を決定する．本研究においては，面の組み合わせを構造化したツリーにエッジ要素を導入して，オクルージョン等の望ましくない状況にも対応ができるようにしている．さらに対象物体とその作業内容を考慮し，ツリーに優先順位を持たせることによって，作業に適したものとすることを可能にしている．本研究で提案した手法により，極限作業ロボットが作業する特殊な環境においても，従来よりも迅速，かつ効率的で正確な位置・姿勢の教示が可能となり，さらにタスクに適した教示ツリーにより，オペレータに対する負担の軽減が可能となる．

第1章　序章

近年，原子力発電施設関連作業ロボットや海底石油生産施設関連作業ロボット，石油生産施設関連防災ロボットなどの極限作業ロボットの研究が積極的に進められている．これらのロボットは放射線や高水圧，高温といった人間が立ち入ることのできない領域で，プラントの保守・点検・修理や災害の状況把握・拡大防止，さらには救援などの高度な作業を行うものである．一般にこの種の作業は失敗が許されず，このような実環境でロボットが自律的に作業を行うためには，センシング，プランニング，マニピュレーションを統合した作業計画機能が不可欠であり，それらを確実に行うシステムが要求される．それゆえ，不確実性を含んだ自動化よりも，人間の介入によりミスのない作業を実行することがもっとも大切なことである．むしろ重要なことはオペレータに負担をかけずに確実な作業を行うことである．

このような背景の中で，ロボットの作業プランニングにおいては作業環境内に存在する物体の形状や位置・姿勢を記述する幾何モデルが重要な役割をもつ．それゆえ，計算機の中に幾何モデルが正確に構築されていくことが望ましい．コンピュータビジョンの分野では，3 次元物体認識技術が急速に発達し，物体の位置・姿勢の決定を同時に自動的に行うシステムが多く開発されてきている [1]-[6]．しかしながら，これらの研究では，環境の 3 次元データがほぼ完全に得られており，かなり理想的な状況を対象としている．極限作業ロボットの作業環境というのは，これとは異なり，一般的に狭い範囲に多くの物体が配置され，オクルージョンが非常によく発生する．さらに，物体の鏡面反射や三角測量にともなう不可計測領域のため，得られるデータはさらに少なくなる．それゆえ，コンピュータビジョンにおいて利用されるそれらの手法を極限作業ロボットの幾何モデリングに直接利用することは，難しい場合が多い．極限作業ロボットの作業環境の幾何モデリングに関しては，正確さ，速さ，効率のためには，オペレータの手助けがいまだ必要なのである．

一般に，工業製品などの物体の形状はデータベースとして蓄えておくことが可能であり，その場合決定するべき要素は位置と姿勢ということになる．原子力発電施設や石油生産施設等，極限作業に関しても，その条件が当てはまる．また環境を計算機に構築する場合に作業環境すべてをモデル化するのでは広大な情報量を必要とし，効率のよい幾何モデルとはいえない．ロボットが行うべき作業内容（タスク）に関係した部分だけを必要な情報とし，モデル化することによって最小限の情報で環境幾何モデルとすることが重要であ

る．そのため環境の中から作業対象となる物体を抽出し，その位置・姿勢を決定することが重要な役割を持つ．原子力用ロボットのビジョンシステムとして，レーザポインタによる計測によって正確に位置と姿勢を与えるシステムが提案された[7]-[9]．しかしながら，このシステムは，最適な位置・姿勢の決定方法の選択，オペレータによるレーザポインタのスポッティング制御，物体のエッジや頂点上のスポッティング，単一視点だけからのモニタリングなど，いくつかの問題を持っていた．これらはオペレータに多大な負担をかけるものである．

これらの問題を解決するために，これまでに，レンジファインダを用いた面ベースによる対話的な位置姿勢の決定法が提案された[10]-[16]．このシステムでは，レンジファイダを使用し，大域かつ綿密な3次元データを獲得するため，オペレータがレーザポインタを制御する必要はない．また位置・姿勢の決定には，面の組み合わせを表した教示ツリーを用いている．物体の形状データより位置・姿勢を決定するために，必要かつ最小限の面の組み合わせを考慮しているため，不定自由度の残ることなく，正確に位置と姿勢を決定することが可能となっている．またオクルージョンや鏡面反射などによって対象物の計測した3次元データが，完全に得られない場合においても位置・姿勢を決定するための面の一部がデータとして得られていることによって，望ましくない状況下においても柔軟に対応できる．しかしながら，状況によっては，計測によって得られる3次元データに，位置・姿勢を決定する最小限の面の組み合わせが必ずしも存在するとは限らない．

そこで，本研究は，位置・姿勢の組み合わせに加え，物体を構成するエッジ要素を用いる線の3次元情報を教示ツリーに導入する手法を提案する．これまでのツリーに位置・姿勢を決定するためにデータの獲得状況により不十分となる面の組み合わせを考慮し，その場合に必要となる各面を構成するエッジ要素のツリーを構築する．これによってオクルージョン等による望ましくない状況や，センサと対象物の配置から位置・姿勢を決定する面の組み合わせが計測された3次元データに存在しない場合においても物体の位置・姿勢の決定が可能で，幾何モデルを生成することができる．また，ロボットに作業を行わせるという点での幾何モデルということを考えると，位置・姿勢が，その行うべき作業内容（タスク）に準じたものとして決定されることが，プランニングを効率かつスムーズに行うために重要な役割を持つことになる．そこで，本手法では，面やエッジの教示ツリーを生成する際に，ロボットが行うべき作業内容（タスク）を考慮し，タスクに関係のある面やエッジほどツリーに高い優先順位をつける．優先順位をつけることによって，オペレータが容易に組み合わせを選択できるとともに，その選択がタスクとのつながりを持っているため，作業教示する場合での信頼性という点でも，大変効果がある．

第2章　物体の位置・姿勢の決定法

2.1　面ベースによる位置姿勢の決定

本研究では，物体の形状データからそれを構成する面成分と，レンジファインダによって得られる3次元データから面データを導出して，その座標と法線ベクトルを用いて照合することにより対象物体の位置・姿勢を決定する事を基本とする．計測された3次元データからエッジを導出し，エッジベースで物体の位置・姿勢を決定することも考えられる．しかし，本研究においては，基本となる部分においてはエッジベースによる位置・姿勢の推定よりも，以下の理由で，面ベースによるそれを優先して考える．

- レンジファインダによって直接得られるデータは，対象物体の表面上のデータである．そのため，面データを用いると，エッジのみのデータを用いるより，データの有効利用率が格段に高くなる．

- エッジをベースとして位置・姿勢を決定する場合，観測されたレンジデータから，エッジ要素を導出することになる．そういった場合，実際にロボットが作業する環境においては，レンジデータのエッジには，物体を構成する実エッジのほかに，オクルージョンによって発生する見かけ上のエッジも含まれるため，それらを識別するのが困難で，効率的なモデリングができない．

以上のことにより，面ベースでの位置・姿勢の決定法を優先させる．次節において，物体の形状データと，レンジデータから導出された面データとを比較する場合，対象物体の位置・姿勢を決定するために必要な面の組み合わせについて説明する．

2.2　位置・姿勢を決定する面の組み合わせ

あらかじめ与えられている形状データの面と，レンジデータから導出した面データとを比較して，その座標と法線ベクトルを用いて物体の位置・姿勢を決定する場合，ひとつの面のみが一致するだけでは不十分である．このような場合について図を用いて例をあげる．

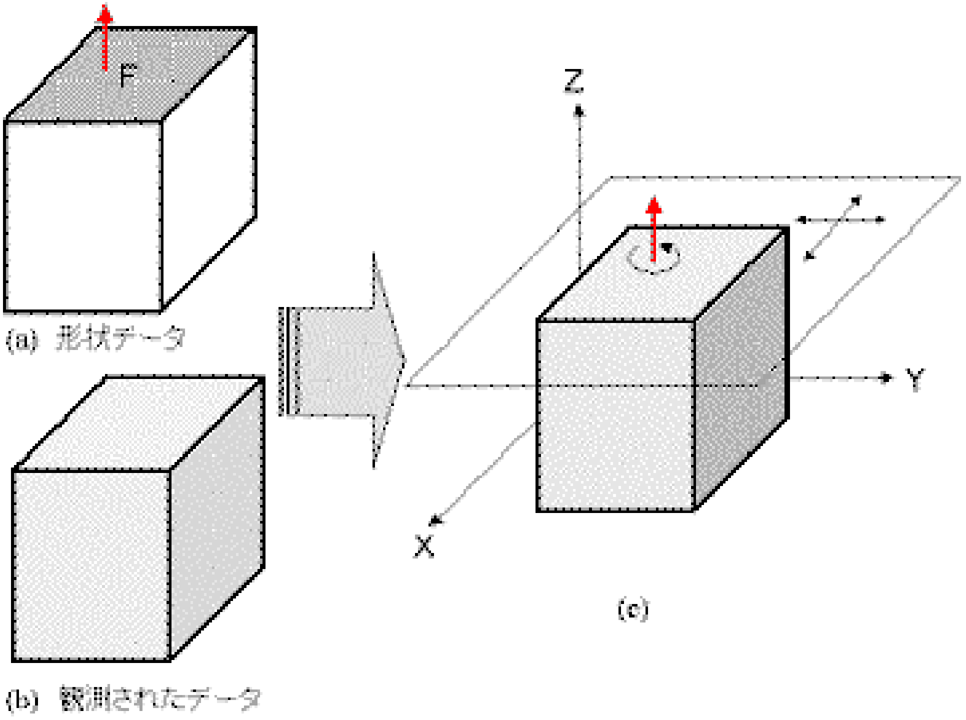


図1.　一つの面のみが一致する場合

上図 1．(b)のような観測されたデータがあるとして，既知である形状データ図 1．(a)の面 F をその位置と法線ベクトルによって，観測された面データと一致させるとする．すると図 1．(c)のようになり，直方体に 3 次元座標を対応させると，一つの面 F のみが一致した場合，Z 軸に平行な回転成分と，XY 平面に平行な並進成分の自由度が残ってしまう．面の一致によって物体の位置・姿勢を決定するためには，このような自由度が残らないようにしなければならない．つまり，複数の面を組み合わせることによって，形状データと観測データの間の不定自由度を無くすことにより，物体の位置・姿勢を決定するということである．本節では，形状データと観測データを一致させるときに，不定自由度が残らない，位置・姿勢を決定するために必要な最小限の面の組み合わせについて，説明する．

一般的に物体は，平面，円筒，楕円体，円錐，球の各面パッチにより構成されており，ほとんどの工業製品を，それらの面要素によって表現することができる．さらに，ここで

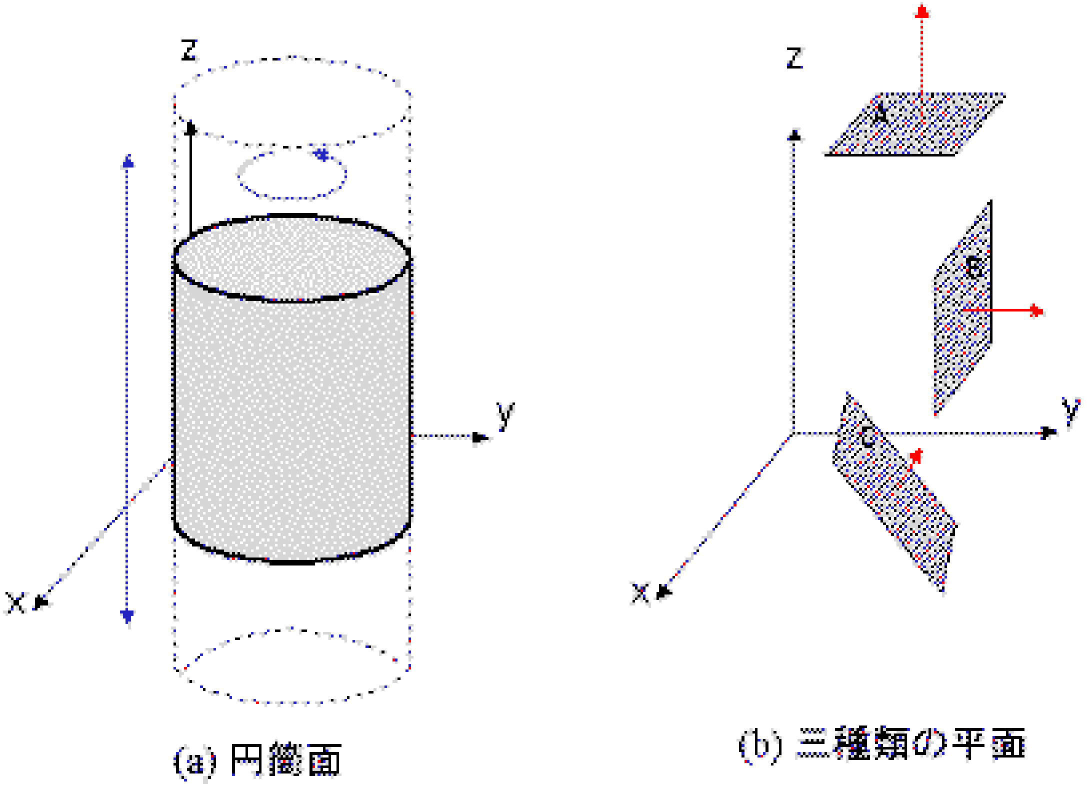
は，位置・姿勢を決定する面の組み合わせが必ず一つまたはそれ以上存在するものとする．また，簡単のために，円筒，楕円体，円錐，球の面要素の内，異なる二種類のものが同時に位置・姿勢の決定に利用される場合は除くものとする．

2.2.1 平面のみを利用する場合

平面のみを一致させる場合，法線が一次独立である三つの面が一致することにより，不定自由度がなくなり，物体の位置・姿勢を決定することができる．

2.2.2 円筒面を利用する場合

(1) 円筒面を一つのみ利用する場合



(a) 円筒面

(b) 三種類の平面

図2.　円筒面を一つのみ利用する場合

対象となる物体の面が円筒面と平面を含む場合を考える．形状データと観測データの間において円筒面が一致すると図2．(a)に示されるように円筒面は，円筒の中心を軸として Z 軸に対して平行な軸周りの回転成分と Z 軸に対して平行な並進成分の不定

自由度を持つ（もしも，物体が軸周りの回転に関して何の属性も持たなければ，不定自由度は Z 軸に平行な並進成分に関する 1 自由度のみとなる）．それらの自由度はその対象物体の持つ他の平面の計測された 3 次元データから決定されなければならない．そこで，物体が円筒面のほかに図 3．ｂに示されるような下記の属性を持つ平面を持つ場合について考える．

- 面 A：XY 平面に平行な面
- 面 B：Z 軸に平行な面
- 面 C：面 A，B 以外の面

上記不定自由度に関する可決定性は表 1．に示される．

表1.　自由度の可決定性

	並進成分	**回転成分**
面 A	**不定**	**確定**
面 B	**確定**	**不定**
面 C	**確定**	**確定**

円筒面のほかに，面 A の属性を持つ面が一致した場合， Z 軸に平行な軸周りの回転成分の自由度は確定するが， Z 軸に平行な並進成分は不定となる．逆に面 B の属性を持つ面が一致した場合，回転成分の自由度は確定するが，並進成分の自由度は確定できない．またそれ以外の面 C の属性を持つ面が一致すれば，どちらの自由度も確定できるため，物体の位置・姿勢が決定できる．つまり，一つの円筒面と平面を含む物体の位置・姿勢は，円筒面が一致するほかに面 C の属性を持つ面，または面 A かつ面 B の属性を持つ二つの面がそれぞれ一致することによって決定することができる．

(2) 複数の円筒面を利用する場合

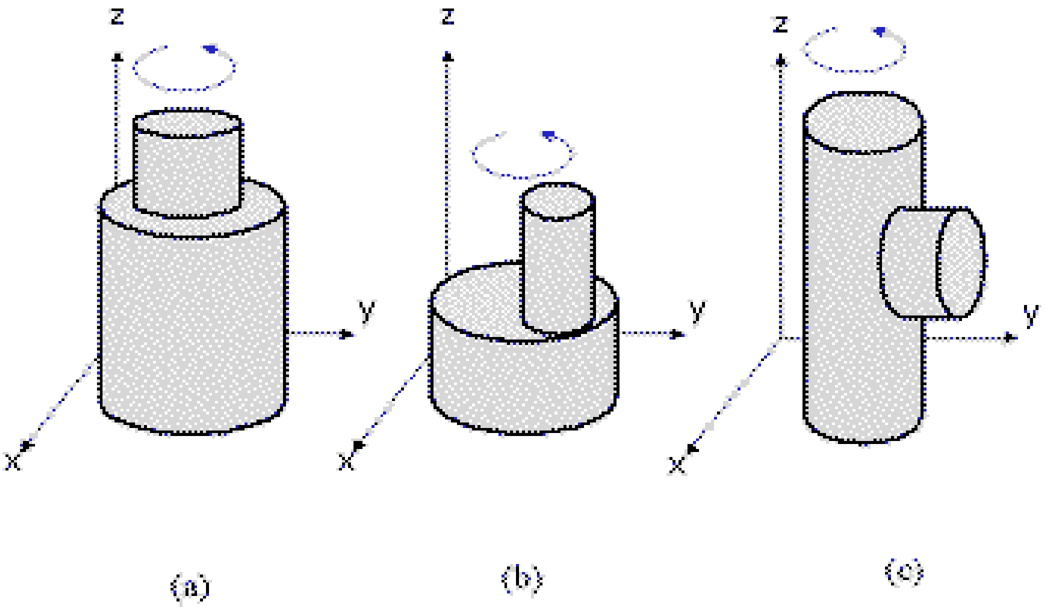


図3.　円筒面が複数含まれる場合

(i) すべての円筒の中心軸が一致している場合

複数の円筒のうち，任意の二つの円筒は図３．(a)のような関係にある．これは上述した円筒面を一つ利用する場合と同等になるため(1)にしたがって位置・姿勢の決定を行う．

(ii)　すべての円筒の中心軸が平行である場合

複数円筒のうち二つの円筒は図３．(b)のような関係にある．Z 軸に平行な軸周りの回転成分は，中心軸が一致しない二つの円筒面によって決まる．さらに，図 2．(b)の中の面 A，あるいは面 C の属性をもつ面が一致することによって，Z 軸に平行な並進成分の不定自由度をフィックスできる．

(iii)　円筒軸がすべて平行というわけではない場合

中心軸が平行でない二つの円筒は図 3．(b)のような関係にあり，形状データ，観測データ間でそれらの二つの円筒面が一致することによって，すべての自由度が決定できる．

2.2.3　円錐体を用いる場合

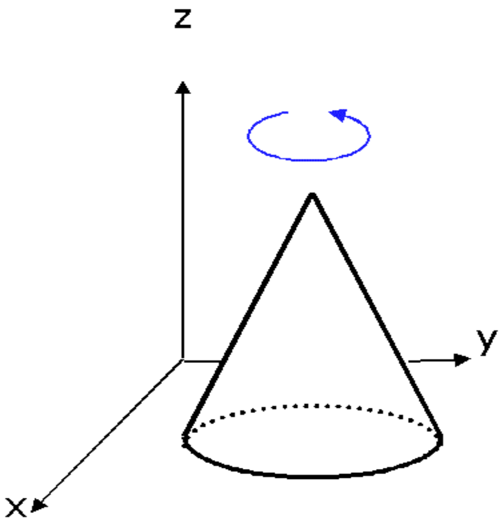


図4.　円錐体を含む場合

図 4．に示されるような円錐体は底面の中心から　Z 軸に平行な軸周りに不定自由度が存在する．この自由度は，そのほかに，　2．2．2 の(1)の一つの円筒面を利用する場合の図　2．(b)における面　B 属性を持つ面が一致する，あるいは面　C の属性を持つ面が一致することによってフィックスできる．また，複数の円錐体が存在する場合上記　2．2．2 の(2)のような円筒面を複数利用する場合と同様に面や円錐体軸を用いて一致させることによって決定できる．

2.2.4　球を利用する場合

(1)　球が一つ存在する場合

対象物が球面を含みその球面を利用して物体の位置・姿勢を決定する場合，回転に関する何らかの属性が存在すれば，物体の回転は，上述　2．2．2 の場合と同様に回転方向をフィックスできる面を一致させることによって決定できる．それ以外の場合，すべての自由度は球の面を一致させることによって決定できる．

(2)　複数の球が存在する場合

(i)　すべての円筒の中心軸が一致している場合

球で結ばれた物体の不定自由度を決定する場合は，一つの円錐体を利用する場合のそれと同等となる．それゆえ，上述　2．2．3 の同等になる．

(ii) (i)　以外の場合

上述した(i)以外の場合は，すべての自由度は，重心がひとつの直線上に含まれない3つの球面を一致させることによって，不定自由度を確定でき物体の位置・姿勢を決定することができる．

2.2.5 楕円体を利用する場合

一楕円体が球，あるいは回転楕円体であれば，それぞれの上述した２．２．３の円錐体を利用する場合，２．２．４の球を利用する場合と同等になる．それ以外の場合は，楕円体の面を一致させることによってすべての自由度を決定することができる．

2.3 実環境における望ましくない条件

前節２．２において，形状データと観測データ間で複数の面を一致させ物体の不定自由度をなくすことにより物体の位置・姿勢を決定すること，またその位置・姿勢を決定するための面の組み合わせについて説明した．実際にセンサを用いて環境のデータを取り込み，環境中の物体の位置・姿勢を決定する場合，その位置・姿勢を決定できる面の組み合わせは，物体のアスペクト(見え方)の違いによって変化する．その組み合わせは一つのアスペクトに対して複数存在することが多い．しかし，望ましくない状況下では，利用できる組み合わせは減少する．極限作業ロボットなど実環境における環境幾何モデリングは，作業環境の特殊性ゆえ無視することのできない状況がある．状況によっては，位置・姿勢を決定するための面が計測で得られる3次元データに反映されていない状況もないとは言えない．本節では，レンジファインダを用いて作業環境の3次元データを取得する際，そのような物体の位置・姿勢の決定に望ましくない条件についていくつか取り上げる．

2.3.1 オクルージョン

原子力発電所の内部は，パイプ，バルブ，ポンプ等が幾重にも複雑に配置されている．このように極限作業ロボットが作業を行う領域は，密接して多くの物体が置かれている．それゆえ，いくつかの物体は他の物体にしばしば隠蔽されてしまう．この場合，隠蔽された領域に含まれない可観測データのみで位置・姿勢の決定を実行しなければならない．

2.3.2 鏡面反射

多くの工業製品は，その素材やコーティングのために光を反射する．レンジファインダのようにレーザ光を照射する測定の場合，そのような面上の3次元データを正確に獲得することは困難である．それゆえ，鏡面反射を起こす面以外のデータを用いて位置・姿勢を決定しなければならないことになる．また，レンジファインダを用いる場合，反射後の光と間違えて認識することによる虚偽の3次元位置データにも注意する必要がある．

2.3.3 三角測量にともなう不可計測領域

レンジファインダによる3次元計測は三角測量に基づくものであるため，データとして獲得できるのは，カメラの視点とレーザの照射位置の両方から見える領域だけである．また，その領域の中でも，レーザ・スリット面と物体の表面の接平面とが平行に近ければ，光量の減少のためその物体の3次元データは獲得できない．さらに，ノイズを除去し精度の高い測定を行うために，カメラ画像の二値量子化の閾値を高くしたり，カメラのレンズの絞りを絞ったりすることにより，獲得できる3次元データの量は一段と減少する．

2.4 エッジ要素の利用

前節でも触れたとおり極限作業ロボットが作業を行う環境は，狭いスペースに多くの物体があり，センサ (レンジファインダ)と対象物体が理想的に配置されているとは限らない．そのため，物体の3次元データが充分に獲得できた場合でも，物体の見え方の角度等などにより，面ベースでの位置・姿勢を決定する際に自由度が残ってしまうような，必要な面の組み合わせが充分に存在しないデータしか得られないことも往々にして起こりうる．そのため本研究では，面ベースで必要な面の組み合わせが存在しない場合に形状データと観測データ間において面を形成するエッジ成分を一致させることで，より過酷な環境での位置・姿勢の決定を可能にする．まず，前節までで導出された基礎領域を面の単位とし，エッジ要素の検出を行う．検出されたエッジ要素を，形状データの3次元の位置情報と方向ベクトル情報によって比較し，面ベースによる一致で残った不定自由度をなくすことによって対象の位置・姿勢を決定する．得られている面データと位置・姿勢の組み合わせのために必要なエッジ成分の関係を以下に示す．

(i) 面が一つのみ一致する場合

計測された3次元データから導出された面データと，形状データの面が一つのみ一致する場合，物体を構成する2つのエッジが一致することによって対象の位置・姿勢を決定することができる．

(ii) 面が二つ一致する場合

レンジデータより導出される面データと，形状データの面が二つ一致し，不定自由度が残る場合，物体を構成するエッジが一つ一致することで位置・姿勢を決定することができる．

2.5 処理の手順

ここで，本研究の具体的な実行システムの概略を説明する．

レンジファインダによって得られた3次元データに前処理が行われる．はじめにレンジデータを面素と呼ばれる微小領域に分割する．分割した各面素が近似的に平面であるとみなし，各面素について平面方程式を導出する．次に，その方程式のパラメータの類似性によって2値画像処理の領域分割に利用するラベリングのアルゴリズムを基に，それらの面素を領域分割し，いくつかの基礎領域に統合する．本研究の手法においては，物体の各面に含まれる複数の基礎領域をオペレータが指示することによって物体の位置・姿勢を決定する．さらにラベル付けされた基礎領域から，領域ごとのエッジ検出を行う．レンジデータの取得からエッジ検出までの流れは，3章で詳しく説明する．

一方で物体の形状データベースより，物体の位置・姿勢を決定するために使用する面の組み合わせにエッジ要素を導入し，順付けて並べた教示ツリーをあらかじめ構築しておく．はじめに，物体のアスペクト(見え方)をデータベースから導く．次に2．2において述べた不定自由度を残すことなく位置・姿勢を決定できる面の組み合わせを各アスペクトに対して導出する．それらの組み合わせを順序付けして並べた教示ツリーを構築する．ツリーの順序付けは，ロボットの作業内容(タスク)を考慮し，タスクに依存性の強い面ほど優先順位を高くすることにより，タスク指向の強い，よりロボット作業を重視したモデリングができるようになっている．次に観測されたデータに位置・姿勢を決定する面の組み合わせが存在しない場合を考慮し，そのときの面の組み合わせにエッジ要素を加えたツリーを再構築する．教示ツリーについては，4章にて詳しく説明する．

次に教示ツリーを用いて幾何モデリングを実行する．ここで，2．3で述べた望ましくない条件が発生すれば，獲得したレンジデータの状況によって位置・姿勢を決定するのに利用する面の組み合わせが変化する．そのとき，オペレータは，得られた3次元データと教示ツリーを比較してツリーの中から，一つの組み合わせを選択する．それゆえ，物体の位置・姿勢はユニークに決定され，その物体の形状データから，モデルをモニタ上に表示することができる．そのとき，さまざまな誤差のために，導出したモデルとレンジデータが一致しなければ，オペレータが修正を行う．

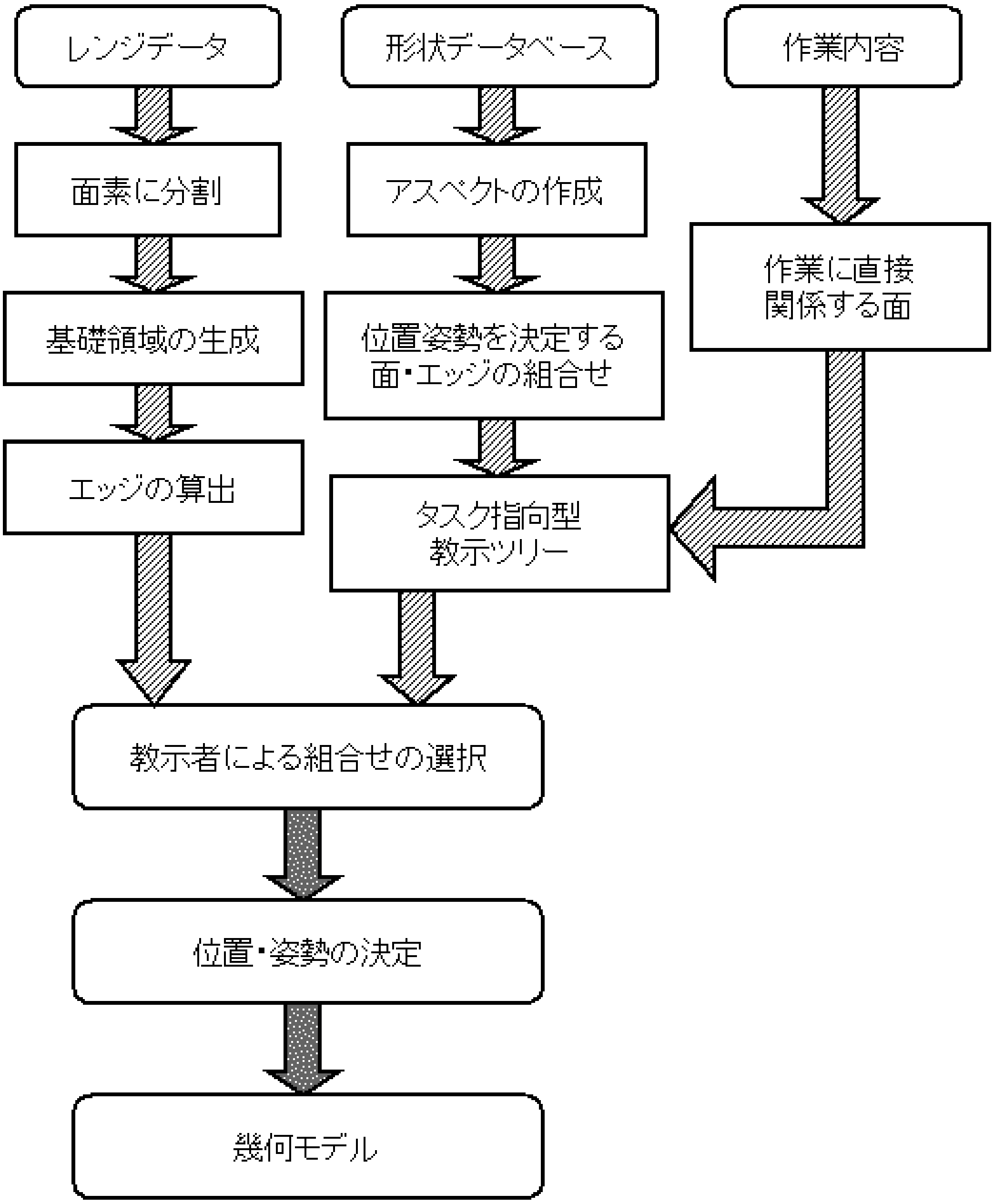
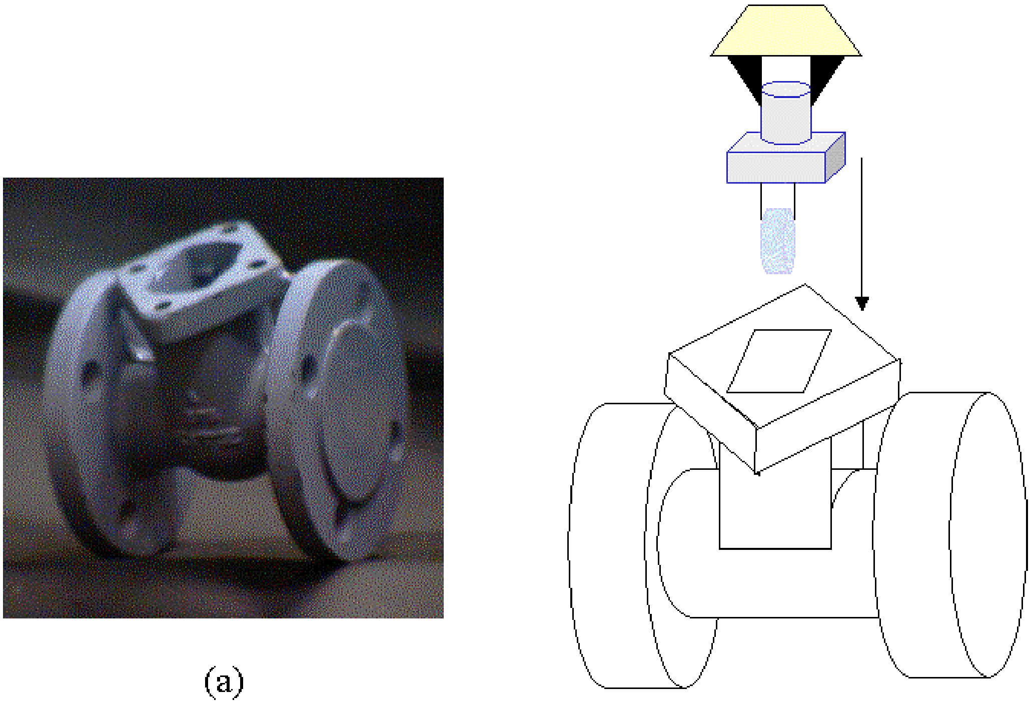

レンジデータ
形状データベース
作業内容
面素に分割
アスペクトの作成
作業に直接
関係する面
基礎領域の生成
位置姿勢を決定する
面・エッジの組合せ
エッジの算出
タスク指向型
教示ツリー
教示者による組合せの選択
位置・姿勢の決定
幾何モデル

図5.　　処理の手順

2.6　対象物と作業内容

実際にロボットが行うべき作業として，本研究においてはバルブに弁を埋め込む作業を取り上げる．下図　6．(a)のようなバルブに対して略図　6．(b)のような埋め込み作業を想定して，以後進めていく．



(a)　(b)

図6.　バルブとロボットタスク

第3章　3 次元データの取得と前処理

3.1　3 次元データの取得

本研究では，作業環境の 3 次元データの獲得にレンジファインダ VIVID 700 (MINOLTA) を用いる．VIVID 700 は光学技術，AF 技術を用いた非接触の 3 次元形状入力装置である (図 7．)．基本原理は、レーザビームによる光切断法を採用している．切断法による 3 次元形状画像入力は，まずシリンドリカルレンズによる水平のスリット光を対象物に照射する．反射光を，CCD で受光し三角測量原理を用いて，ラインの距離情報を得る．次に対象物に照射するスリット光をガルバノミラーで上下方向に走査することによって，シーン全体の 3 次元画像データを得る (図 8．)[17]-[19].

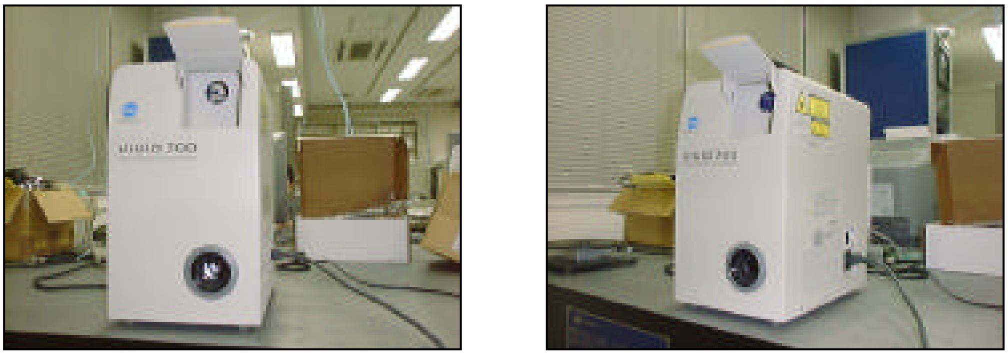


図7.　**3 次元計測器** VIVID 700

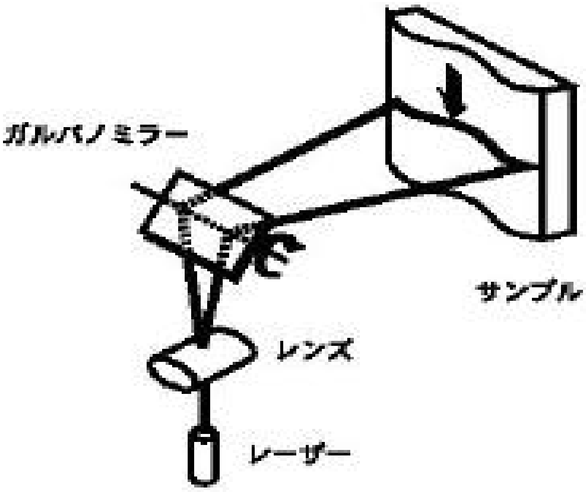

ガルバノミラー
サンプル
レンズ
レーザー

図8. VIVID 700における3次元計測の基本原理

データは200×200の観測点を持ち付属のユーティリティソフトにより40000行からなっており，各行に観測された点の座標(なければ空行)が記されているASCIIデータフォーマットに変換できる．バルブの模型をVIVID 700によって3次元データを計測し，ASCIIフォーマットに変換し，二次元に投影して，表示させたものが図. 9になる．

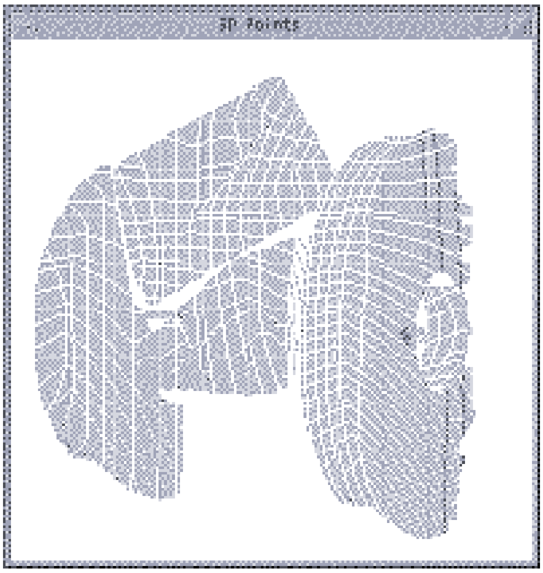


図9. 計測されたバルブの3次元データ

3.2　面素

得られたレンジデータの前処理として，まず，面素と呼ばれる微小領域に分割する．前節で述べたとおりレンジファインダによって計測されたデータは，200×200 の 2 次元配列で番号付けがされている．その中で 3 次元のデータ情報を持つ点においては，2 次元配列で見て隣接する 4 点同士を統合し，これを面素とする．面素を s_i（i＝1,2,3…）と表す．

分割された各面素は，近似的に平面であると見なし，各面素から平面方程式を導出し，法線を算出する．とある点 P_0 に着目し，それを含む面素内における他の任意の 2 点 P_1，P_2 とする．

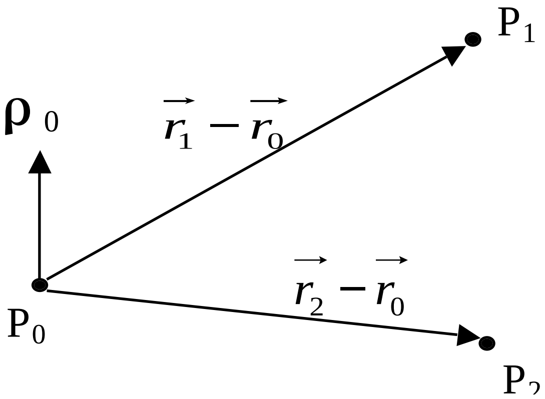


図10.　3 点からの平面の構成

3 点の空間ベクトル成分を $\vec{r_0}, \vec{r_1}, \vec{r_2}$ とすると平面方程式は，以下のように表す．

$$\vec{r}\ \rho_0 = \rho \qquad (1)$$

によって 3 つの線分が構成されるが，このうち 2 つに着目すると求める平面の法線は線分 $r_1 - r_0$ ，$r_2 - r_0$ に直行する．そこで求める平面の法線ベクトルは，

$$\rho_0 = \{(\vec{r_2} - \vec{r_0}) \times (\vec{r_1} - \vec{r_0})\} / |(\vec{r_2} - \vec{r_0}) \times (\vec{r_1} - \vec{r_0})| \qquad (2)$$

から求まる（ここで，×は外積）．これと $\vec{r_0}, \vec{r_1}, \vec{r_2}$ のいずれかの値を（1）に入れることによってρが求まる．

以上の処理によって 3．1 で表示した計測された 3 次元データの各面素から法線データを表示したものを図 10．に示す．

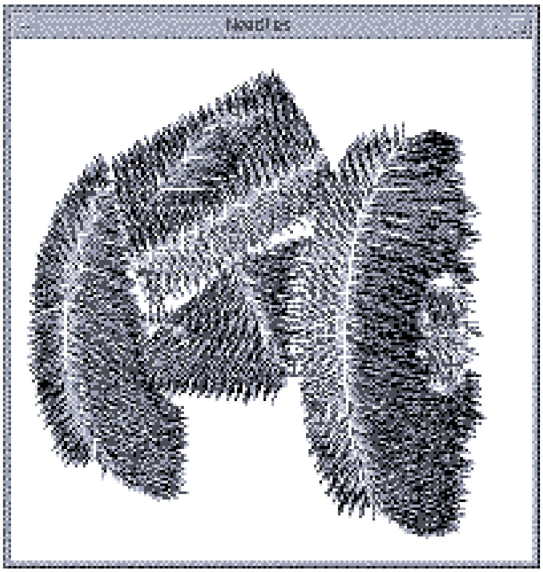


図11.　面素分割による法線の算出

3.3　基礎領域の生成

3．2 によって分割した面素を，導出された平面方程式，とりわけ法線ベクトルの角度を一定の閾値によって類似度の高いものごとに統合し，基礎領域 r_j（*j*=1,2,3…）を生成する．

統合の方法は，2 値画を領域分割する際に利用される伝播法におけるラベリング手法をもとにした距離データラベリングによる．　2 値画像処理における簡単に説明する．

伝播法における 2 値画像のラベリング

2 値画像において画素値を持つ任意の点の隣接して画素値を持つ点をすべて連結させて，その連結成分ごとに異なるラベルを付与することをラベリングという．図　11．を例にあげて伝播法によるラベリングを説明する．まず，画像をラスタ走査し，ラベルのついていない画素値を持つ点ａが見つかると，ａにラベルをつける．次にａに連結する画素値を持つ点ｂ，ｃ，ｄにａと同じラベルをつけるとともに，スタックにこれらの画素を格納する．続いてスタックの一番上に存在する画素ｂを取り出し，ｂに連結する画素値を持ちラベル付けされていない画素に対して同じ処理をする．この操作を繰り返し行うと，スタックの画素がなくなったとき，ａに連結している画素値を持つ画素はすべてラベルが付与され，連結成分が一つに同定できる．この後次の画素からラスタ操作を再開して新しい連結成分の開始点を探索し，同様の処理を行う．[21]

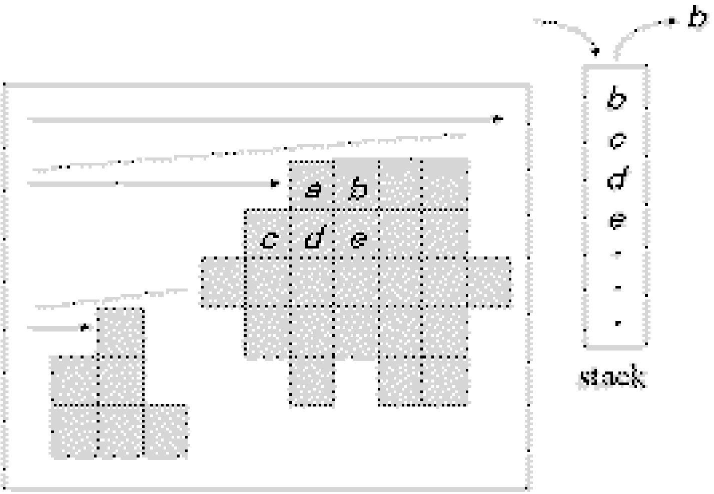


図12.　伝播法による 2 値画像のラベリング

距離データラベリングによる基礎領域生成

2 値画像の伝播法によるラベリングに基づき，　3．2 で導出した面素を単位として得られた 3 次元データをラベリングし，基礎領域を以下の手順で生成する．

① レンジデータを，2 次元にラスタ走査し（レンジファイダによって観測されたときに，データは 200×200 の 2 次元の番号付けがなされている），3 次元データを持ちラベル付けがされていない点が見つかると，その点を含む面素を l_0 とし面素を含む点にラベルをつける．またこのときの l_0 を種として格納しておく．

② ラベルをつけた点の近傍をすべて探索し，ラベルがついていない 3 次元データを持つ面素がある場合，その面素の法線ベクトルの成分と格納してある種の法線ベクトルの成分とを比較し，一定の閾値より低い値を示した場合，その点に種 l_0 と同じラベルをつける．ラベルをつけた面素は，ラベル付けされた順にスタックに格納する．

③ スタックの中から一番早くラベル付けされた面素を取り出し，その面素に対して（2）の処理を繰り返す．

④ （2），（3）の処理をスタックの中の面素がなくなるまで繰り返す．これによって種 l_0 に連結し，かつ l_0 と法線ベクトル成分の類似度の高い面素同士に同一のラベルが付与され，基礎領域 $\boldsymbol{r_j}$ が生成される．

⑤ 一つのラベルを付与し終わると，再びラスタ走査し，（1）から以下の処理をすべての面素がラベル付けされるまで繰り返す．

以上の方法によって生成された基礎領域を図 13．に示す．

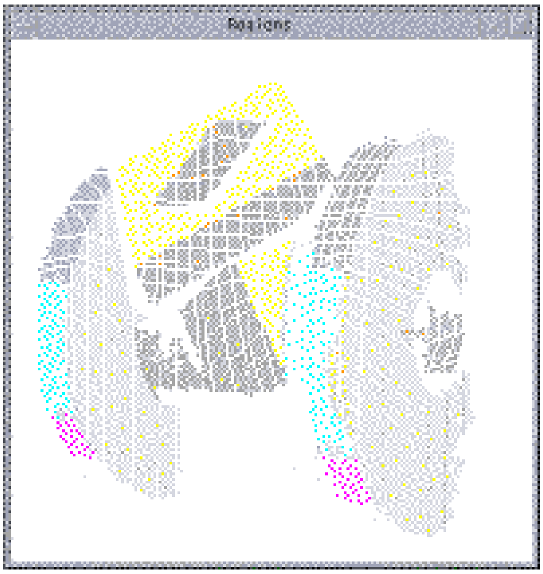


図13.　ラベリングによる基礎領域

生成された基礎領域 r_j は観測された 3 次元データの面データの一部として，オペレータが指示することによって物体の位置・姿勢の決定に用いる．物体の各面 F_j（j=1,2,3…）に含まれる複数の基礎領域をオペレータが指示することによって物体の位置・姿勢を決定する．このとき，指示は下記にしたがって行う．

(i) 面 F_j に含まれる基礎領域 r_j が一つの場合

その基礎領域 r_i を指示する．

(ii) 面 F_j に含まれる基礎領域 r_i が複数の場合

その面 F_j に含まれる一つまたは複数の基礎領域 r_i を指示する．このとき多くの基礎領域を指示すれば，物体の位置・姿勢の精度は高くなる．但し，計算量は多くなる．

3.4　エッジ検出

前節で基礎領域を生成し，領域ごとにラベル付けされたものに対してエッジ要素 e_k(k=1,2,3,…)の検出を行う．処理の手順は，以下に示す．

① ラベル付けされた各データ点において，同一ラベルの累積が一定の値より少ないものに対してそのラベルを 0(領域として無効を意味する)とする．

② 計測された 3 次元データを 2 次元にラスタ走査し，ラベル付けされている点が見つかればその近傍にある 3 次元データを持っている点を探索する．

③ 近傍の点すべてが同一のラベルを持っている点でなければ，その点が基礎領域のエッジの要素となる点であると仮定し，その点にエッジ要素 e_k であるという情報を格納する．

このようにして基礎領域からエッジ要素である点のみをピックアップし，表示したものを図 14．に示す．

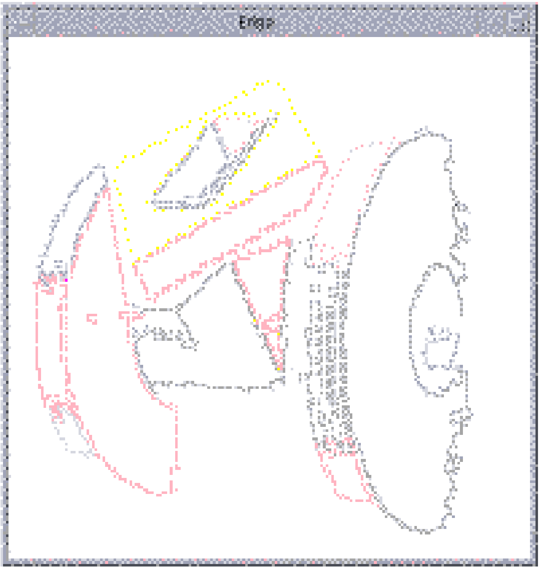


図14.　検出されたエッジ要素

第4章　タスク指向型教示ツリーの構築と位置・姿勢の教示

多くの工業製品は，定められた規格に基づいて設計されており，その図面などを基にして形状をデータベースとして保存しておくことが可能である．本研究では，その保存されたデータベースを物体の位置・姿勢の決定に利用する．与えられている形状データベースから，位置・姿勢を決定するために必要な面，エッジの組み合わせを物体の見え方別にツリー構造化する．オペレータは物体の位置・姿勢が決定できる面・エッジの組み合わせを複数の中から，一つ選択することを行う．ツリーには，優先順位が付けられており，その順序は，求められる作業内容（タスク）と密接に関連付けられている．これによりオペレータにかかる負担の軽減が実現される．本章では，2.6 で触れたバルブのその埋め込み作業について，物体の形状データから位置・姿勢決定のための教示ツリー構築までの流れを説明する．

4.1　形状データベース

本研究では，形状のデータベースは，与えられている図面から，ソリッドモデラ「SOLVER」というツール [22]-[25]を用いて 3 次元モデルを作成し，それをデータベースとして保存する．また，保存されたデータは，設計・生産技術において図面データの交換のために業界標準的に利用されている CAD フォーマットである DXF によって書き出される．

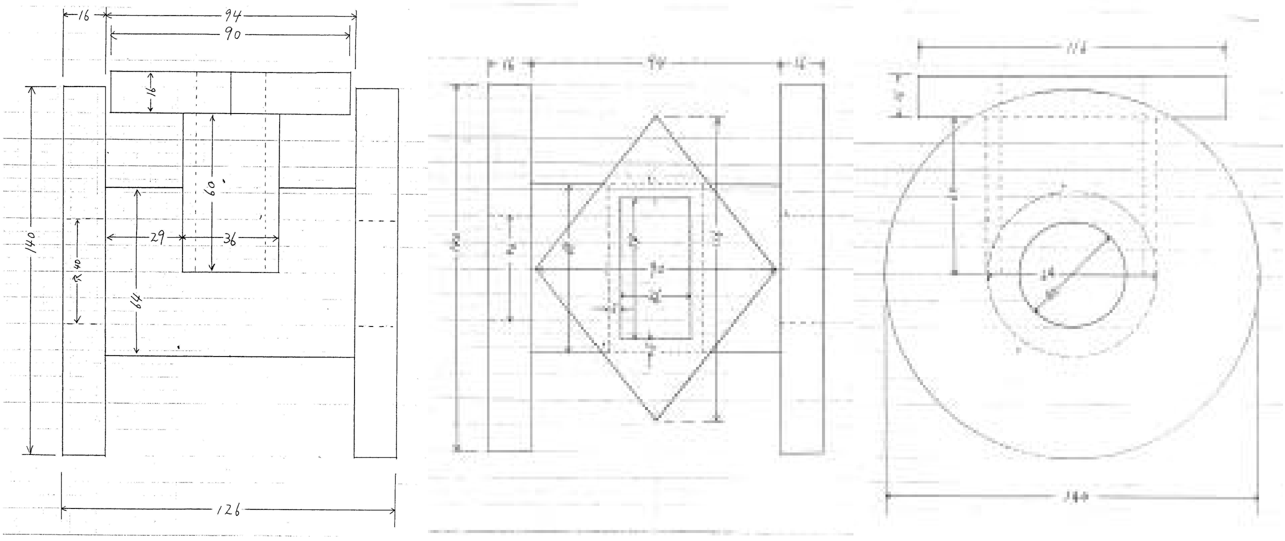
4.1.1　SOLVER による形状データの作成

ソリッドモデラ，は，“ 物体認識 ”（ロボットの視覚）のためにモデルを生成し，計算機内に任意の立体をつくり表示させるツールである．このツールでは，作成した立体を保存，回転・並進，合成することが可能である．パラメータを与えることによって基本となる素立体を作成し，それに対して回転・並進や合成の処理を行うことによって様々な物

体の 3 次元形状を作成することができる．「 SOLVER」によって生成される基本素立体は，以下の 5 つである．

- 直方体
- 球・楕円体
- 回転体
- 凸多面体
- 円錐・円錐台

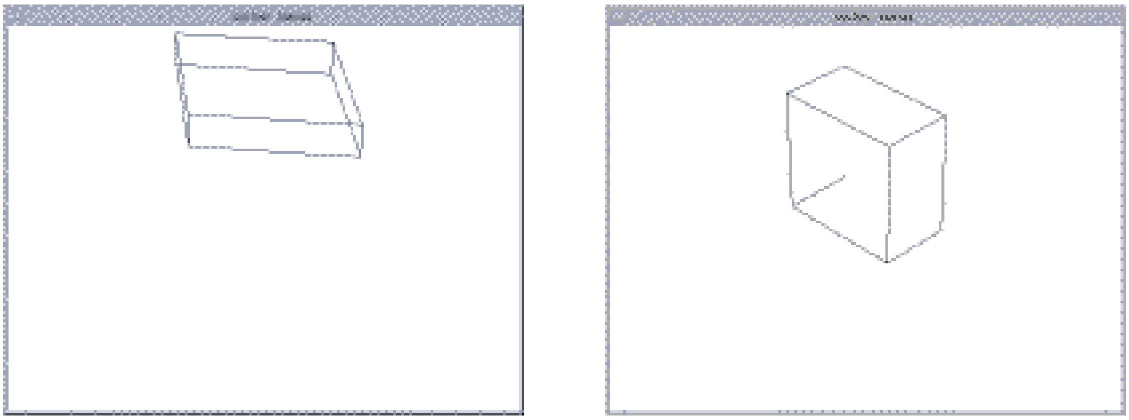
本研究で実験に使用するバルブについて，図 15．の設計図を基に 3 次元形状を作成する過程を以下に記す．



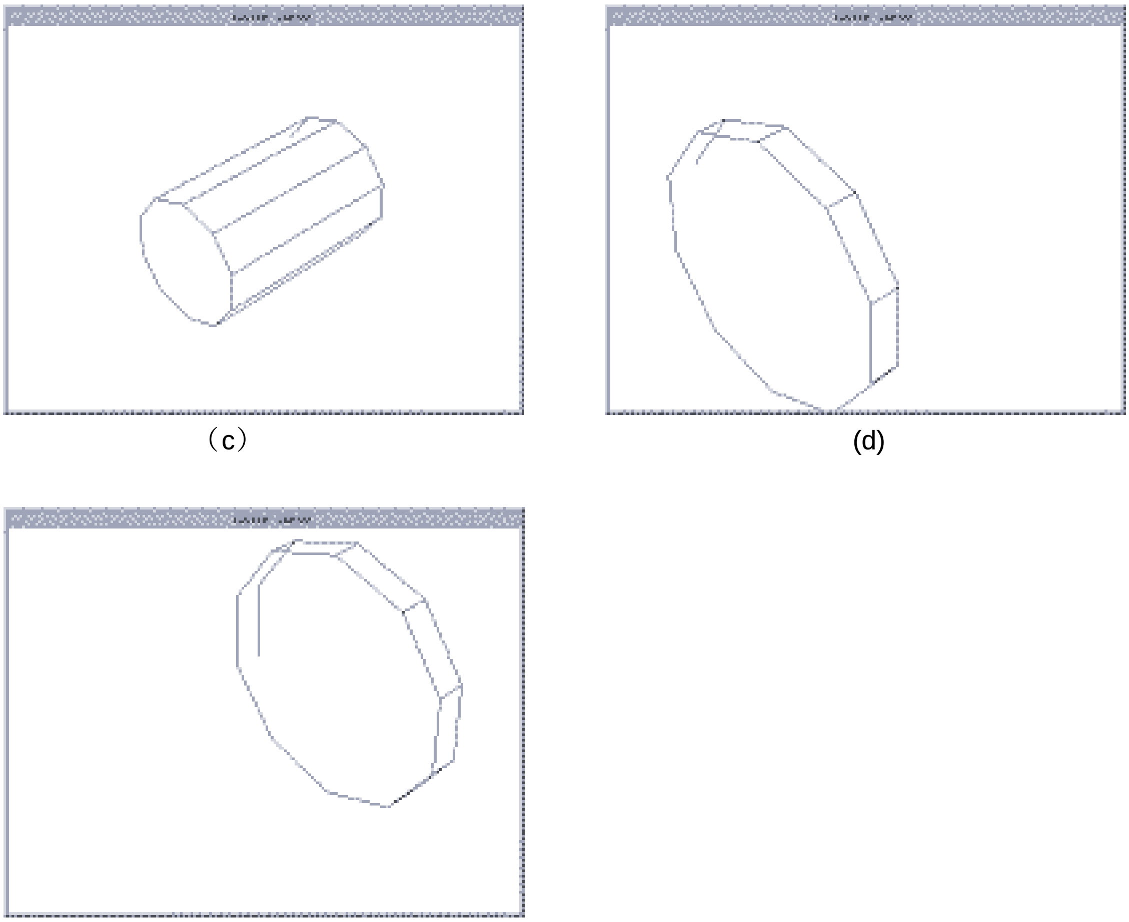
16
94
90
16
60
29
36
140
40
64
126
116
140

図15.　バルブの設計図

(1)　素立体 (a)～(e)（図 16．）を作成する．



(a)　(b)



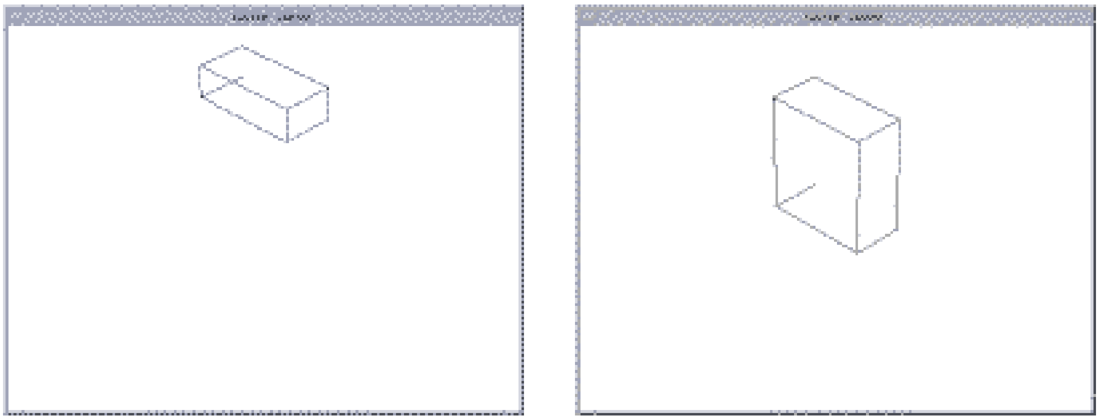
（c）

(d)

(e)

図16.　素立体の生成　(1)

(2)　素立体 (f)，（g）（図 17.）をそれぞれ作成する.



(f)

(g)

図17.　素立体の生成　(2)

(3)　(1)において作成した素立体同士を合成（和集合）する（図 18.）.

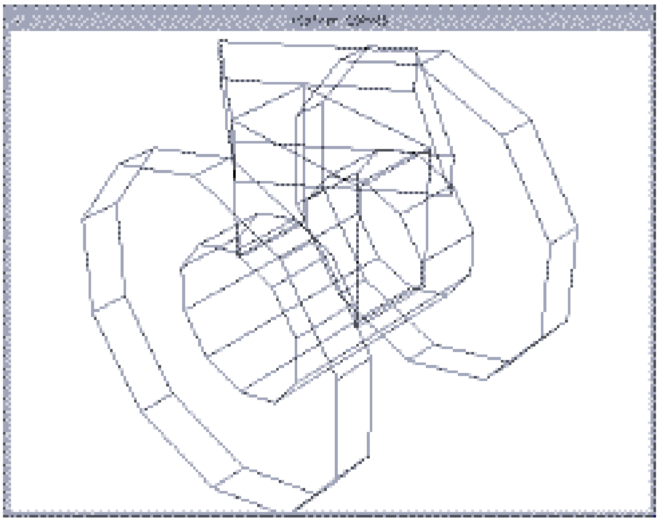


図18.　合成した（和集合）物体

(4)　(3)で合成した物体から (2)において生成した物体をそれぞれ引く（図 19.）.

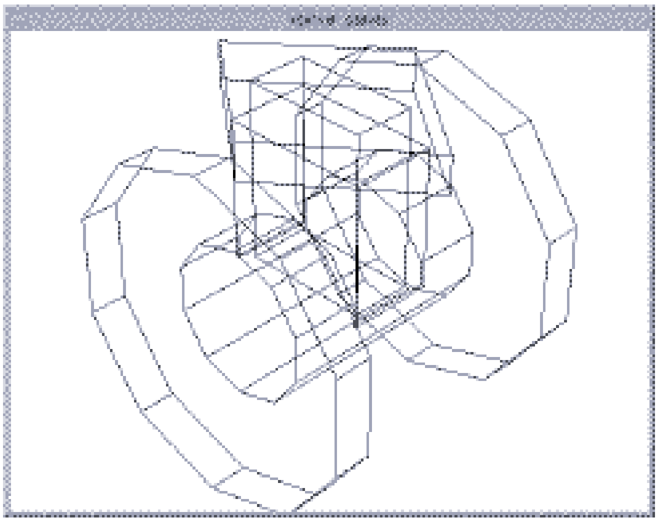


図19.　(3)から (2)を引いたもの

このようにして作られたバルブの形状データを図 20. に示す.

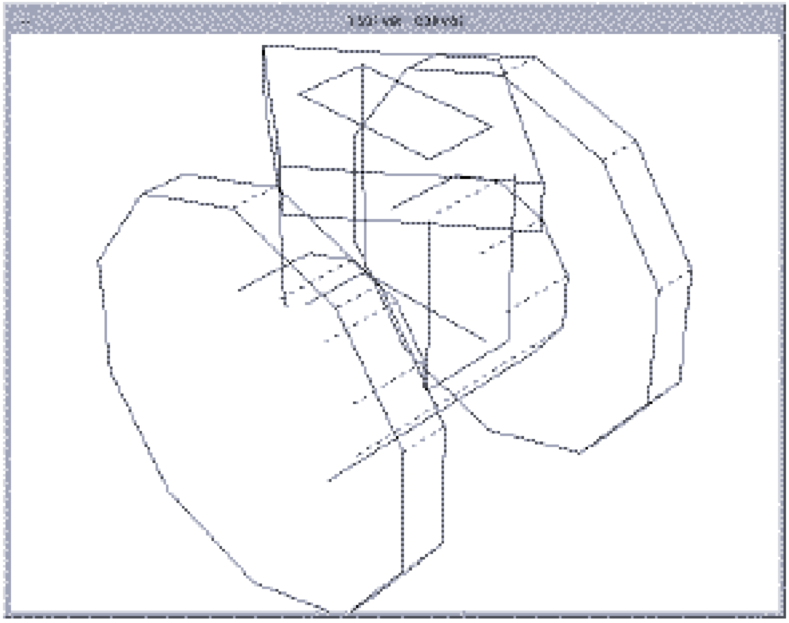


図20.　バルブの形状データ

4.1.2 DXF フォーマットによるデータ表現

DXF(Drawing Interchange File)は，もともとはオートデスク社製の CAD システム「AutoCAD」で使用される外部ファイル形式であり，オートデスク社により定められた．その外部ファイル形式は，2 次元および 3 次元形状をサポートし，他のメーカでも広く利用されている．DXF は ASCII ファイル形式で記述され，CAD システム同士の中間データ形式として普及している．当初，DXF は AutoCAD 間の図面データの互換性などを対象としており，異なった CAD システム間のデータ交換用ファイルではなかった．しかしながら，AutoCAD が世界中で使用されるようになると，他の CAD システムも AutoCAD の図面データを取り込む機能を備えるようになった．また，他の CAD システムが DXF をサポートできるようにするために，オートデスク社は，積極的に仕様などの情報を公開している．それにより現在では多くの CAD システムで用いられ，中間ファイルフォーマットとしては，他のフォーマットに比べて、ほぼ業界標準に近い位置にある．

DXF ファイルの 3 次元データ構造

DXF 形式のファイルは，ASCII コードのテキスト形式のファイルである．その構造は，大きく分けてヘッダ部，データ記述部，フッタ部の 3 つからなる．DXF において形状の 3 次元データを記述する場合，そのデータは通常 4 点から構成されるポリゴンパッチを組み合わせることにより表現される．データ記述部において，一つ一つのポリゴンデータについて毎回ポリゴン面を記述するための宣言を書いてから，そのポリゴンを構成する座標データを記述する（図 21.）．

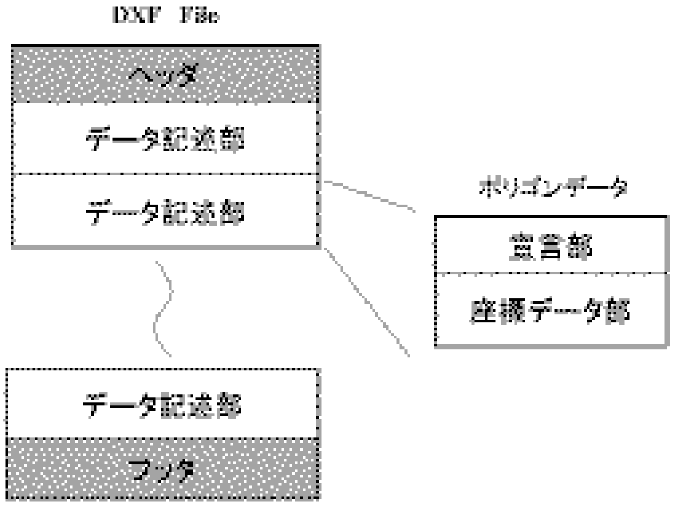


図21.　DXF ファイルの全体構造

本研究において，DXF フォーマットのファイル形式を扱うことに関して，以下のような利点が考えられる．

- 世界標準となっているデータ交換のためのファイル形式であるため，システム全体としての汎用性が高くなる．
- 本研究において用いるレンジファインダ（VIVID 700）や SOLVER が DXF によるデータの書き出しが可能であるため，データの互換性があり，スムーズなモデリングが可能となる．
- ポリゴンパッチによるデータ記述であるために，面ベースでの位置・姿勢の決定には最適である．
- ASCII ファイル形式（テキスト形式）で記述されていて，閲覧やエディタなどで簡単に編集ができ，扱いやすい．

以上のことにより，本研究では，形状データ，距離データの相互のやりとりに DXF フォーマットのファイル形式を用いる．

4.2　アスペクトの導出

2 章で述べた位置・姿勢を決定できる面の組み合わせは，センサと物体の配置から起こるデータの獲得状況，つまり，物体の見え方によって変化する．本節では，物体に対して起こりうる見え方（アスペクト）を，測地ドームを用いたアスペクト法 [26]によって可能な限り列挙する方法を説明する [27]．

4.2.1　測地ドームによるアスペクト法

測地ドーム（geodesic dome）は，球面を比較的一様な小三角に分割するものである．分割は，正20面体を基とし，多数の切子面を持つ多面体を作る．

正20面体は，12個の頂点と，20個の面，30個のエッジを持っている．このとき，その中心がデカルト座標系の原点にあり，かつ，各頂点は中心から単位距離にあるものとする．そこで，

$$t\text{（黄金分割比）}=\frac{1+\sqrt{5}}{2} \tag{4.1}$$

$$a=\frac{\sqrt{t}}{5^{1/4}} \tag{4.2}$$

$$b=\frac{1}{\left(\sqrt{t}\,5^{1/4}\right)} \tag{4.3}$$

$$c=a+2b=\frac{1}{b} \tag{4.4}$$

$$d=a+b=\frac{t^{3/2}}{5^{1/4}} \tag{4.5}$$

$$\begin{aligned} A&=\text{エッジが原点において張る角度}\\ &=\mathrm{across}\left(\frac{\sqrt{5}}{5}\right) \end{aligned} \tag{4.6}$$

と定義すると，

半径と一つのエッジの間の角度$=b=\mathrm{across}(b)$

エッジの長さ$=2b$

原点からエッジ中心までの距離$=a$

原点から面中心までの距離$=\dfrac{ta}{\sqrt{3}}$

となる．12の頂点は次の位置になる．

$$(0,\pm a,\pm b)$$

$$(\pm b,0,\pm a)$$

$$(\pm a,\pm b,0)$$

したがって 20 面体の面の中心は次のように与えられる .

$$1/3(\pm d,\pm d,\pm d)$$

$$1/3(0,\pm a,\pm c)$$

$$1/3(\pm c,0,\pm a)$$

$$1/3(\pm a,\pm c,0)$$

正 20 面体の面をさらに細かく分割するための，いくつかの方法から考えられる最も簡単なやり方は，各エッジを n 個に等分して n2 個の合同な三角形を作る．そして，それらを球面の半径方向に押し出して，それらの最後の位置を決めるようにする．

このようにして本研究では， 80 の小三角を用意し，各三角の重心点をビューポイントとし球の中心にある物体を観測する．

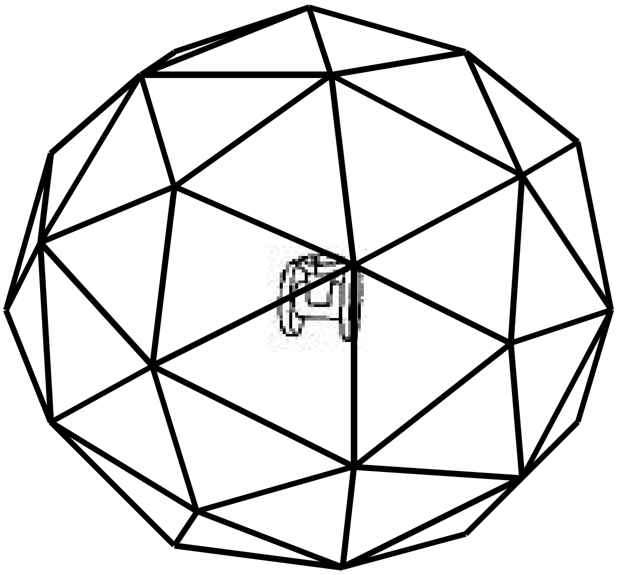


図22. 測地ドーム

このようにして各ビューポイントからアスペクトを導出する．なお，本研究で実験に利用するバルブにおいては，弁の埋め込み作業というタスクという観点から，バルブの上半分のみを考える．さらに，バルブの形状の対称性を考慮し， 1／8 半球に含まれる小三角の重心をビューポイントとして扱う．これらのビューポイントから観測されたアスペクト群を図 23．に示す．

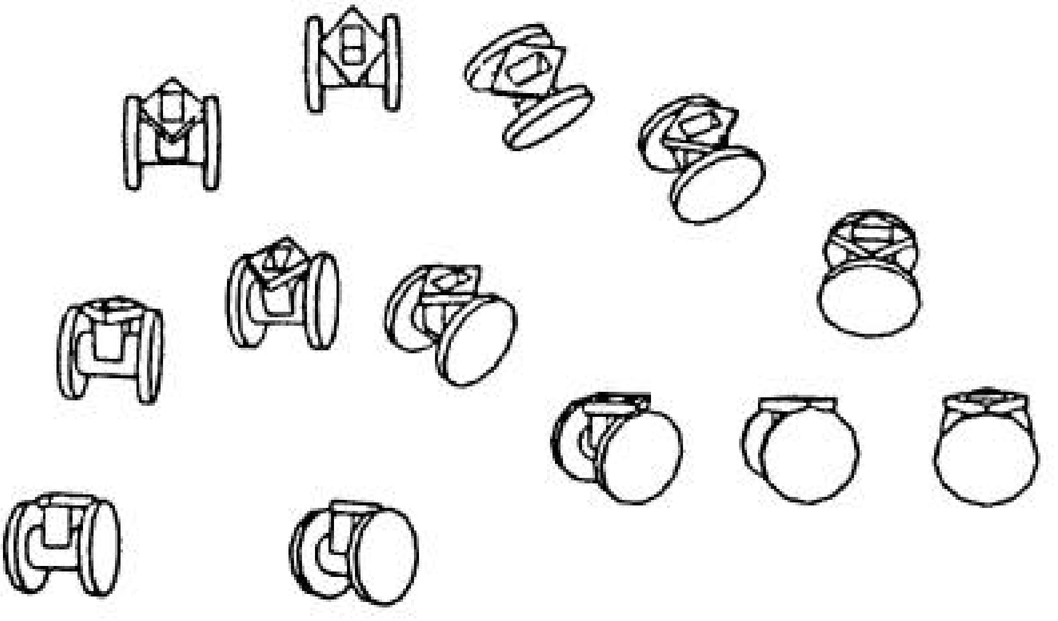


図23.　導出されたアスペクト群

4.2.2　位置・姿勢を決定に利用する面

ここで, 対象物体であるバルブに対して, 位置姿勢を決定するために用いる面　F_m(m＝1,2, …,6) を選択する (図 24.).

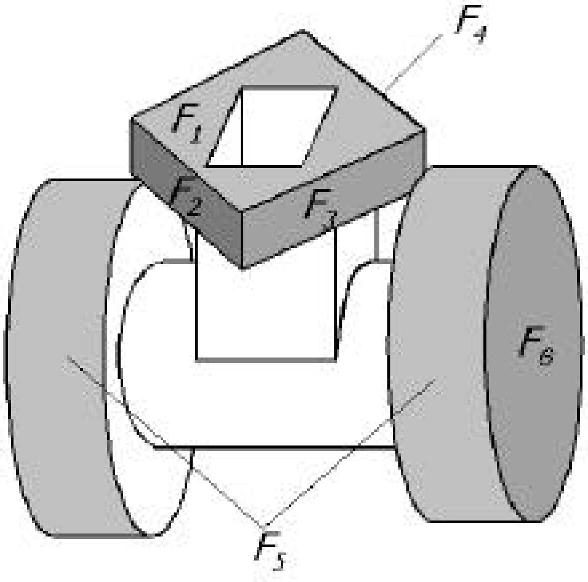


図24.　教示面の選択

このような 8 つの面を位置・姿勢の決定のために選択するとした場合，アスペクトごとに教示に必要な面がそれぞれ変わってくる．また，同一の面が見えている場合もあり，このとき位置・姿勢の決定には同じ組み合わせを適用することとなる．このような同一の面が見えている場合は，それらを同じアスペクトグループとして，グループ化することができる (図 25．)．このようにアスペクト群をグループ化したものを代表アスペクトとする

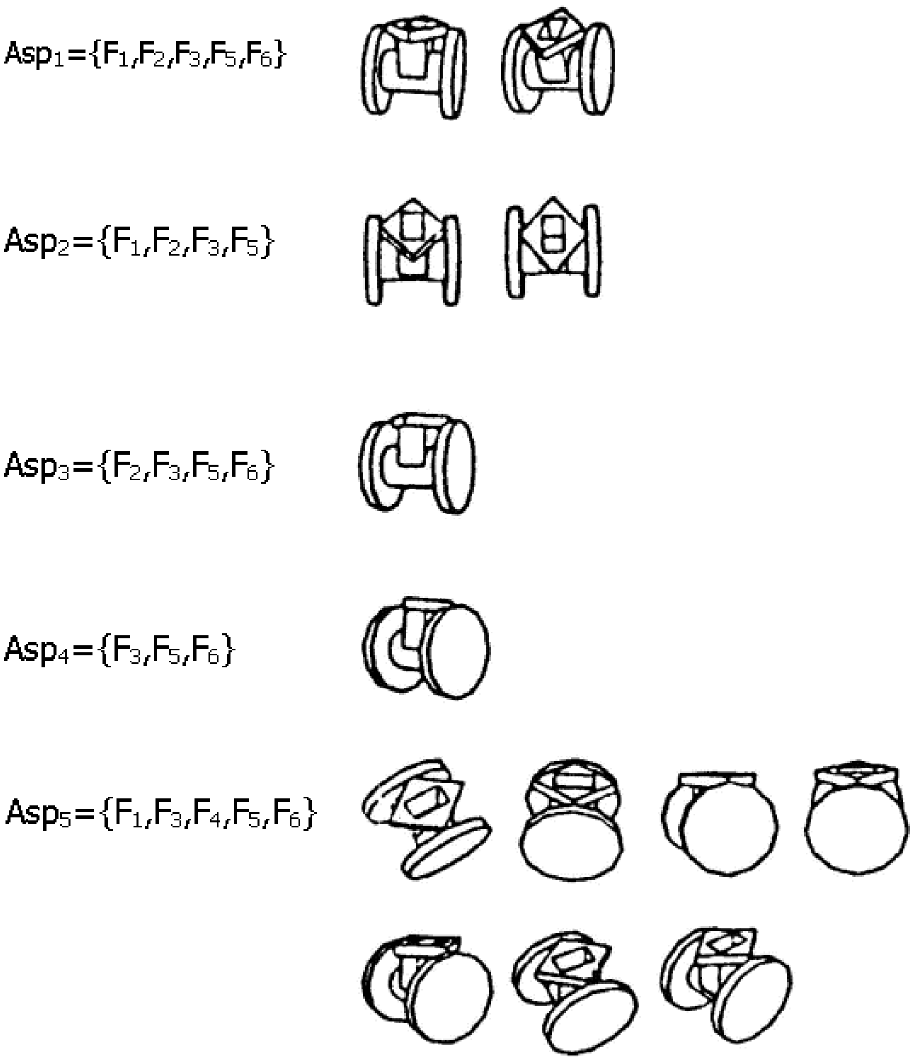

$Asp_1=\{F_1,F_2,F_3,F_5,F_6\}$
$Asp_2=\{F_1,F_2,F_3,F_5\}$
$Asp_3=\{F_2,F_3,F_5,F_6\}$
$Asp_4=\{F_3,F_5,F_6\}$
$Asp_5=\{F_1,F_3,F_4,F_5,F_6\}$

図25.　アスペクトグループ

4.3　ツリーの構築

前節において導出したアスペクトグループに対し，位置・姿勢を決定するために　2．2 で述べた面の組み合わせを導入して，教示ツリーを構築する．また，ツリーを構築する際にタスクとの関連性を考慮し，作業に関係した面に対して優先順位を持たせる．

作業に関連した面

マニピュレーションプランニングにおいて，環境をモデリングすることは，最初に行われる作業であると考える．ここで重要となるのは，単に物体の位置・姿勢を決定することではなく，与えられた環境から，作業の計画・行動系に対して，抽出すべき情報であることである．そこで本研究では，ロボットが行うべき作業内容を把握し，教示ツリーを構築する際にタスクに関連した面に関して優先順位を持たせることによってタスク指向の強い教示ツリーを構築する．

2．6 において述べた本研究において求められるタスクは，バルブの上面にある孔に弁を埋め込む「挿入タスク」である．この作業においてタスクに最も関連性の強い面は，図 24．F2 ということになる (図 25．).

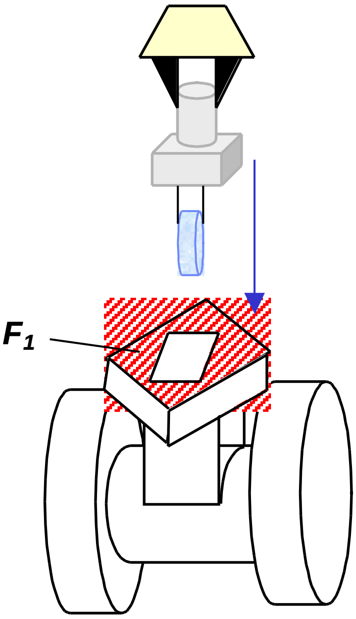


図26.　タスクに関係する面

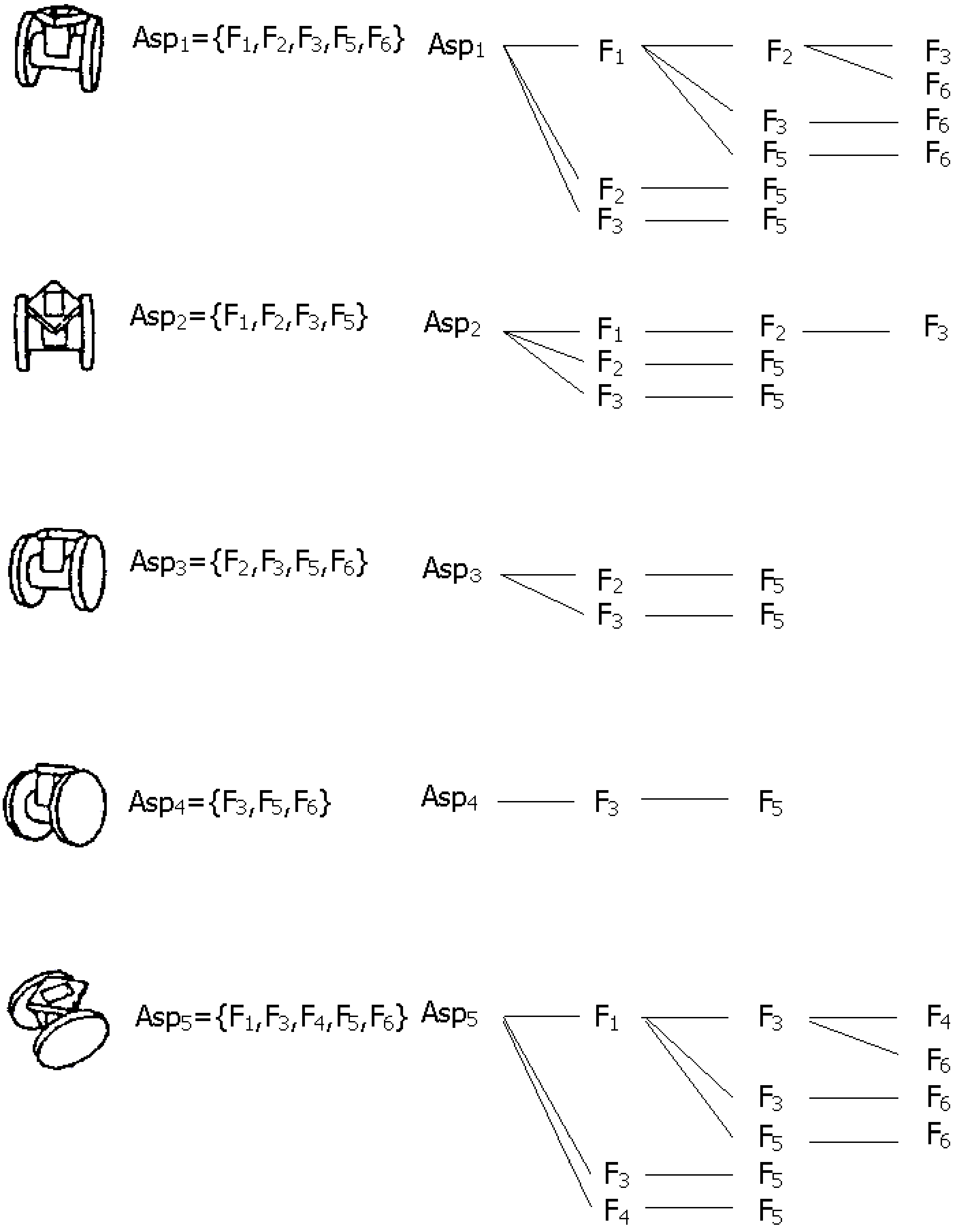
以上のことを踏まえ，以下の条件で優先順位を持たせたツリーを構築する．

(ア)　タスクに直接関係した面（ F2）を含む面の組み合わせを最優先する．

(イ)　1．に接している面を含む組み合わせを上位にする　(ここでは ,F2,F3,F4 がそれにあたる).

(ウ)　　教示面の数が少ないものを上位にする.

この条件のもとで，アスペクトグループと位置・姿勢を決定するための面の組み合わせにより，構築したタスク指向型教示ツリーを以下に示す.



$Asp_1=\{F_1,F_2,F_3,F_5,F_6\}$
Asp_1
F_1 F_2 F_3
F_6
F_3 F_6
F_5 F_6
F_2 F_5
F_3 F_5
$Asp_2=\{F_1,F_2,F_3,F_5\}$
Asp_2
F_1 F_2 F_3
F_2 F_5
F_3 F_5
$Asp_3=\{F_2,F_3,F_5,F_6\}$
Asp_3
F_2 F_5
F_3 F_5
$Asp_4=\{F_3,F_5,F_6\}$
Asp_4 F_3 F_5
$Asp_5=\{F_1,F_3,F_4,F_5,F_6\}$
Asp_5
F_1 F_3 F_4
F_6
F_3 F_6
F_5 F_6
F_3 F_5
F_4 F_5

図27.　面ベースによるタスク指向型教示ツリー

4.4　エッジ要素

本研究では，観測された 3 次元データから前処理を行って導出された基礎領域に，その物体の位置・姿勢を決定できる組み合わせが含まれなかった場合に，面を構成するエッジ要素を用いて位置・姿勢を決定する．ここで，4.4.2 と同様に実際に教示に利用するエッジ E_l(l＝1,2,3⋯,15)を選択する．

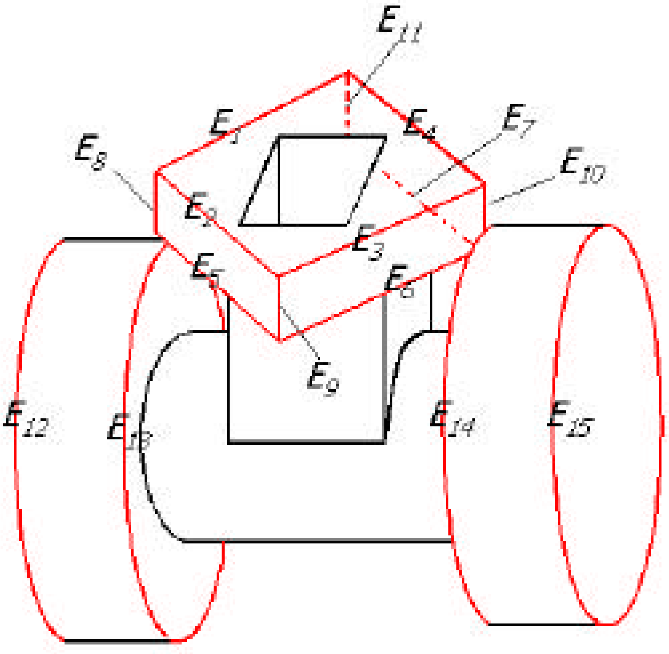

E_{11}
E_1
E_4
E_7
E_8
E_{10}
E_2
E_3
E_5
E_6
E_9
E_{12}
E_{13}
E_{14}
E_{15}

図28.　教示エッジ

4.5　エッジベースツリーの構築

前節で選択されたエッジ要素を，ツリー構造化する．ここでは，観測された 3 次元データに 4.2.2 で選択した面が，最低一つは含まれると仮定して，観測されうる面の組み合わせすべてに対して，その場合に位置・姿勢が決定することができるエッジの組み合わせを考慮し，それを 4.3 と同様にタスク指向に順序付けをし，ツリーを構築する．

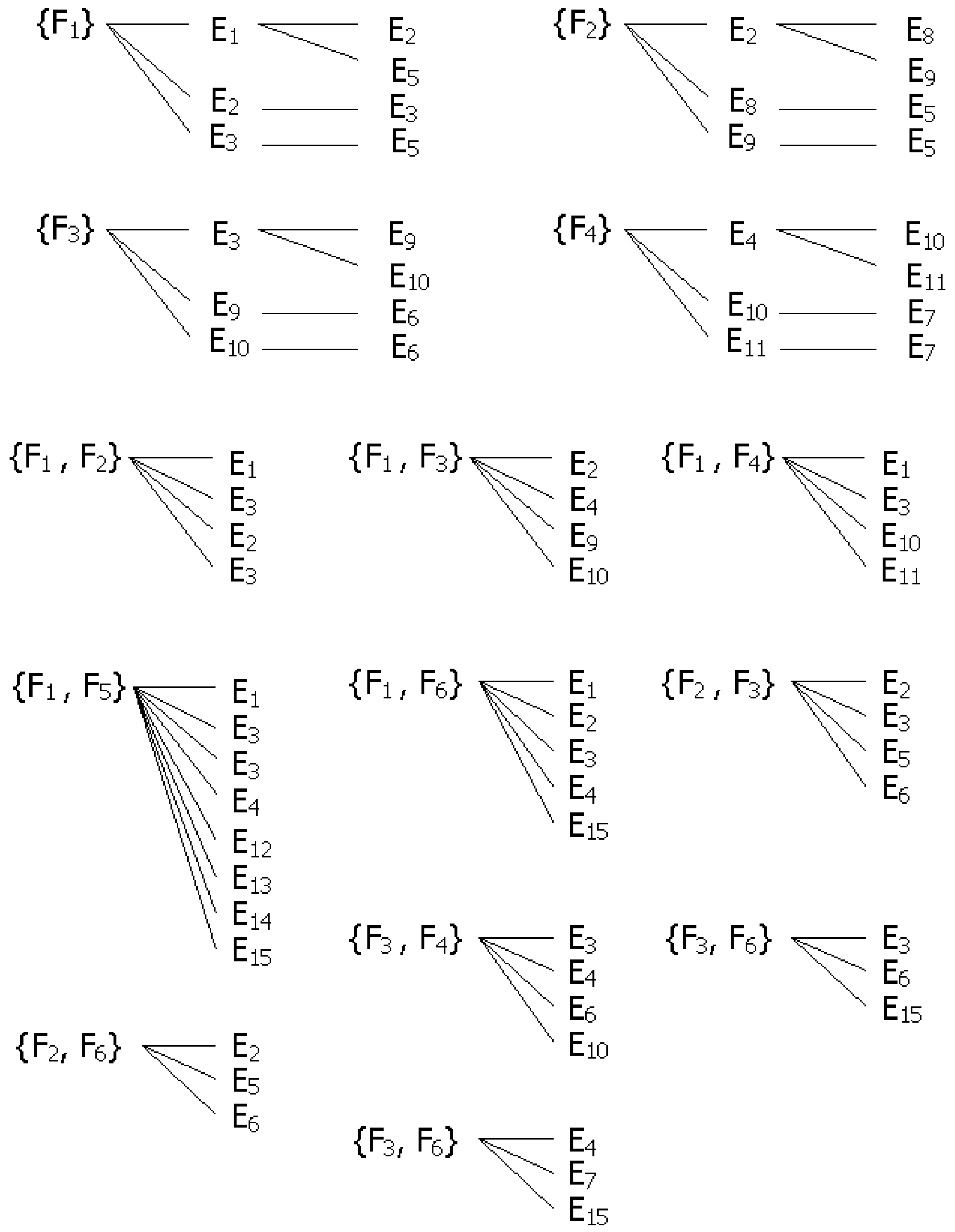

$\{F_1\}$
E_1
E_2
E_5
E_2
E_3
E_3
E_5
$\{F_2\}$
E_2
E_8
E_9
E_8
E_5
E_9
E_5
$\{F_3\}$
E_3
E_9
E_{10}
E_9
E_6
E_{10}
E_6
$\{F_4\}$
E_4
E_{10}
E_{11}
E_{10}
E_7
E_{11}
E_7
$\{F_1, F_2\}$
E_1
E_3
E_2
E_3
$\{F_1, F_3\}$
E_2
E_4
E_9
E_{10}
$\{F_1, F_4\}$
E_1
E_3
E_{10}
E_{11}
$\{F_1, F_5\}$
E_1
E_3
E_3
E_4
E_{12}
E_{13}
E_{14}
E_{15}
$\{F_1, F_6\}$
E_1
E_2
E_3
E_4
E_{15}
$\{F_2, F_3\}$
E_2
E_3
E_5
E_6
$\{F_3, F_4\}$
E_3
E_4
E_6
E_{10}
$\{F_3, F_6\}$
E_3
E_6
E_{15}
$\{F_2, F_6\}$
E_2
E_5
E_6
$\{F_3, F_6\}$
E_4
E_7
E_{15}

図29.　エッジベースの教示ツリー

図の {} 内に示されているのは，4.2.2 で選択された各面である．レンジファインダによって観測された 3 次元データから基礎領域を導出したものの中で，位置・姿勢を決定できる組み合わせが存在しなかった場合に，そのときに起こりうる面の組み合わせが各 {} 内に示されている．そのとき，位置・姿勢を決定するのに必要なエッジの組み合わせが各ツリーに構成されている．構築されたツリーは，4.3 において構築した面ベースによるツリーの一部には組み込まれず，面，エッジそれぞれが独立したものとして与えられる．面ベースのツリーにエッジツリーを導入すると，一つのアスペクトに対してツリーの枝葉の部分が多量になる．実際に位置・姿勢を教示する際，オペレータはツリーの一部を選択するため，一つのアスペクトに対してデータの量が増えると，教示を行うオペレータにとっての負担がその分増大してしまう．面ベースによるツリーとエッジベースのツリーを独立させて構築することは，オペレータにかかる負担を軽減するという点で意味をもっている．観測されたデータから形状の位置・姿勢をオペレータとの対話的処理によって教示する具体的な流れに関しては，次節で説明する．

4.6 教示の流れ

3 章で述べた前処理によって，獲得した 3 次元データより導出した基礎領域，またはエッジ要素と，4 章で述べた形状データより構築した面ベース，エッジベースのそれぞれの教示ツリーをもちいて，対象物体の位置・姿勢を，システムとオペレータが対話的な処理を行うことによって教示する．

まず，レンジファインダによって計測された 3 次元データをモニタ上に表示する．このとき 3 次元データの前処理によって基礎領域，エッジ要素はすでに導出されており，基礎領域ごとに色付けされたものがモニタに表示され，オペレータは獲得された 3 次元データにどの面がデータとして獲得されているか一目でわかるようになっている．表示されたデータから，オペレータは，その見かけの形状によりあらかじめグループ分けされているアスペクトのうちどのアスペクトに属するかを判断し，代表アスペクトの中から一つを選択する．アスペクトが選択されると，システムはそのアスペクトに構築されている面ベースの教示ツリーを，それにつけられた優先順位の高い順に表示していく．ここでオペレータが，表示されたツリーの一つと，獲得されている 3 次元データに基礎領域として反映されている面データとを比較し，面同士の対応が取れているならばそれを位置・姿勢の決定に用いる面の組み合わせとして選択する．ここでシステムが位置・姿勢を決定できる面がすべてデータとして観測されているかどうかをオペレータに対して問いかけてくる．もし，3 次元データ上にそれらの面がすべて計測されているならば，そのツリーに示されている面だけを用いて，エッジツリーは使用せずに面データ（法線ベクトル，位置）の一致のみによって物体の位置，姿勢を決定する．物体の位置・姿勢は，ツリーに示されている面データに対して，それに対応する計測された 3 次元データから，それに対応する基礎領域 r_i（3.3 参照）を一つまたは複数指示することによってなされる．これとは異なり，オクルージョン等により計測された 3 次元データに，ツリーに示されている面のすべてが反映されていない場合の処理について考える．このときにはまず，ツリーに示さ

れている面の中で，計測された 3 次元データに基礎領域として導出されているものを，オペレータが判断し，一つ，もしくは 2 つの面をシステムに入力する．入力された面によりシステムが，その面の組み合わせによる場合のエッジツリーをピックアップし，優先順位の高い順にエッジの組み合わせを表示する．オペレータはその中の一つを選択することによって，位置，姿勢を決定するための面，エッジの組み合わせが決定される．ここで，先ほど述べた面のみの位置姿勢の決定同様，ツリーによって示される面，エッジの組み合わせに対応する基礎領域 r_j，エッジ e_k を選択することによって，物体の位置・姿勢が決定される．このような流れに従って計算機内にあらかじめ保存されている物体の形状データの位置・姿勢が決定されると，最初に 3 次元データを表示したモニタ上に，そのモデルを重ねて表示する．そのとき，観測された 3 次元データと導出されたモデルが様々な誤差のために完全に一致していなければ，オペレータが手作業によりモデルの微調整を行う．

上述した物体の位置・姿勢を教示する一連の流れを図示すると，以下のようになる．

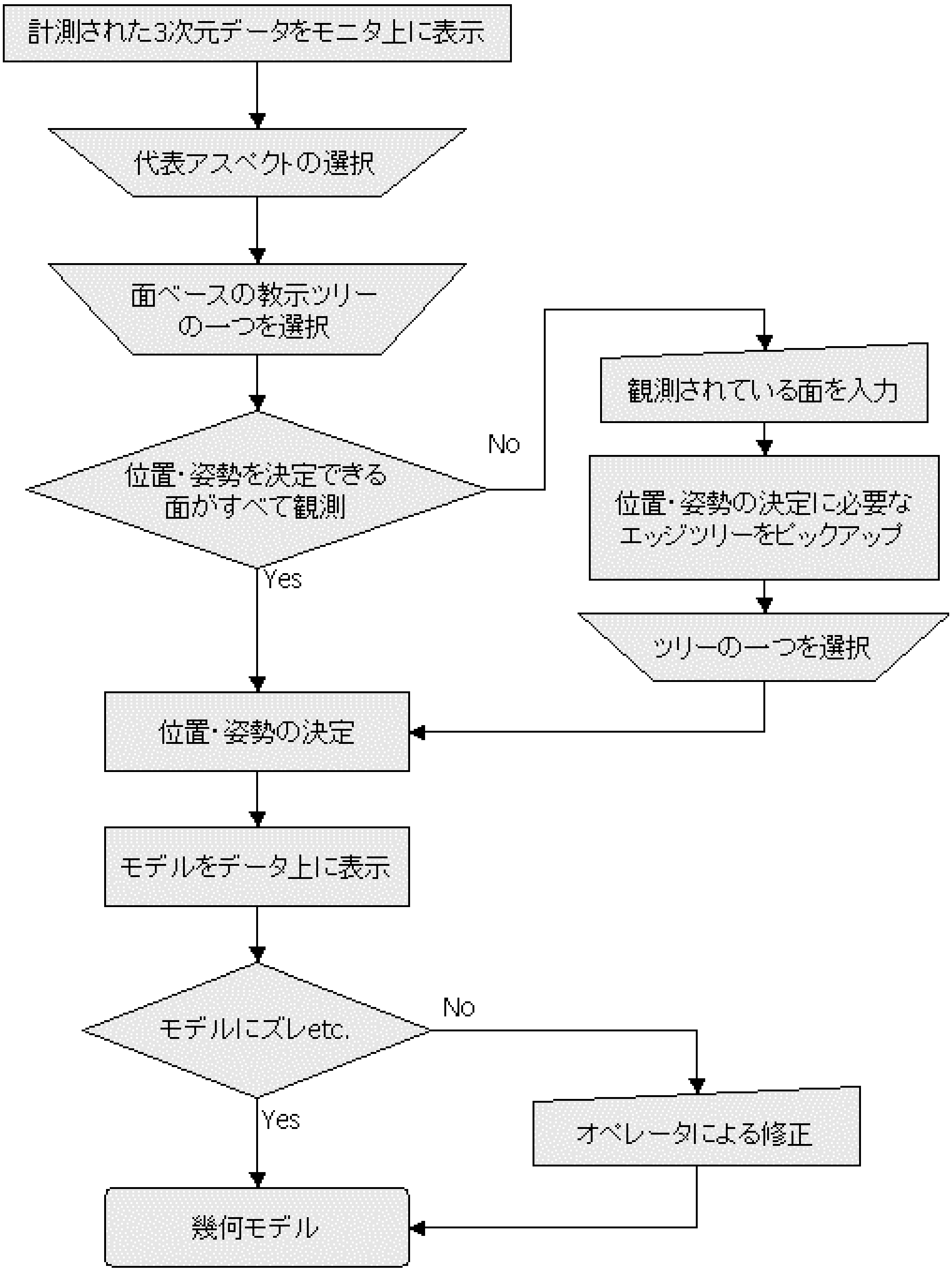
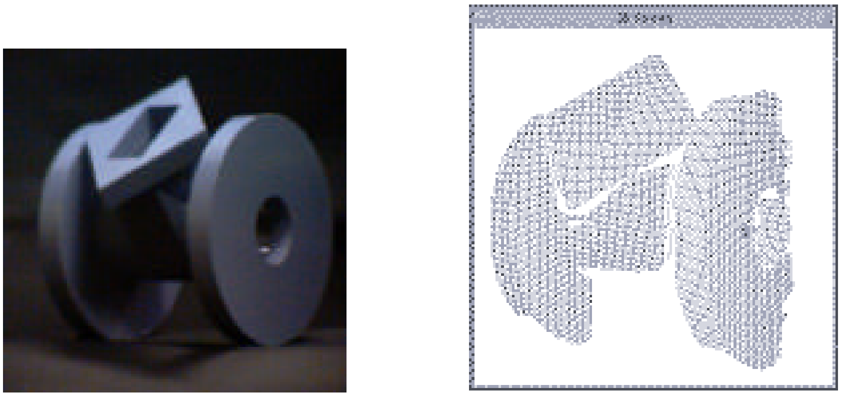

計測された3次元データをモニタ上に表示
代表アスペクトの選択
面ベースの教示ツリーの一つを選択
位置･姿勢を決定できる面がすべて観測
No
Yes
観測されている面を入力
位置･姿勢の決定に必要なエッジツリーをピックアップ
ツリーの一つを選択
位置･姿勢の決定
モデルをデータ上に表示
モデルにズレetc.
No
Yes
オペレータによる修正
幾何モデル

図30.　教示の流れ

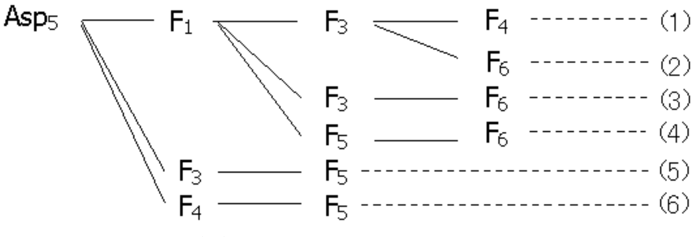
第5章　実験結果

ここで，本研究において例に挙げたバルブの位置・姿勢をこれまで述べてきた処理の流れに沿って教示する実験を行う．ここでは，理想的な環境，面による位置・姿勢の決定が不可能である場合，オクルージョンがある場合の3通りの場合について実験を行い本研究における手法の有効性を示す．

5.1　理想的な環境における場合



（a）カメラ画像　　（b）3次元データ



Asp_5
F_1
F_3
F_4
F_6
F_3
F_6
F_5
F_6
F_3
F_5
F_4
F_5
(1)
(2)
(3)
(4)
(5)
(6)

（c）対応するアスペクト

図31．　理想的な環境における教示データ

まず始めに，障害物などがなく，理想的な状態に配置されている例について処理結果を示す．図31．（a）の状態は，センサと物体の間にはパイプや他のバルブなどの障害物などはなくかなり理想的な状態に配置されている．このような状態にある場合にレンジファインダによって計測された3次元データは，図31．（b）のようになり，鏡面反射を起こしていることもなく，物体の3次元データも十分に獲得されている．この図31．（b）に表示されたデータと4.2.2の図24．によって教示に利用する面とを比較して，オペレータは，4.2.2図25．に示した代表アスペクトの中から，一つを選択する．この場合，図31．（b）に示された図で計測されていることが確認できるのは，F_1，F_3，F_5，F_6である．よって図25．における代表アスペクトの中でAsp5を選択する．なお，この3次元データの獲得状況においては,代表アスペクトの中でAsp1を選択することも可能である（Asp2においてもF_1，F_3，F_6，F_6の要素が含まれているため）．このような3次元データの獲得状態によって2つの代表アスペクトが選択可能な場合は，オペレータが任意にどちらか一つを選択する．次にオペレータは，Asp5に構成されてある，教示ツリーにおける面の組み合わせの中から一つを選択する．図31．（c）においてツリーの各組み合わせに(1)～(6)までの番号を振り，優先順位は(1)が一番高く，以下(2)，(3)・・・となっていく．獲得された3次元データの面（F_1，F_3，F_5，F_6）を含み，その中で一番高い優先順位をもつ面の組み合わせを考えると，（3）がそれにあたり，物体の位置・姿勢を決定するためにオペレータは（3）に示された面の組み合わせを選択する．このようにして，選択された面の組み合わせを用いて，物体の位置・姿勢を決定し，それを3次元データに重ねて表示した結果を，図32．に示す．

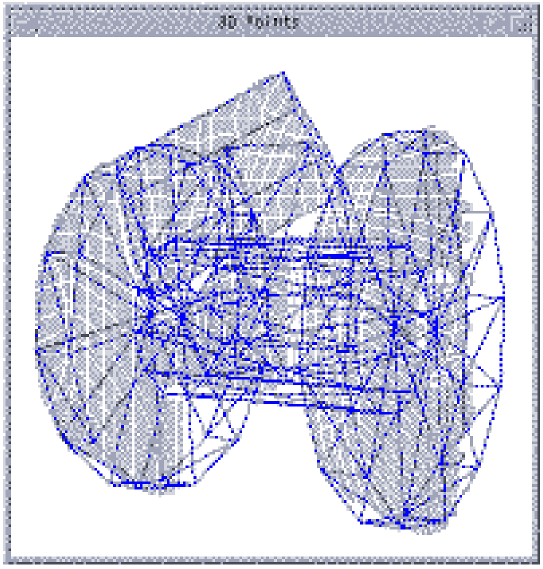
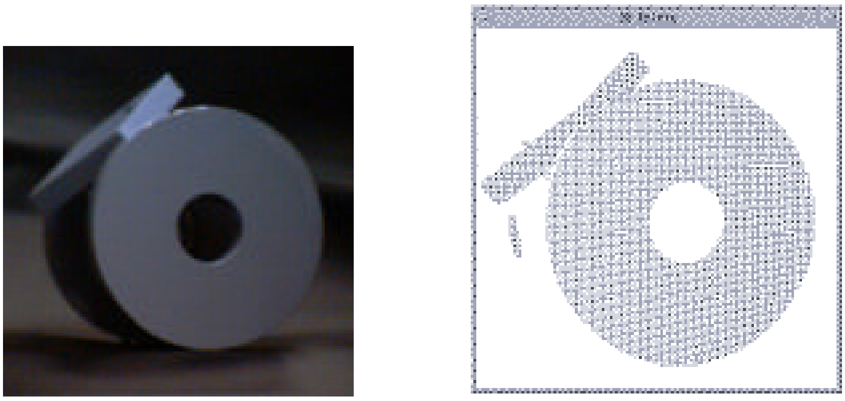
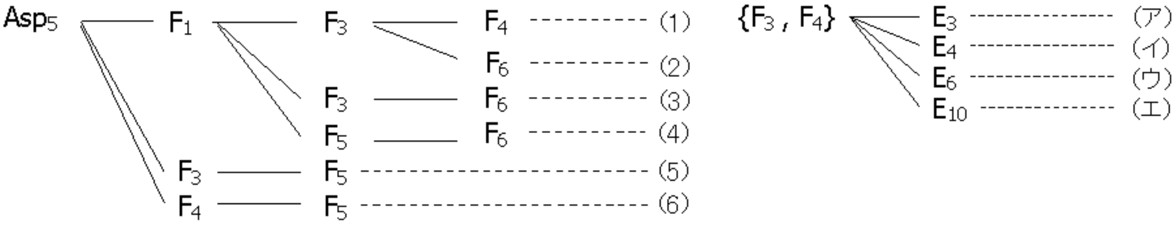


図32．　面ベースによる処理結果

5.2　エッジツリーを用いる場合



（a）カメラ画像　　　　（b）3 次元データ



（c）面ベースのツリー　　　　（d）エッジツリー

図33.　エッジツリーを用いる場合

ここでは，図 33.（a）のような状態にバルブが配置されている場合に関して位置・姿勢の教示を行う．図 33.（b）の獲得された 3 次元データを見ると，計測されている面は，図 24. においての F_3，F_4，F_5 ということになる．これによって対応する代表アスペクトは前節同様 Asp5 ということになる．ここで，Asp5 に構築されている面ベースの教示ツリーを見ると，3 次元データとして見えている面を利用して位置・姿勢が決定できる面の組み合わせが存在していないことがわかる．これは獲得された 3 次元データでは，面同士を一致させたときに物体の位置・姿勢が決定できないことを表している．このとき，オペレータは，計測されている面（F_3，F_4，F_5）を含んでいるもので，かつ優先順位の高い面の組み合わせを選ぶことになる．ここでは図 33.（c）において（1）の組み合わせがそれにあたる．次に，オペレータは，選択した面の組み合わせの中で，3 次元データに面が計測されているものをシステムに入力する，データとして計測されている面のうち，組み合わせ（1）に含まれるものは，F_3，F_4 である．入力された面を基に，システムが，4.5 によって構築されてあるエッジベースの教示ツリーの中から，それらの面を含むツリーをピックアップする．このようにして選択されたエッジベースのツリーから，オペレータが，

獲得された3次元データに，計測されたエッジ要素を判断し，ツリーにおけるエッジの組み合わせの一つを選択する．このとき，面の入力がなされた時点で，モニタ上に3次元データから導出されたエッジ要素が表示され，オペレータは計測されているエッジ要素を容易に判断できる．この処理において図33．（d）において（ア）のエッジが選択される．このような一連の対話処理によって選択された面・エッジの組み合わせ（F_3，F_4，E_3）を用いてバルブの位置・姿勢を決定する．処理結果を図34．に示す．

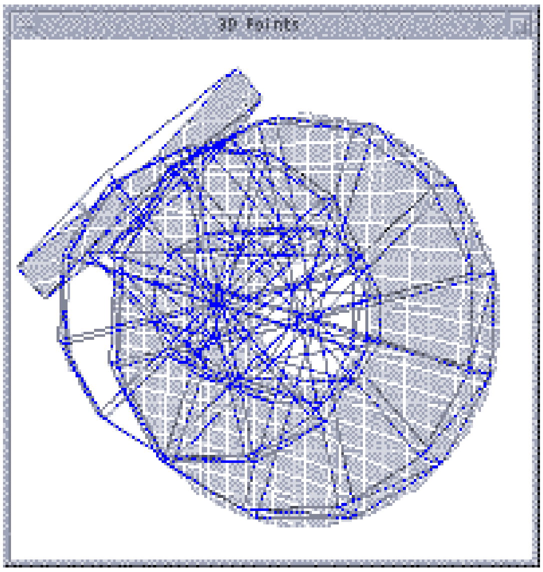
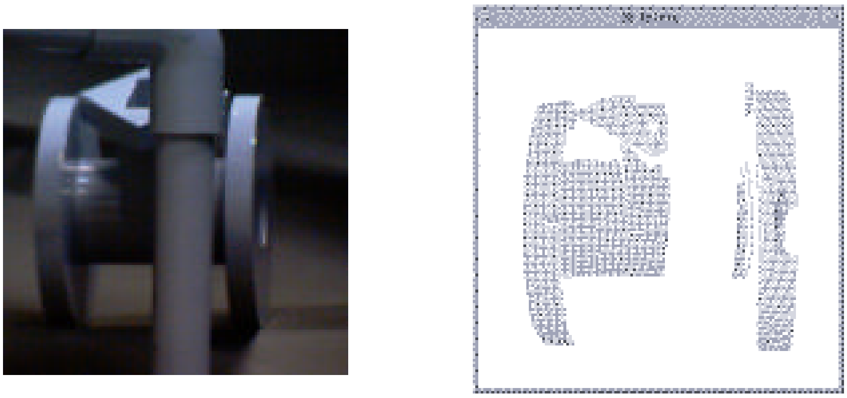


図34.　エッジを用いる処理結果

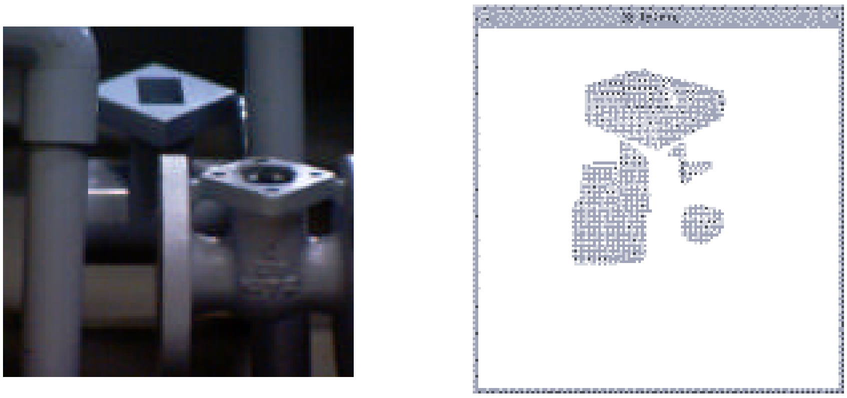
5.3　オクルージョンが起こっている場合

（i）



（a）カメラ画像　　（b）3 次元データ

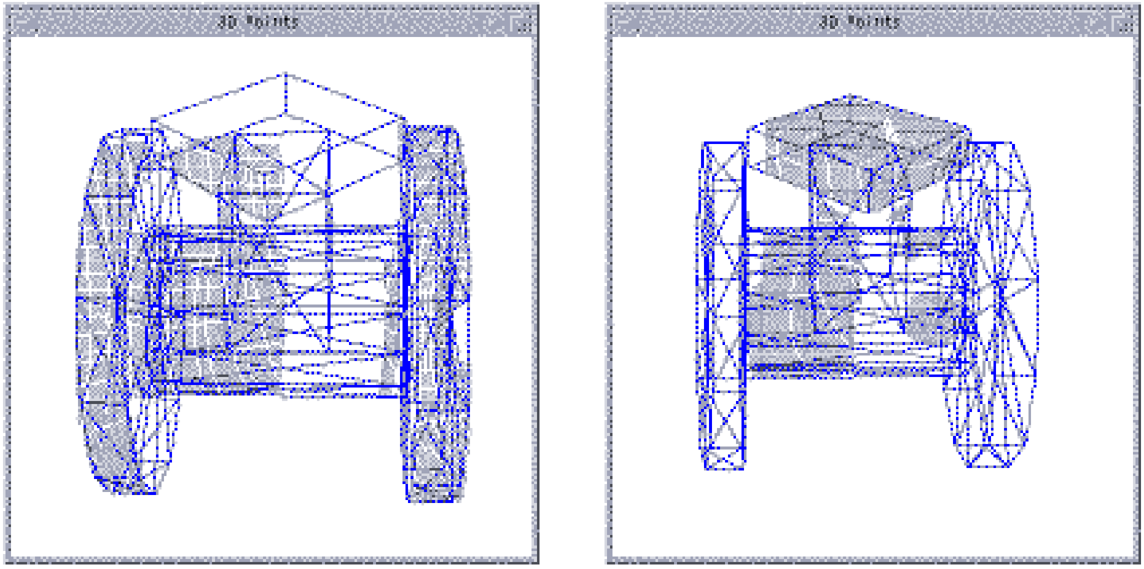
（ii）



（a）カメラ画像　　（b）3 次元データ

図35.　オクルージョンが起こっている場合

ここでは，3 次元データの獲得に望ましくない状況として物体にオクルージョンが起こっている場合について考えてみる．図 35．（i），（ii）におけるそれぞれのカメラ画像（a）では，それぞれパイプやバルブによる，求めるバルブが一部分しか見えていないことが確認できる．このような場合，レンジファンダによって獲得される 3 次元データは，それぞれ（b）のようになり，オクルージョンによってかなりの部分がデータに反映されていない．このような場合において，前節までに述べた面ベースの教示ツリー，エッジベースの教示ツリーを用いた対話処理によってバルブの位置・姿勢の教示を行った場合の結果を図 36．に示す．


3D Points
3D Points

（i）　（ii）

図36.　オクルージョンにおける処理結果

図36．の（i），（ii）はそれぞれ図35．における（i）（ii）にそれぞれ対応している．図に示されたとおり，このようなオクルージョンによって対象の3次元データが充分に獲得されていない場合でも，物体の位置・姿勢は正しく決定される．

第6章 まとめ

本研究において，レンジファインダで獲得した物体の3次元データに対して，面ベース，またエッジベースの教示ツリーを用いて，オペレータとの対話的処理により物体の位置・姿勢を教示する手法を提案した．本手法は物体の位置・姿勢を決定するために，必要かつ最小限の面の組み合わせが考慮され，ツリー構造化されている．また望ましくない状況が起こった場合に，3次元データの獲得状況によって面の組み合わせが不可能になることを考慮して，その場合の各面の組み合わせごとに，位置・姿勢を決定するために必要かつ最小限のエッジの組み合わせが，同じくツリー構造化されている．これにより，オクルージョンや鏡面反射等の3次元データ獲得に望ましくない条件下でさえも効果的に物体の位置・姿勢を決定できる．原子力プラント等での極限作業ロボットの作業環境を考えた場合，狭い範囲に多くの物体が配置されオクルージョンはかなり発生する．したがって本手法はこのようなロボットの作業環境に適しているといえる．

また，ロボット作業の一部としてセンシングを行い，物体のモデルを構築する場合，そのモデルは，タスクにおける信頼度が重要である．本研究では，ロボットタスクを考慮し，ツリーの各組み合わせは，タスクに関係した面ほど高い優先順位を持つように順序付けが施されている．そのため，ツリーの構造がタスクに関して意味をもち，タスク実行上の信頼性が高くなる．これにより，望ましくない環境と，求められる作業内容の，両方に適したモデルを計算機上に構築することができる．

さらに，本研究では，オペレータとの対話的処理により物体の位置・姿勢を教示する．極限作業ロボットはセンシングから実行に至るまですべての段階でミスが許されない．そのためオペレータの介入は避けられない現状があるが，本手法における対話的処理は，オペレータにかかる負担を少なくすることができ，また，効率的かつ正確に位置・姿勢の教示を行う点においても有効である．

本研究では，作業環境の望ましくない条件として，オクルージョンを取り上げたが，鏡面反射や，三角測量における不可計測領域に関しては，具体的な実験を行うまでには至らなかった．今後，このような他の望ましくない状況下においても例を挙げ実験を行うことが課題として残る．何か定量的な評価基準を設けて，システムの全体の信頼性や精度を高めることなども考えられる．ほかに今後の課題としては，オペレータかかる負担をさらに軽減するためにシステムの自動化を進めること，さらに構築したモデルを基に，仮想環

境などを用いて，オフラインで実行プログラムを作ることや，実際にマニピュレーション作業を行うことが考えられる．

謝辞

本研究を行うにあたり多くのご教示，熱心なご指導を賜りました東京商船大学交通電子機械工学専攻情報システム研究室の大島正毅教授に深く感謝致します．また，有益なご教示，ご意見をいただきました同研究室の全へい東助教授ならびに長谷川為春様に心より感謝いたします．

SOLVER について，使用許可をいただいた，電子技術総合研究所の越川和忠様，植芝俊夫様，新潟大学の山本正信教授のお三方に，大変お世話になり，心より感謝いたします．

さらに，本研究を進めるにあたり実験データの獲得やその他多大なご指導，ご意見をいただきました電子技術総合研究所知能システム部の中村晃様に深く感謝致します．

参考文献

[1] 大島正毅，白井良明：“3 次元情報を用いた物体認識”，信学論 D-Ⅱ Vol.J65-D No.5 pp.629-636(1982.5)

[2] 大島正毅，白井良明：“3 次元モデルを用いるシーン解釈の一手法”，第 1 回ロボット学会学術講演会　No.1112(1983.12)

[3] 大崎，山本：“ステレオ画像からの　3 次元近似モデルのフィッティング”，電子情報通信学会論文誌，vol.81-D-Ⅱ，no.6，pp.1259‐1268，1998

[4] 清水，出口：“計測誤差を考慮した距離画像からの精密な姿勢推定”，電子情報通信学会論文誌，vol.82-D-Ⅱ，no.12，pp.2298‐2306，1999

[5] A.E.Johnson，M.Hebert：“Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes”，IEEE PAMI，vol.21，no.5，pp.433-449，1999

[6] 角，富田：“ステレオビジョンによる　3 次元物体の認識”，電子情報通信学会論文誌，vol.80-D-Ⅱ，no.5，pp.1105‐1112，1997

[7] T.Hasegawa：“A Interactive Systems for Modeling and Monitoring a Manipulation Environment”，IEEE Trans. Systs.，Man，Cybern，vol.SMC-12，no.3，pp.250-258，1982

[8] T.Hasegawa，S.kameyama：“Geometric Modeling of Manipulation Environment with Interactive Teaching and Automated Accuracy Improvement”，20th Int. Symp. on Industrial Robots，pp.419-426，1989

[9] T.Hasegawa，T.Suehiro，T.Ogasawara，T.Matsui，K.Kitagaki，K.Takase：“An Integrated Tele-Robotics System with a Geometric Environment Model and Manipulation Skills”，IEEE Int. Workshop on Intelligent Robots and Systems ’90，pp.335-341，1990

[10] 中村晃，小笠原司，筑根秀男，大島正毅：“環境モデルにおける教示手順の体系化”，第 12 回日本ロボット学会学術講演会，　pp.1141-1142（1994.6）

[11] Akira Nakamura，Tsukasa Ogasawara，Hideo Tsukune，Masaki Oshima，“Surface-Based Geometric Modeling of General Objects Using Teaching Trees”：IROS’95，（1995）

[12] Akira Nakamura, Tsukasa Ogasawara, Hideo Tsukune and Masaki Oshima：“Surface-Based Geometric Modeling with an Edge Using a Teaching Tree” Proceedings of 7th International Conference on Advanced Robotics (ICAR'95), Vol.1，pp.489-494，Sant Feliu de Guixols, Spain，September 1995.

[13] Akira Nakamura, Tsukasa Ogasawara, Hideo Tsukune, Masaki Oshima："Surface-Based Geometric Modeling of Geometric Objects Using Task-Oriented Teaching Trees", IROS'96, （1995.11）

[14] Akira Nakamura, Tsukasa Ogasawara, Hideo Tsukune, Masaki Oshima："Surface-Based Geometric Modeling Using Teaching in Consideration of Backprojection", ICAR'97, pp.985-989, 1997

[15] Akira Nakamura, Kazuo Machida and Hideo Tsukune："Geometric Modeling Using Range Data and an Edge Image Derived from a Range Finder" Proceedings of the 30th International Symposium on Robotics (30th ISR), pp.233-240, Tokyo, Japan, October 1999

[16] 中村晃，小笠原司，筑根秀男，大島正毅："極限作業ロボットのための教示ツリーを用いた面ベースの環境モデリング"，日本機械学会論文集（C 編），No.99-1127，pp.143-150（2000.9）

[17] 井口，佐藤："三次元画像計測"，昭晃堂，1990

[18] 糊田："高速光切断法を用いた 3 次元形状入力システム"，計測技術，pp.38-42，1996

[19] 藤原，今井，藤井："3 次元形状計測システム VIVID におけるデータ処理"，第 34 回パターン計測部会研究会資料，pp.1-8，1996

[20] 佐藤："アクティブレンジファインダと距離画像"，日本ロボット学会誌，vol.13，no.3，pp.311-314，1995

[21] 土屋，深田："画像処理"，コロナ者，1990

[22] 越川和忠："SOLVER (A SOlid Modeler for Vision Engineering Research)"，電総研制御部視覚システム研究室（1985.4）

[23] 越川和忠："仮想ソリッドモデラ solver94 (SOlid modeler for Vision Engineering Research)"，電総研制御部視覚システム研究室，(1994.6)

[24] 土肥浩，石塚満："ソリッドモデラと幾何学的推論を組み込んだ 3D ビジョンシステム"，電子情報通信学会論文誌，D-Ⅱ，Vol.37，no.10，pp.1167-1686，1995.10

[25] 大島正毅，白井良明："物体認識のための照合用データ構造"，第 2 回日本ロボット学会学術講演会，No.2311 (1982.11)

[26] K.Ikeuchi，T.Kanade："Automatic Generation of Object Recognition Programs"，Proc. IEEE，vol.76，no8，pp.1016-1035，1988

[27] 福村，D.H.Ballad，C.M.Brown："コンピュータビジョン"，日本コンピュータ協会，1987

[28] 三浦純：“ センサ情報に基づく行動決定のための環境モデリング ”，日本ロボット学会誌， Vol.18 No.3，pp.325－330（2000.4）

[29] K.Ikeuchi，M.Robert：“ Task Oriented Vision ”，Proc. of Int. Conf. on Intelligent Robots and Systems，pp.2187-2194，1992

[30] 森英雄：“ 動物行動学から何を学ぶか ”，日本ロボット学会誌， vol.14，no.4，1992

[31] 鷲見和彦：“ 工業用視覚システムにおける認識系のモデル ”，日本ロボット学会誌， Vol.18，No3，pp.343-348，2000

[32] “ 作業の教示とプログラミング ”，日本ロボット学会誌， Vol.17，No2，1999

[33] K.Higuchi，M.Hebert，K.Ikeuchi：“ Building 3D models from Unregistred range image ”，Graphic modeling and Image Processing，vol.57，no.4，pp.315-333，July 1995

[34] 池内克史，末廣尚士：“ 視覚による組立作業理解のための作業モデルとそれに基づく動作生成 ”，日本ロボット学会誌， Vol.11，No.2，pp.281-290，1993

[35] 三浦純，池内克史：“ 作業の目的を考慮した視覚認識戦略の生成 ”，日本ロボット学会誌， Vol.14，No.4，pp.574-585，1996

[36] K.Weber，S.Venkatesh，M.V.Srinivasan：“ Insect Inspired Behaviours for the Autonomous Control of Mobile Robots ”，in From Living Eyes to Seeing Machines. M.V. Srinivasan and S.Venkatesh (eds.)，Oxford Univ. Press，pp.226-248，1997

[37] 松本，稲葉，井上：“ 視野画像列を利用した経路決定の基づくナビゲーション ”，日本ロボット学会誌， vol.15，no.2，pp.236-248，1997

[38] 前田，久野，白井：“ 固有空間解析に基づく移動ロボットの位置認識 ”，電子情報通信学会論文誌， vol.80-D-Ⅱ，no.6，pp.1530－1538，1997

[39] D.Koller，Q.-T.Luong，J.Malik：“ Using Binocular Stereopsis for Lane Following and Lane Changing ”，Proc. IEEE Symp. on Intelligent Vehicles，1994

[40] K.Onoguchi，N.Takeda，M.Watanabe：“ Planar Projection Stereopsis Method for Road Extraction ”，Proc. 1995 Int. Conf. on Intelligent Robots and Systems，vol.1，pp.249-256，1995

[41] B.J.Kuipers et al.：“ A Robust Qualitative Method for Robot Spatial Learning ”，Proc. AAAI-88，pp.774-779，1988

[42] 宮下，石黒，辻：“ T-net：実環境における正確なロボットの誘導と環境構造の獲得 ”，日本ロボット学会誌， vol.14，no.7，pp.986-993，1996

[43] L.Robert，M.Buffa，M.Hebert：“ Weakly-Calibrated Stereo Perception for Rover Navigation ”，Proc. 1994 DARPA Image Understanding Workshop，pp.1317-1324，1994

[44] Z.Zhang et al. ：“ A Robust Technique for Matching Two Uncalibrated Images though the Recovery of the Unknown Epipolar Geometry ”， Artifical Intelligence， vol.78， pp.87-119， 1995

[45] J.Miura， Y.Shirai ：“ Modeling Obstacles and Free Spaces for Mobile Robot using Stereo Vision with Uncertainty ”， Proc. 1994 IEEE Int. Conf. on Robotics and Automation， pp.3368-3373， 1994

[46] 滝沢，白井，三浦：“ 注視・ズームを用いた自律移動ロボットのための 3D シーン記述の選択的精密化 ”，日本ロボット学会誌， vol.13，no.7，pp.963-970，1994

[47] O.D.Faugeras，E.Le Bras-Mehlman，J.D.Boissonnat ：“ Representing Stereo Data with the Delaunay Triangulation ”， Artifical Intelligence， vol.44， pp.41-87， 1990

[48] 登尾，浪花，有本：“ クワッドツリーを利用した移動ロボットの高速経路生成アルゴリズム ”，日本ロボット学会誌， vol.7，no.5，pp.403-412，1989

[49] J.S.Zelek ：“ Performing Concurrent Robot Path Execution and Computation in Real-Time ”， Proc. 1996 AAAI Fall Symp. on Flexible Computation in Intelligent Systems， pp.160-167， 1996

[50] 三浦，白井：“ プランニングコストと視覚情報の不確かさを考慮した移動ロボットの視覚と行動のプランニング ”，人工知能学会誌， vo.13，no.4，pp.588-596，1998

[51] 浅田，野田，細田：“ ロボットの行動獲得のための状態空間の自立的構成 ”，日本ロボット学会誌， vol.15，no.6，pp.886-892，1997

[52] H.Ishiguro，R.Sato，T.Ishida ：“ Robot Oriented State Space Construction ”，Proc. 1996 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems， pp.1496-1501， 1996

[53] A.Elfes ：“ Sonar-Based Real-World Mapping and Navigation ”， IEEE J. of Robotics and Automation， vol.3， no.3， pp.249-265， 1987

[54] N.Ayache，O.D.Faugeras ：“ Maintaining Representations of the Environment of a Mobile Robot ”， IEEE Trans. on Robotics and Automation， vol.5， no.6， pp.804-819， 1989

[55] 三浦，白井：“ ステレオ視におけるあいまいな対応付けのモデリングとあいまいさ解消のための視点選択 ”，日本ロボット学会誌， vol.12，no.8，pp.1222-1230，1994

[56] S.Thurn et al. ：“ MINERVA：A Second Generation Mobile Tour-Guide Robot ”， Proc. 1999 Int. Conf. on Robotics and Automation， pp.639-643， 1991

[57] S.Atia，G.Hager ：“ Real-Time Vision-Based Robot Localization ”，Proc. 1991 IEEE Int. Robotics and Automation， pp.639-643， 1991

[58] I.D.Horswill：“ Polly: A Vision-Based Artifical Agent ”，Proc. AAAI-93，pp.824-829，1993

[59] 池内，M.Hebert：“ 複数視点レンジデータからの 3 次元物体モデルの構築 ”，電子情報通信学会論文誌，vol.79-D-Ⅱ，no.8，pp.1354 - 1361，1996

[60] 佐藤，村瀬：“ 距離画像を用いた高速多面体近似 ”，情報処理学会，vol.36，no.10，2303-2309，1995

[61] 奥村，本谷，出口：“ 対象の一部の距離画像からの一般化円筒表現の生成 ”，情報処理学会コンピュータビジョン研究会，pp.73-80，110-10，1998

[62] 諸岡，査，長谷川：“ 複数の距離画像の統合による 3 次元物体モデル生成のための視点計画 ”，電子情報通信学会論文誌，vol.82-D-Ⅱ，no.3，pp.371 - 381，1999

[63] 大原，山本：“ 光切断法と投影法の同時測定による 3 次元形状のボクセルモデル入力法 ”，電子情報通信学会論文誌，vol.82-D-Ⅱ，no.4，pp.743 - 750，1999

[64] 古川，清水，佐々木：“ 距離画像の機構像を用いた形状記述データ生成 ”，電子情報通信学会論文誌，vol.81-D-Ⅱ，no.9，pp.2130 - 2138，1998

[65] 原，査，長谷川：“ 最適ポリゴン近似と連続変形法による 3 次元モデル生成法 ”，電子情報通信学会論文誌，vol.82-D-Ⅱ，no.3，pp.458 - 467，1999

[66] 池内：“ 画像による実物体モデルの生成 ”，日本ロボット学会誌，vol.16，no.6，pp.763-766，1998

[67] 杉本，富田：“ ステレオによる幾何モデリング ”，日本ロボット学会誌，vol.15，no.3，pp.431-483，1997

[68] X.Jiang，H.Bunke：“ Edge Detection in Range Images Based on Scan Line Approximation ”，Computer Vision and Image Understanding，vol.73，no.2，pp.183-199，1999

付録１ DXF フォーマットのファイル

```
0
SECTION          ヘッダ
2
ENTITIES
0
3DFACE           宣言部
8
0
10
45
20
0
30                              データ記述部
76
11
0
21
58
31
76                 データ部
12
-45
22
0
32
76
13
0
23
-58
33
76
0
・
・
・
```

```
・
・
・
0
ENDSEC
0
EOF
```

フッタ

付録 2 プログラム

```
#include <stdio.h>
#include <stdlib.h>
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <sys/types.h>
#include <math.h>

#define   center     100
#define   pai          3.141592
#define   N             40000
#define   K             10000

static     Window      root,win0,win1;
static     Display    *disp;
static     Visual      *vis;
static     Colormap cmap;
static     GC             gc;
static     XSetWindowAttributes atr;
static     XImage      *image;
static     roop;
static     XColor      c1,cc;
static     FILE        *fp1,*fp2;

static     int          hdp,hd[N];
static     int          r_max,sh_max;
static     int          label_num[N],lb_max;
static     double    dx[N],dy[N];
static     double    x_av,y_av,z_av;

void         win_set();
void         Get_Range();
void         Get_Shape();
void         Range_Disp();
void         Shape_Disp();
void         Shape_Normal();
void         Get_teaching_Face();
void        Range_Normal();
void        Range_Labeling();
void        Get_teaching_Face();
void        Rotate_S2R();
void        Inv_Matrix(double m[][3],double inv_m[][3]);
void        Prod_Matrix(double a[][3],double b[][3],double prod[][3]);
void        Rotate(int i,double R[][3]);
void        Point_set(double,double,double,double *,double *);
void        Point_set2(double,double,double,double *,double *);
void        Points_Display(int,double,double);
void        Range_Label_Disp();
void        Line_Display(double,double,double,double);

struct pdata{
   double x,y,z;
};

struct {
   struct pdata p;
   struct pdata n;
   int       lb;
}p[N];

struct fdata{
   struct pdata p1;
   struct pdata p2;
   struct pdata p3;
   struct pdata p4;
   struct pdata n;
   int       teach_num;
}face[K];

main()
{
      int i;

      Get_Range();

      Get_Shape();
```

```
        Shape_Normal();

        Get_teaching_Face();

        for(i=0;i<=r_max;++i){

            p[i].n.x = 0.0;
            p[i].n.y = 0.0;
            p[i].n.z = 0.0;

            p[i].lb = 0;
            label_num[i] = 0;

        }

        Range_Normal();

        Range_Labeling();

        Rotate_S2R();

        win_set();

        Range_Disp();
        /*
        Range_Label_Disp();
        */
        Shape_Disp();

        getchar();

}

void win_set(void)
{
        static char names0[]={"3D Points"};
        static char names1[]={"Regions"};
        static char *name_return0;
        int i;
        disp = XOpenDisplay(NULL);
        root = RootWindow(disp,0);
        vis = DefaultVisual(disp,0);
        cmap = DefaultColormap(disp,0);
        gc = XCreateGC(disp,root,0,0);
        atr.backing_store = Always;

        XAllocNamedColor(disp,cmap,"white",&c1,&cc);

        win0 = XCreateSimpleWindow(disp,root,100,100,400,400,2,0,c1.pixel);
        XChangeWindowAttributes(disp,win0,CWBackingStore,&atr);
        XStoreName(disp,win0,names0);
        XFetchName(disp,win0,&name_return0);
        XMapWindow(disp,win0);

}


void Get_Range(void)
{

        int i,n;
        double ax,ay,az;
        char s[256],fname[50];

        i=0; n=0;
        ax=0; ay=0; az=0;

  target1:

        printf("range_data filname = ?¥n");
        scanf("%s",fname);
        fp1=fopen(fname,"r");

        if(fp1==NULL){

            printf("¥n404 File not Found!¥n¥n");
            goto target1;

        }
```

```
	getchar();

	/*	レンジデータを p[]に取り込む	*/
	while(fgets(s,256,fp1)!=NULL){

	   if( s[0]=='#' || s[0]=='w' || s[0]=='h')continue;
	   if(s[0]=='*' && s[1]=='*')continue;

	   sscanf(s,"%lf %lf %lf",&p[i].p.x,&p[i].p.y,&p[i].p.z);

	   if(p[i].p.z!=0.0){

	      hd[hdp] = i;
	      hdp = ++hdp;

	      n = ++n;

	      ax = ax+p[i].p.x;
	      ay = ay+p[i].p.y;
	      az = az+p[i].p.z;

	   }

	   r_max = i;
	   i = ++i;

	}

	/*	レンジデータの (x,y,z)の平均	*/
	x_av = ax/n;
	y_av = ay/n;
	z_av = az/n;
	/*
	printf(" x_av. = %lf¥n y_av. = %lf¥n z_av. = %lf¥n",x_av,y_av,z_av);
	*/
}

void Get_Shape(void)
{
    int i,j,imax;
    char s[K][256],fname[50];

    i=0; j=0; imax=0;

target2:

    printf("cad_data filname = ?¥n");
    scanf("%s",fname);
    fp2=fopen(fname,"r");

    if(fp2==NULL){

        printf("¥n404 File not Found!¥n¥n");
        goto target2;

    }

    getchar();

    /*  CAD データファイルからを文字列  s[][]に取り込む    */
    while(fgets(s[i],256,fp2)!=NULL){
        /*
        printf("%s",s[i]);
        */
        imax = i;
        i = ++i;

    }

    /*  文字列 s[][]から CAD データを face[]に取り込む    */
    for(i=0;i<=imax;++i){

        if(s[i][0]=='3' && s[i][1]=='D'){

            i = i+4;
            sscanf(s[i],"%lf",&face[j].p1.x);

            i = i+2;
            sscanf(s[i],"%lf",&face[j].p1.z);
    i = i+2;
    sscanf(s[i],"%lf",&face[j].p1.y);

    i = i+2;
    sscanf(s[i],"%lf",&face[j].p2.x);

    i = i+2;
    sscanf(s[i],"%lf",&face[j].p2.z);

    i = i+2;
    sscanf(s[i],"%lf",&face[j].p2.y);

    i = i+2;
    sscanf(s[i],"%lf",&face[j].p3.x);

    i = i+2;
    sscanf(s[i],"%lf",&face[j].p3.z);

    i = i+2;
    sscanf(s[i],"%lf",&face[j].p3.y);

    i = i+2;
    sscanf(s[i],"%lf",&face[j].p4.x);

    i = i+2;
    sscanf(s[i],"%lf",&face[j].p4.z);

    i = i+2;
    sscanf(s[i],"%lf",&face[j].p4.y);
    /*
    printf("p1 = {%lf    %lf    %lf}¥n",face[j].p1.x,face[j].p1.y,face[j].p1.z);
    printf("p2 = {%lf    %lf    %lf}¥n",face[j].p2.x,face[j].p2.y,face[j].p2.z);
    printf("p3 = {%lf    %lf    %lf}¥n",face[j].p3.x,face[j].p3.y,face[j].p3.z);
    printf("p4 = {%lf    %lf    %lf}¥n¥n",face[j].p4.x,face[j].p4.y,face[j].p4.z);
    */
    sh_max = j;
    j = ++j;

}
```

```
        }

}


void Shape_Normal(void)
{
        int i;
        double nx,ny,nz,sp;
        double r1x,r1y,r1z,r2x,r2y,r2z;

        i=0;

        for(i=0;i<=sh_max;++i){

            r1x = face[i].p2.x-face[i].p1.x;
            r1y = face[i].p2.y-face[i].p1.y;
            r1z = face[i].p2.z-face[i].p1.z;

            r2x = face[i].p3.x-face[i].p1.x;
            r2y = face[i].p3.y-face[i].p1.y;
            r2z = face[i].p3.z-face[i].p1.z;

            nx = r1y*r2z - r1z*r2y;
            ny = r1z*r2x - r1x*r2z;
            nz = r1x*r2y - r1y*r2x;

            sp = sqrt(nx*nx+ny*ny+nz*nz);

            face[i].n.x = -(nx/sp);
            face[i].n.y = -(ny/sp);
            face[i].n.z = -(nz/sp);

            if(face[i].n.x == -0.0) face[i].n.x = 0.0;
            if(face[i].n.y == -0.0) face[i].n.y = 0.0;
            if(face[i].n.z == -0.0) face[i].n.z = 0.0;

            /*
            printf("normal x = %lf¥t normal y = %lf normal z = %lf¥n",
                        face[i].normal.x,face[i].normal.y,face[i].normal.z);
            */
        }

}


void Get_teaching_Face(void)
{

        int i;

        for(i=0;i<=sh_max;++i)
            face[i].teach_num = 0;

        i = 0;
        face[i].teach_num = 1;

        i = 1;
        face[i].teach_num = 3;

        i = 2;
        face[i].teach_num = 2;

        i = 4;
        face[i].teach_num = 4;

        i = 54;
        while(i<100){

            face[i].teach_num = 5;

            i = ++i;
            face[i].teach_num = 5;

            i = i+3;

        }

        i = 102;
        while(i<148){
            face[i].teach_num = 5;

            i = ++i;
            face[i].teach_num = 5;

            i = i+3;

        }

        i = 56;
        while(i<=100){

            face[i].teach_num = 6;

            i = i+4;
            face[i].teach_num = 6;

        }


}



void Range_Normal()
{
        int i,j;
        double u1x,u1y,u1z,u2x,u2y,u2z,u3x,u3y,u3z,u4x,u4y,u4z;
        double v1x,v1y,v1z,v2x,v2y,v2z;
        double sp,hx,hy,hz;

        for(i=0;i<=hdp;i=i+1){

            j = hd[i];

            u1x = p[j-200].p.x-p[j].p.x;
            u1y = p[j-200].p.y-p[j].p.y;
            u1z = p[j-200].p.z-p[j].p.z;

            u2x = p[j-1].p.x-p[j].p.x;
            u2y = p[j-1].p.y-p[j].p.y;
            u2z = p[j-1].p.z-p[j].p.z;

            u3x = p[j+1].p.x-p[j].p.x;
            u3y = p[j+1].p.y-p[j].p.y;
            u3z = p[j+1].p.z-p[j].p.z;

            u4x = p[j+200].p.x-p[j].p.x;
            u4y = p[j+200].p.y-p[j].p.y;
            u4z = p[j+200].p.z-p[j].p.z;

            v1x = u1y*u2z-u1z*u2y;
            v1y = u1z*u2x-u1x*u2z;
            v1z = u1x*u2y-u1y*u2x;

            v2x = -(u3y*u4z-u3z*u4y);
            v2y = -(u3z*u4x-u3x*u4z);
            v2z = -(u3x*u4y-u3y*u4x);

            hx = (v1x+v2x)/2.0;
            hy = (v1y+v2y)/2.0;
            hz = (v1z+v2z)/2.0;

            sp = sqrt(hx*hx+hy*hy+hz*hz);

            p[j].n.x = hx/sp;
            p[j].n.y = hy/sp;
            p[j].n.z = hz/sp;

        }

}



void Range_Labeling()
{
        int i,n;
        int u_kin,d_kin,l_kin,r_kin;
        int label,seed,stack[N],st_check,st_num;
        int lb_table[N],lb_min;
```

```
double nx_sub,ny_sub,nz_sub,border=0.6;

label = 0;
lb_min = 0;

for(i=0;i<=hdp;++i)
   lb_table[i]=0;

for(i=0;i<=hdp;++i){

   n = hd[i];

   if(p[n].lb == 0){

      label = ++label;
      p[n].lb = label;
      label_num[label] = label_num[label]+1;
      seed = n;

      st_check = 0;
      st_num = 1;
      stack[st_check]= n;

      while(st_check<st_num){



         n = stack[st_check];
         u_kin    = n-200;
         l_kin    = n-1;
         r_kin    = n+1;
         d_kin    = n+200;



         if(p[n].lb==label){

            if((p[u_kin].p.z)!=0.0 && p[u_kin].lb==0){

               nx_sub = p[u_kin].n.x-p[seed].n.x;
               ny_sub = p[u_kin].n.y-p[seed].n.y;
               nz_sub = p[u_kin].n.z-p[seed].n.z;
   if(nx_sub<0.0) nx_sub = -nx_sub;
   if(ny_sub<0.0) ny_sub = -ny_sub;
   if(nz_sub<0.0) nz_sub = -nz_sub;

   if(nx_sub<border && ny_sub<border && nz_sub<border){

    p[u_kin].lb = label;
    label_num[label] = label_num[label]+1;

    stack[st_num]=u_kin;
    st_num = st_num+1;

   }
}

if((p[l_kin].p.z)!=0.0 && p[l_kin].lb==0){

   nx_sub = p[l_kin].n.x-p[seed].n.x;
   ny_sub = p[l_kin].n.y-p[seed].n.y;
   nz_sub = p[l_kin].n.z-p[seed].n.z;

   if(nx_sub<0.0) nx_sub = -nx_sub;
   if(ny_sub<0.0) ny_sub = -ny_sub;
   if(nz_sub<0.0) nz_sub = -nz_sub;

   if(nx_sub<border && ny_sub<border && nz_sub<border){

    p[l_kin].lb = label;
    label_num[label] = ++label_num[label];

    stack[st_num]=l_kin;
    st_num = st_num+1;

   }
}

if((p[r_kin].p.z)!=0.0 && p[r_kin].lb==0){

   nx_sub = p[r_kin].n.x-p[seed].n.x;
```

```
        ny_sub = p[r_kin].n.y-p[seed].n.y;
        nz_sub = p[r_kin].n.z-p[seed].n.z;

        if(nx_sub<0.0) nx_sub = -nx_sub;
        if(ny_sub<0.0) ny_sub = -ny_sub;
        if(nz_sub<0.0) nz_sub = -nz_sub;

        if(nx_sub<border && ny_sub<border && nz_sub<border){

         p[r_kin].lb = label;
         label_num[label] = ++label_num[label];

         stack[st_num]=r_kin;
         st_num = st_num+1;

        }
    }

    if((p[d_kin].p.z)!=0.0 && p[d_kin].lb==0){

        nx_sub = p[d_kin].n.x-p[seed].n.x;
        ny_sub = p[d_kin].n.y-p[seed].n.y;
        nz_sub = p[d_kin].n.z-p[seed].n.z;

        if(nx_sub<0.0) nx_sub = -nx_sub;
        if(ny_sub<0.0) ny_sub = -ny_sub;
        if(nz_sub<0.0) nz_sub = -nz_sub;

        if(nx_sub<border && ny_sub<border && nz_sub<border){

         p[d_kin].lb = label;
         label_num[label] = ++label_num[label];

         stack[st_num]=d_kin;
         st_num = st_num+1;

        }
    }

}
```

```
          st_check = ++st_check;

        }

      }
    }

    for(i=0;i<=hdp;++i){

      n = hd[i];

      if(label_num[p[n].lb]<50){
        p[n].lb=0;
        continue;
      }

      else if(lb_table[p[n].lb]==0){

        lb_min = ++lb_min;
        lb_table[p[n].lb] = lb_min;

      }
    }

    for(i=0;i<=hdp;++i){

      n = hd[i];

      if(lb_table[p[n].lb]!=0)

        p[n].lb = lb_table[p[n].lb];

    }

    lb_max = lb_min;

    printf("Region = %d¥n",lb_min);

}
```

```
void Rotate_S2R(void)
{
        int i,n,x,y,z;
        double ac_x,ac_y,ac_z,count,sp;
        double x1,y1,z1,x2,y2,z2,u1,v1,w1,u2,v2,w2;
        double a1,a2,a3,a4,b1,b2,b3,b4,c1,c2;
        double al0,be0,ga0,de0,al,be,ga,de;
        double A[3][3],B[3][3],C[3][3];
        double invA[3][3],R0[3][3],R[3][3],P[3];

        x1 = face[0].n.x;
        y1 = face[0].n.y;
        z1 = face[0].n.z;

        x2 = face[1].n.x;
        y2 = face[1].n.y;
        z2 = face[1].n.z;

        ac_x = 0; ac_y = 0; ac_z = 0; count = 0;

        for(i=0;i<=hdp;++i){

            n = hd[i];

            if(p[n].lb == 1){

                ac_x = ac_x + p[n].n.x;
                ac_y = ac_y + p[n].n.y;
                ac_z = ac_z + p[n].n.z;
                count = ++count;

            }
        }

        u1 = ac_x/count;
        v1 = ac_y/count;
        w1 = ac_z/count;
```

```
sp = sqrt(u1*u1+v1*v1+w1*w1);

u1 = u1/sp;
v1 = v1/sp;
w1 = w1/sp;

ac_x = 0.0; ac_y = 0.0; ac_z = 0; count = 0.0;

for(i=0;i<=hdp;++i){

   n = hd[i];

   if(p[n].lb == 7){

      ac_x = ac_x + p[n].n.x;
      ac_y = ac_y + p[n].n.y;
      ac_z = ac_z + p[n].n.z;
      count = ++count;

   }
}

u2 = ac_x/count;
v2 = ac_y/count;
w2 = ac_z/count;

sp = sqrt(u2*u2+v2*v2+w2*w2);

u2 = u2/sp;
v2 = v2/sp;
w2 = w2/sp;

a1 = w1/sqrt((double)(1.0-v1*v1));
a2 = u1/sqrt(1.0-v1*v1);
a3 = sqrt(1.0-v1*v1);
a4 = v1;

A[0][0] = a1;
A[0][1] = -(a2*a4);
A[0][2] = a2*a3;
```

```
A[1][0] = 0.0;
A[1][1] = a3;
A[1][2] = a4;
A[2][0] = -a2;
A[2][1] = -(a1*a4);
A[2][2] = a1*a3;

printf("(u1,v1,w1)*A =¥n");
printf("%lf¥t",u1*A[0][0]+v1*A[1][0]+w1*A[2][0]);
printf("%lf¥t",u1*A[0][1]+v1*A[1][1]+w1*A[2][1]);
printf("%lf¥n",u1*A[0][2]+v1*A[1][2]+w1*A[2][2]);

if(z1==0.0 || y1*y1==0.0)
   b1 = 1.0;
else
   b1 = z1/sqrt(1-y1*y1);

if(x1==0.0 || y1*y1==0.0)
   b2 = 0.0;
else
   b2 = x1/sqrt(1-y1*y1);

if(y1*y1==0.0)
   b3 = 1.0;
else
   b3 = sqrt(1-y1*y1);

b4 = y1;

B[0][0] = b1;
B[0][1] = -(b2*b4);
B[0][2] = b2*b3;
B[1][0] = 0.0;
B[1][1] = b3;
B[1][2] = b4;
B[2][0] = -b2;
B[2][1] = -(b1*b4);
B[2][2] = b1*b3;

printf("(x1,y1,z1)*B =¥n");
printf("%lf¥t",x1*B[0][0]+y1*B[1][0]+z1*B[2][0]);
printf("%lf¥t",x1*B[0][1]+y1*B[1][1]+z1*B[2][1]);
printf("%lf¥n",x1*B[0][2]+y1*B[1][2]+z1*B[2][2]);

al0 = x2*b1-z2*b2;
be0 = -x2*b2*b4+y2*b3-z2*b1*b4;
ga0 = u2*a1-w2*a2;
de0 = -u2*a2*a4+v2*a3-w2*a1*a4;

if(al0==0.0 || (al0*al0+be0*be0)==0.0)
   al = 0.0;
else
   al = al0/sqrt(al0*al0+be0*be0);

if(be0==0.0 || (al0*al0+be0*be0)==0.0)
   be = 0.0;
else
   be = be0/sqrt(al0*al0+be0*be0);

if(ga0==0.0 || (ga0*ga0+de0*de0)==0.0)
   ga = 0.0;
else
   ga = ga0/sqrt(ga0*ga0+de0*de0);

if(de0==0.0 || (ga0*ga0+de0*de0)==0.0)
   de = 0.0;
else
   de = de0/sqrt(ga0*ga0+de0*de0);



c1 = al*ga+be*de;
c2 = -al*de+be*ga;

C[0][0] = c1;
C[0][1] = -c2;
C[0][2] = 0.0;
C[1][0] = c2;
C[1][1] = c1;
C[1][2] = 0.0;
```

```
        C[2][0] = 0.0;
        C[2][1] = 0.0;
        C[2][2] = 1.0;

        Inv_Matrix(A,invA);

        Prod_Matrix(B,C,R0);

        Prod_Matrix(R0,invA,R);

        for(i=0;i<=sh_max;++i){

            Rotate(i,R);

        }
}


void Inv_Matrix(double m[][3],double inv_m[][3])
{
        double a,b,c,p,q,r,x,y,z;
        double det;

        a = m[0][0];
        b = m[0][1];
        c = m[0][2];
        p = m[1][0];
        q = m[1][1];
        r = m[1][2];
        x = m[2][0];
        y = m[2][1];
        z = m[2][2];

        det = 1/(a*q*z-a*r*y-p*b*z+p*c*y+x*b*r-x*c*q);

        inv_m[0][0] = det*(q*z-r*y);
        inv_m[0][1] = det*(c*y-b*z);
        inv_m[0][2] = det*(b*r-c*q);
        inv_m[1][0] = det*(r*x-p*z);
        inv_m[1][1] = det*(a*z-c*x);
        inv_m[1][2] = det*(c*p-a*r);
        inv_m[2][0] = det*(p*y-q*x);
        inv_m[2][1] = det*(b*x-a*y);
        inv_m[2][2] = det*(a*q-b*p);

}

void Prod_Matrix(double a[][3],double b[][3],double prod[][3])
{
        int i,j,k;

        for(i = 0; i<3; i++){
                for(j = 0; j<3; j++){
                        prod[i][j] = 0.0;
                        for(k = 0; k<3; k++){
                                prod[i][j] = prod[i][j] + a[i][k] * b[k][j];
                        }
                }
        }
}

void Rotate(int n,double R[][3])
{
        double a[3];

        a[0] = R[0][0]*face[n].p1.x+R[1][0]*face[n].p1.y+R[2][0]*face[n].p1.z;
        a[1] = R[0][1]*face[n].p1.x+R[1][1]*face[n].p1.y+R[2][1]*face[n].p1.z;
        a[2] = R[0][2]*face[n].p1.x+R[1][1]*face[n].p1.y+R[2][2]*face[n].p1.z;

        face[n].p1.x = a[0];
        face[n].p1.y = a[1];
        face[n].p1.z = a[2];

        a[0] = R[0][0]*face[n].p2.x+R[1][0]*face[n].p2.y+R[2][0]*face[n].p2.z;
        a[1] = R[0][1]*face[n].p2.x+R[1][1]*face[n].p2.y+R[2][1]*face[n].p2.z;
        a[2] = R[0][2]*face[n].p2.x+R[1][1]*face[n].p2.y+R[2][2]*face[n].p2.z;

        face[n].p2.x = a[0];
        face[n].p2.y = a[1];
        face[n].p2.z = a[2];
```

```
	a[0] = R[0][0]*face[n].p3.x+R[1][0]*face[n].p3.y+R[2][0]*face[n].p3.z;
	a[1] = R[0][1]*face[n].p3.x+R[1][1]*face[n].p3.y+R[2][1]*face[n].p3.z;
	a[2] = R[0][2]*face[n].p3.x+R[1][1]*face[n].p3.y+R[2][2]*face[n].p3.z;

	face[n].p3.x = a[0];
	face[n].p3.y = a[1];
	face[n].p3.z = a[2];

	a[0] = R[0][0]*face[n].p4.x+R[1][0]*face[n].p4.y+R[2][0]*face[n].p4.z;
	a[1] = R[0][1]*face[n].p4.x+R[1][1]*face[n].p4.y+R[2][1]*face[n].p4.z;
	a[2] = R[0][2]*face[n].p4.x+R[1][1]*face[n].p4.y+R[2][2]*face[n].p4.z;

	face[n].p4.x = a[0];
	face[n].p4.y = a[1];
	face[n].p4.z = a[2];
	/*
	printf("i = %d¥n",n);

	printf("p1.x = %lf¥tp1.y = %lf¥tp1.z = %lf¥n",
			face[n].p1.x,face[n].p1.y,face[n].p1.x);

	printf("p2.x = %lf¥tp2.y = %lf¥tp2.z = %lf¥n",
			face[n].p2.x,face[n].p2.y,face[n].p2.x);

	printf("p3.x = %lf¥tp3.y = %lf¥tp3.z = %lf¥n",
			face[n].p3.x,face[n].p3.y,face[n].p3.x);

	printf("p4.x = %lf¥tp4.y = %lf¥tp4.z = %lf¥n¥n",
			face[n].p4.x,face[n].p4.y,face[n].p4.x);
			*/

}


void Range_Disp(void)
{
	int i,j;
	double px,py;
	for(i=0;i<=(hdp-1);++i){

	   j = hd[i];

	   p[j].p.x = p[j].p.x-x_av;
	   p[j].p.y = p[j].p.y-y_av;
	   p[j].p.z = p[j].p.z-z_av;

	   Point_set(p[j].p.x,p[j].p.y,p[j].p.z,&px,&py);

	   Points_Display(j,px,py);

	}

}


void Shape_Disp(void)
{
	int i;
	double px1,py1,px2,py2,px3,py3,px4,py4;

	for(i=0;i<=sh_max;++i){

	   Point_set2(face[i].p1.x,face[i].p1.y,face[i].p1.z,&px1,&py1);
	   Point_set2(face[i].p2.x,face[i].p2.y,face[i].p2.z,&px2,&py2);
	   Point_set2(face[i].p3.x,face[i].p3.y,face[i].p3.z,&px3,&py3);
	   Point_set2(face[i].p4.x,face[i].p4.y,face[i].p4.z,&px4,&py4);

	   Line_Display(px1,py1,px2,py2);
	   Line_Display(px2,py2,px3,py3);
	   Line_Display(px3,py3,px4,py4);
	   Line_Display(px4,py4,px1,py1);

	}

}

void Point_set(double x,double y,double z,double *px,double *py)
/*	三次元の　距離データを二次元に投影する	*/
```

```
{
        double ax = 0.0*pai/180,    /* x 軸回りの回転角    */
                ay = 0.0*pai/180,    /* y 軸回りの回転角    */
                az = 0.0*pai/180,    /* z 軸周りの回転角    */
                vp = -300.0,          /*  投影中心           */
                l = 20.0,             /* x 方向の移動量       */
                m = -25.0,            /* y 方向の移動量       */
                n = -100.0,           /* z 方向の移動量       */
                x1,y1,z1,x2,y2,z2,h;




        x1 = x;
        y1 = y*cos(ax)-z*sin(ax);
        z1 = y*sin(ax)+z*cos(ax);

        x2 = x1*cos(az)-y1*sin(az);
        y2 = x1*sin(az)+y1*cos(az);
        z2 = z1;

        h = -x2*sin(ay)/vp+z2* cos(ay)/vp+n/vp+1;

        *px = (x2*cos(ay)+z2*sin(ay)+l)/h;
        *py = -((y2+m)/h);

}


void Point_set2(double x,double y,double z,double *px,double *py)
/*        三次元の   距離データを二次元に投影する          */
{
        double ax = 0.0*pai/180,    /* x 軸回りの回転角    */
                ay = 0.0*pai/180,    /* y 軸回りの回転角    */
                az = 0.0*pai/180,    /* z 軸周りの回転角    */
                vp = -300.0,          /*  投影中心           */
                l = 29.0,             /* x 方向の移動量       */
                m = -35.0,            /* y 方向の移動量       */
                n = -150.0,           /* z 方向の移動量       */
                x1,y1,z1,x2,y2,z2,h;
        x1 = x;
        y1 = y*cos(ax)-z*sin(ax);
        z1 = y*sin(ax)+z*cos(ax);

        x2 = x1*cos(az)-y1*sin(az);
        y2 = x1*sin(az)+y1*cos(az);
        z2 = z1;

        h = -x2*sin(ay)/vp+z2* cos(ay)/vp+n/vp+1;

        *px = (x2*cos(ay)+z2*sin(ay)+l)/h;
        *py = -((y2+m)/h);

}



void Points_Display(int i,double px,double py)
{
        int x,y;

        px=px+50;
        py=py+50;
        x=3*(double)px;
        y=3*(double)py;

        dx[i] = x;
        dy[i] = y;

        XAllocNamedColor(disp,cmap,"black",&c1,&cc);
        XSetForeground(disp,gc,c1.pixel);

        XDrawPoint(disp,win0,gc,x,y);

        XFlush(disp);
}

void Range_Label_Disp()
{
        int i,j,n,x,y,iro,l_max=0;
```

```
for(i=1;i<=lb_max;++i){

   for(j=0;j<=hdp;++j){

      n = hd[j];

      if(p[n].lb==i){


         x = dx[n];
         y = dy[n];

         iro = p[n].lb;

         while(iro>12)
            iro = iro-12;

         if(iro==1){
            XAllocNamedColor(disp,cmap,"red",&c1,&cc);
            XSetForeground(disp,gc,c1.pixel);
         }

         if(iro==2){
            XAllocNamedColor(disp,cmap,"green",&c1,&cc);
            XSetForeground(disp,gc,c1.pixel);
         }

         if(iro==3){
            XAllocNamedColor(disp,cmap,"blue",&c1,&cc);
            XSetForeground(disp,gc,c1.pixel);
         }

         if(iro==4){
            XAllocNamedColor(disp,cmap,"yellow",&c1,&cc);
            XSetForeground(disp,gc,c1.pixel);
         }

         if(iro==5){
            XAllocNamedColor(disp,cmap,"cyan",&c1,&cc);
            XSetForeground(disp,gc,c1.pixel);
}

if(iro==6){
    XAllocNamedColor(disp,cmap,"magenta",&c1,&cc);
    XSetForeground(disp,gc,c1.pixel);
}

if(iro==7){
    XAllocNamedColor(disp,cmap,"gray70",&c1,&cc);
    XSetForeground(disp,gc,c1.pixel);
}

if(iro==8){
    XAllocNamedColor(disp,cmap,"red4",&c1,&cc);
    XSetForeground(disp,gc,c1.pixel);
}

if(iro==9){
    XAllocNamedColor(disp,cmap,"green4",&c1,&cc);
    XSetForeground(disp,gc,c1.pixel);
    }

if(iro==10){
    XAllocNamedColor(disp,cmap,"blue4",&c1,&cc);
    XSetForeground(disp,gc,c1.pixel);
}

if(iro==11){
    XAllocNamedColor(disp,cmap,"yellow4",&c1,&cc);
    XSetForeground(disp,gc,c1.pixel);

}

if(iro==12){
    XAllocNamedColor(disp,cmap,"cyan4",&c1,&cc);
    XSetForeground(disp,gc,c1.pixel);

}
XDrawPoint(disp,win1,gc,x,y);
XFlush(disp);
```

```
                }

            }

            printf("Region_Number = %d¥n",i);
            getchar();

        }

}


void Line_Display(double px1,double py1,double px2,double py2)
{
        int x1,y1,x2,y2;

        px1=px1+50;
        py1=py1+50;
        px2=px2+50;
        py2=py2+50;

        x1=3*(double)px1;
        y1=3*(double)py1;
        x2=3*(double)px2;
        y2=3*(double)py2;

        XAllocNamedColor(disp,cmap,"red",&c1,&cc);
        XSetForeground(disp,gc,c1.pixel);
        XSetLineAttributes(disp,gc,1,LineSolid,CapButt,JoinMiter);

        XDrawLine(disp,win0,gc,x1,y1,x2,y2);

        XFlush(disp);

}
```