Zバッファ法を利用したオブジェクトおよび衝突検出

山地秀美† 新藤義昭†

本論文では，Zバッファ法を利用した，仮想空間におけるオブジェクトおよび衝突検出の手法を提案する．本手法によるオブジェクト検出は，検出領域をビューボリュームとして定義し，描画を行う．ここで生成されたZ値に，最遠点より小さい値の画素があれば，その位置にオブジェクトが存在する．衝突検出では2度の描画を行う．最初に，検出を行うオブジェクトの位置から検出方向に，そのオブジェクト以外を描画する．次に，検出領域の逆の位置から，検出するオブジェクトだけを描画する．ここで得られるZ値から，検出するオブジェクトと他のオブジェクトとの距離を求める．本手法は，オブジェクトの動き，形状やその変化によらず，1フレームの描画速度と検出を行う回数のみに依存する．

A Technique for Object and Collision Detection by Z-buffer

Hidemi Yamachi† and Yoshiaki Shindo†

This paper proposes a new technique to detect objects and collisions of geometric models in 3D cyber space. This technique uses Z-buffer values generated by the rendering system. To detect objects, the target area of detection is defined as the view volume and the scene is drawn. If some objects exist in the target area, some pixel's Z-values are less than the max value of Z-buffer. To detect collision, the viewpoint is set at the sensor object and the scene is drawn toward the direction to move. If no object is detected in the target area, no collision will occur. Otherwise, the viewpoint is set at the direction of the destination and the scene is drawn to obtain the Z-values of the sensor object's front face. The distance between the sensor object and target object(s) is obtained from these two Z-values. This technique costs once or twice drawing processes and it is independent from object's motion, complexity of their shape or deformation.

1. はじめに

仮想空間におけるオブジェクトの衝突検出は，CAD，ロボティックス，計算幾何学，コンピュータグラフィックスなど広範な分野で研究が進められ，さまざまなアルゴリズムが提案されている．

その多くは，オブジェクト相互の幾何学的な交差を求めるものである．AABB tree，OBB treeやsphere treesなどのように，外接する凸多面体（凸包）を再帰的に定義した階層木を使う手法，ボロノイ分割やBSP，octreeなどによって空間を分割し，検出対象となる隣接オブジェクト対を絞り込む手法などがある[1)~7)]．凸包を定義するための計算コストは形状の複雑さに強く依存し，形状が変化するたびに再定義する必要がある．また，形状変化，オブジェクトの動きや隣接するオブジェクトの数などにより，検出コストが大きく変化する．

これに対し，幾何学的な交差を計算しない方法として，ラスタライズ処理を利用したアルゴリズムが提案されている[8),9)]．このうちオブジェクト相互の衝突に関するShinyaらの手法[8)]は，隠面消去のために用いられるZバッファ法を利用し，オブジェクト外部の特定の視点から見たときにオブジェクトが存在するZ値の範囲を画素ごとに調べ，その重なりから衝突を判定する．すべてのオブジェクトを1つずつ描画してZ値を調べる．視点位置から見える前面のZ値だけでなく，背面のZ値も求める必要がある．この手法は，ポリゴン数Nに対し$O(N)$で検出できる．また，アルゴリズムが単純なため実装が容易で，グラフィックスハードウェアの高速化の恩恵を直接受けることができる．しかし，表面すべてのZ値を求めなければならないため，非凸多面体の衝突検出には有効ではない．

本論文で提案する手法*Cyber Radar*は，検出を行うオブジェクト（センサオブジェクト*Sensor-object*）の位置に検出のための視点（検出視点*Prove-point*）を

† 日本工業大学工学部情報工学科
Department of Computer and Information Engineering, Nippon Institute of Technology

設定し，運動によって掃き出す領域をビューボリュームとして Z 値の配列（距離配列 *Distance-array*）を作成する．オブジェクトが検出されれば，オブジェクト側から見たセンサオブジェクトの距離配列を作成する．2 つの距離配列からオブジェクト間の距離を求め，衝突を検出する．

このアルゴリズムは，以下の特徴を持っている．

(1) 検出視点はオブジェクトに設定されるため，オブジェクトの目として，オブジェクトとともに移動しながら検出を行う．
(2) 移動しようとする距離に応じて検出空間を定義するため，他のオブジェクトとの衝突を連続時間で検出できる．
(3) 1 回または 2 回の，低解像度で簡略化（テクスチャマップ，照光計算，半透明処理などの省略）した描画コストで衝突検出を行うことができる．
(4) オブジェクトの数，形状および変形，動きに依存せずに同一のアルゴリズムで検出できる．
(5) アルゴリズムが単純なため，実装が容易である．
(6) 検出領域のサイズ，解像度を変更することにより，検出精度を動的に変更できる．

本論文では，2 章でオブジェクト検出，3 章で衝突検出のアルゴリズムの詳細を述べる．4 章では，検出コストの測定結果およびグラスプシミュレーションへの応用について述べる．

2. *CyberRadar* によるオブジェクト検出

2.1 オブジェクト検出の手順

検出領域の一端に検出視点を設定し，直交投影法によるビューボリュームを定義し検出領域を描画することによって，視線方向の距離配列を取得する．この距離配列を，探査距離画像（*Range Distance Scope*）と呼ぶ．

探査距離画像を R とし，R を構成する各画素を，R_{ij} とする．i, j はそれぞれ距離配列の縦横の要素を表す．投影面の横の画素数を w_p，縦の画素数を h_p とすると，i, j はそれぞれ 1 から w_p，および h_p までの値をとる．検出領域の最遠点までの距離を d とすると，オブジェクトの有無は，以下の基準で判定する．

Some objects exist $min(R) < d$ (1)

No object exists $min(R) = d$ (2)

Cyber Radar によるオブジェクト検出アルゴリズムを以下に示す．

1. Set the drawing conditions (Fig.1)
   Set the prove point
   Set the direction toward the target area
   w_s, h_s := Width and height of the target area
   d := Distance of the target area
   w_p,h_p := Projection width and height
2. Draw the scene
3. Get R (Range Distance Scope)
4. *if* $min(R) = d$
   return false (no object is detected)
   else
   return true (object is detected)

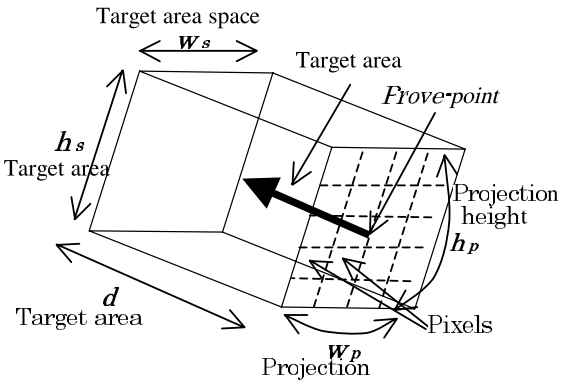


図 1 オブジェクト検出のためのビューボリューム定義
Fig. 1 Definition of viewing volume for object detection.

このアルゴリズムによって，同時に 1 つ以上のオブジェクトが検出できる．検出されたオブジェクトを識別し，その数を知るためには，色コードによる描画を利用することができる．これについては，2.3 節で述べる．

2.2 ビューボリュームの定義

透視投影法では，透視投影変換のために投影面上の位置から実際の位置を求めるには逆変換が必要になる．また投影面上に視点を設定することはできない．そこで，直交投影法で描画を行う（**図 1**）．投影面の幅と高さおよび画素数は，検出領域の大きさと検出精度によって，動的に変更する．

2.3 描画の簡略化と検出オブジェクトの識別

オブジェクト検出のために行う描画は，Z 値を利用するためだけに行われる．したがって，テクスチャ処理，照光処理，シェーディング，半透明処理を省略することができる．こうした簡略化により，描画コストは低減する．

これらの簡略化は同時に，色コードによるオブジェクト識別法[10] を併用できる．各オブジェクトにあらかじめ色コードを付与しておき，その色コードで描画する．探査距離画像からオブジェクトが検出されたら，カラーバッファの対応する画素の色を調べる．これに

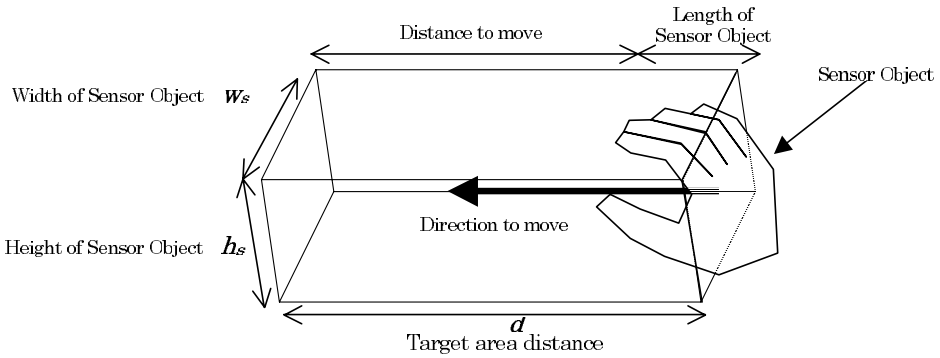


図 2 衝突検出のためのビューボリューム定義
Fig. 2 Definition of viewing volume for collision detection.

より，複数のオブジェクトが検出された場合でも，それらを識別できる．

3. *CyberRadar* による衝突検出

3.1 衝突検出の手順

衝突検出は，センサオブジェクトと検出対象のオブジェクトが向き合う対向面の間の距離を調べることによって行う．検出対象オブジェクトの対向面の凹凸は，探査距離画像から知ることができるが，センサオブジェクトの情報も必要である．そこで，センサオブジェクトを描画して取得した距離配列を利用する．これをマスク距離画像（*Mask Distance Scope*）と呼ぶ．

探査距離画像とマスク距離画像から，対応する画素の間の距離を求め，照準距離画像（*LookAt Distance Scope*）を作成する．

マスク距離画像を M，それぞれの画素の値を M_{ij} とする．ここで，i, j は 2.1 節と同様である．照準距離画像も同様に，L および L_{ij} で表す．

ビューボリュームの奥行き d は，オブジェクトの移動距離とセンサオブジェクトの長さの和にする．検出視点から検出されたオブジェクトまでの各画素の距離は，

$$d - R_{ij} \tag{1}$$

移動先からセンサオブジェクトまでの距離は，

$$d - M_{ij} \tag{2}$$

となる．これらの値から，照準距離画像の画素の値は次のように求められる．

$$\begin{aligned} L_{ij} &= d - (d - R_{ij}) - (d - M_{ij}) \\ &= R_{ij} + M_{ij} - d \end{aligned} \tag{3}$$

L の最小値 $min(L)$ が d 以下であれば，移動によって衝突が起きる．この最小値と移動速度から，衝突の発生する時間が，連続時間で求められる．

衝突検出の手順を以下に示す．

1. Set up the drawing condition (Fig.2)
   Set the prove point at Sensor Object
   Set the direction to move
   w_s,h_s := Width and height of Sensor Object
   d := Distance to move
   + Length of Sensor Object
   w_p,h_p := Projection width and height
2. Draw the scene without Sensor Object (Fig.3)
3. Get R (Range Distance Scope)
4. *if* $min(R) = d$
   return false(no collision)

1. において，ビューボリュームは，センサオブジェクトの移動によって掃き出される空間を包含する幅と高さ，奥行きを持つように定義する（**図 2**）．2. では，センサオブジェクト以外を描画して探査距離画像を取得する（**図 3**）．探査距離画像にオブジェクトが検出されなければ，衝突は発生しない．検出された場合は，以下の処理を続ける．

5. Reset the drawing condition (Fig.4)
   Set the prove point at the destination
   Reverse the direction
6. Draw the scene (only Sensor Object)
7. Set M (*Mask Distance Scope*) (Fig.4)
   Swap right and left pixel values
8. Set L (*LookAt Distance Scope*)(Fig.5)
   $L = R + M - d$
9. *if* $min(L)$ >Distance to move
   return false (no collision is detected)
   else
   return true (collision is detected)

5. で検出視点を移動先に設定し，検出方向を逆にする．センサオブジェクトだけを描画して，マスク距離画像を取得する（**図 4**）．ここで，探査距離画像と対

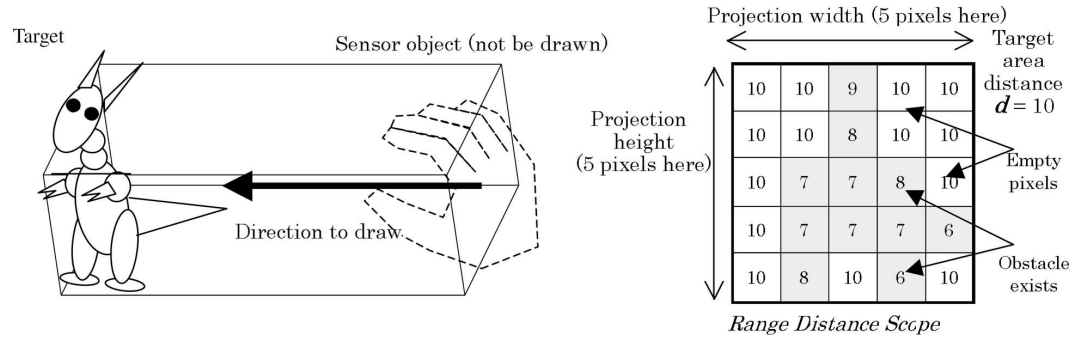


図 3 衝突検出のための探査距離画像の作成
Fig. 3 Generation of *Range Distance Scope* for collision detecion.

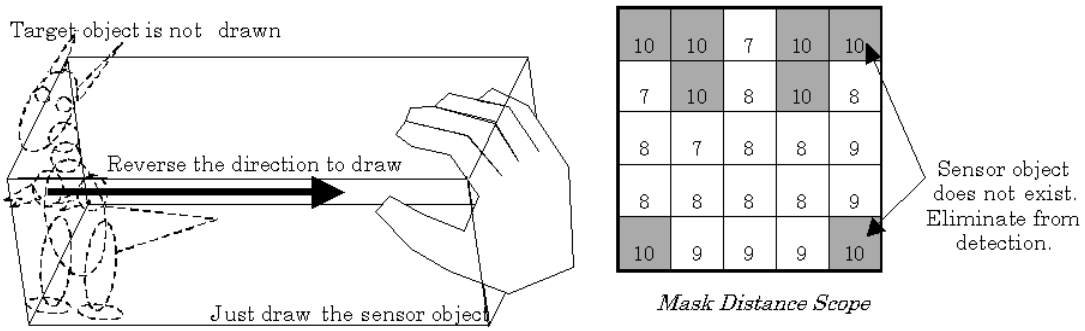


図 4 マスク距離画像の作成
Fig. 4 Generation of *Mask Distance Scope*.

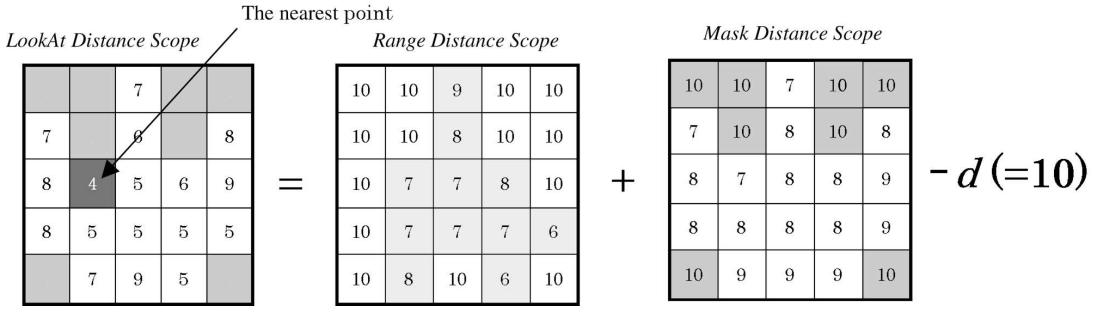


図 5 照準距離画像の作成
Fig. 5 Generation of *LookAt Distance Scope*.

応させるために，左右を逆にしなければならない．

8. で照準距離画像を作成し，移動距離以下の画素の有無を調べる（**図 5**）．このとき，マスク距離画像で d に等しい値を持つ画素は，センサオブジェクトの存在しない領域を意味する．これらの画素は，衝突検出から除外される．8. の処理は，以下ののように行われる．

repeat $i := 1$ to w_p, $j := 1$ to h_p
  if $M_{ij} = d$
    L_{ij} := Eliminate Mark
  else
    $L_{ij} := R_{ij} + M_{ij} - d$

9. の *if* 文は，センサオブジェクトの存在する画素に対してだけ処理される．マスク距離画像上のセンサオブジェクトの形状が非凸または，離れた複数の形状から構成されていても，この処理によって衝突の判定が可能になる．

3.2 衝突中心と衝突面の法線ベクトルの推定

衝突が検出されたとき，衝突する画素の位置から，

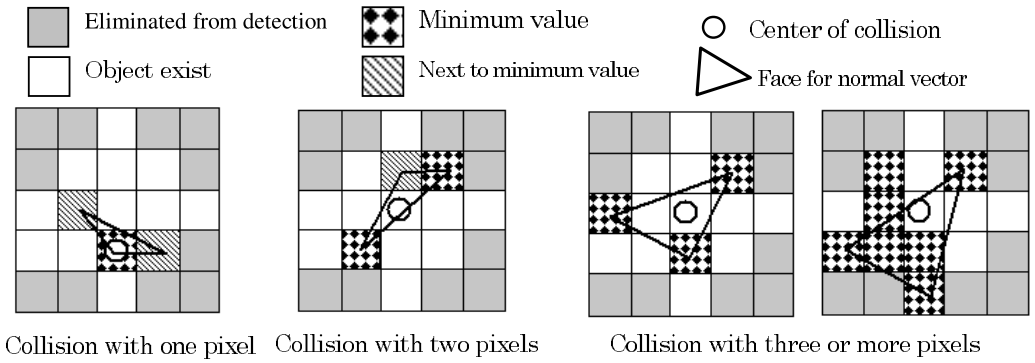


図 6 衝突中心と衝突面の法線ベクトルの推定
Fig. 6 Estimation of center of collision and normal vector.

衝突中心位置および衝突面の法線ベクトルを推定できる．衝突する画素の数は，1 点，2 点，3 点以上の 3 つの場合に分けることができる．3 点以上の場合は，すべての点の外接凸多角形を求め，その重心を衝突中心とする．法線はいずれかの 3 点から求められる．2 点の場合は 2 点の中点を，1 点の場合はその点を衝突中心とする．法線を求めるために，衝突する画素の周辺の，最も接近した画素を利用する（**図 6**）．

3.3 マスク距離画像の再利用と，ビューボリュームの動的定義

本手法では，衝突検出のために 2 度の描画が必要である．しかし，検出方向から見たセンサオブジェクトの形状が変化しなければ，1 度作成したマスク距離画像を再利用できる．

センサオブジェクトの形状や姿勢が変化した場合，マスク距離画像を作成する前に，ビューボリュームの幅と高さを再定義しなければならない．これも，マスク距離画像を利用して，以下のように行う．

```
    if Shape or moving direction changed
      MaskReset := true
    ........
  if MaskReset = true then begin
    if ws, hs = MaxSize then begin
      ws, hs := Get size from Mask
      MaskReset := false
    end
  else begin
      ws, hs := MaxSize
      Create Mask Distance Scope
  end
```

予想されるセンサオブジェクトの最大サイズ（ここでは MaxSize）をあらかじめ定義しておく．マスク距離画像を作成するとき，ビューボリュームの幅と高さを，最大サイズに設定する．次の検出の際に，マスク距離画像を解析して，センサオブジェクトの存在する位置の画素の座標から，その幅と高さを求め，適切なサイズのマスク距離画像を作成する（図 9）．

4. 実験および評価

4.1 性能評価実験の種別と実験環境

Cyber Radar による衝突検出コストは，探査距離画像とマスク距離画像の作成のための描画コストおよび，その解析コストに分けられる．描画コストは GPU（Graphics Processing Unit）の描画能力に大きく依存し，解析処理は CPU のみに依存する．そのため本論文では Windows2000 と OpenGL を使用し，文献 11) を参考に CPU と GPU に関して**表 1** に示す 4 つの環境を用い，それぞれについて以下のような実験を行った．

(1) 衝突検出コストの測定および距離画像作成のための描画と通常描画の比較
(2) 複数の *Cyber Radar* を同時に使った場合の衝突検出コストの変化

最後に，より複雑な動作を行うグラスプシミュレーションへの適用例を示した．

4.2 衝突検出コストの測定実験

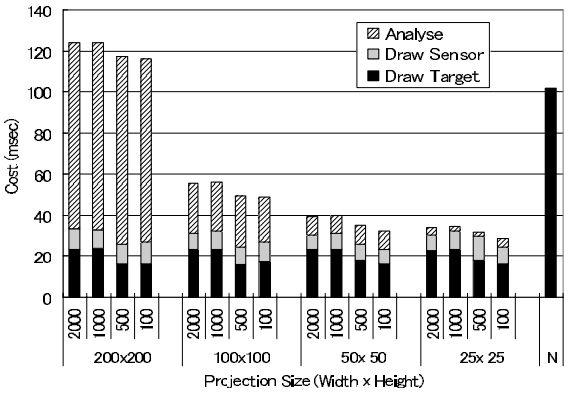
距離画像作成のための描画は，テクスチャマップ，照光計算，半透明処理などを省略して描画するため，通常よりも高速な描画が期待できる．また距離画像の解析は画素ごとに行われるため，画素数により検出コストが異なる．

そこで，以下の条件で実験および評価を行った．

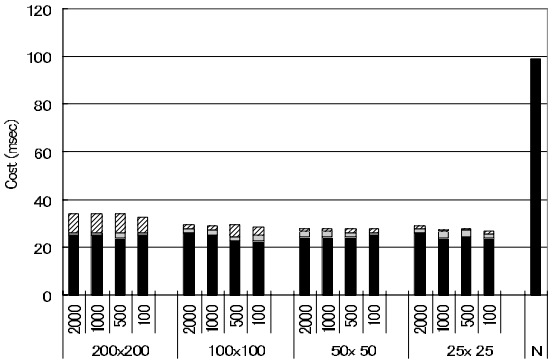
(1) 仮想空間に，テクスチャ，透明，非テクスチャの不透明の 3 種類の球を 2:1:1 で配置した．
(2) リアルタイムでの衝突検出を目的にしているた

表 1 実験に用いた 4 種類の実験モデルの構成
Table 1 Models of hardware for experiment.

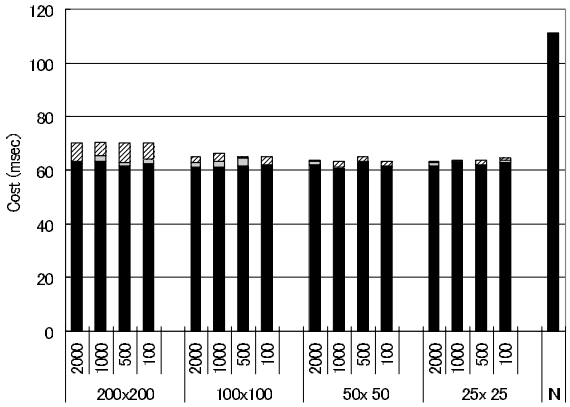
model	CPU	GPU	uses
Low-end	Pentium3 450 MHz	3dLab Oxgen VX1	3D-Game
Middle-class	Pentium3 1 GHz	NVIDIA GeForce2MX	Light CG-Animation, 3D-Game
High-class	Pentium 4 1.8 GHz	NVIDIA GeForce3	High Level CG-Animation, Visualization
Professional	Xeon 1.8 GHz x2	3dLab Wildcat 5110	3D-CAD, Visualization, High Level CG-Animation



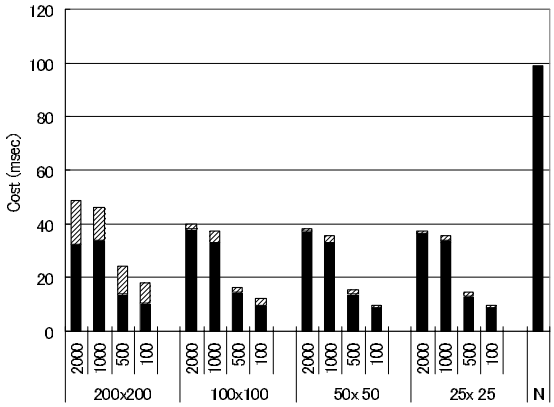
(a) Low-end model: 13,074 Polygons



(b) Middle-class model: 26,898 Polygons



(c) High-class model: 46,194 Polygons



(d) Professional model: 100,626 Polygons

2000-100 : Viewing Volume Size (Width, Height and Depth)
200×200-25×25 : Projeciton Size (Width and Height)
N : Normal Drawing

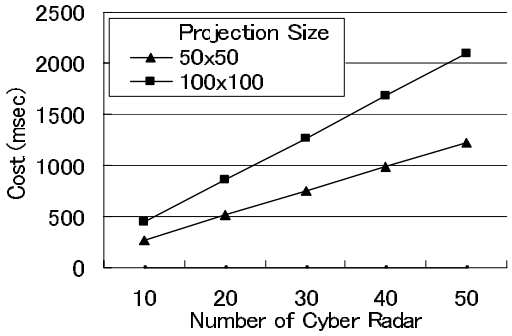
図 7 衝突検出コスト
Fig. 7 Cost of collision detection.

め，通常描画のフレームレートが 100 msec 前後になるように，センサオブジェクト（ポリゴン数 1250 の球）を含めたポリゴン数を調整した．

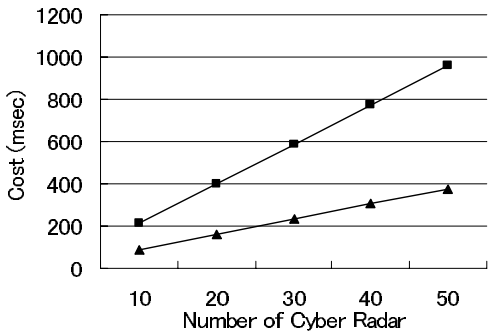
(3) 検出対象となるビューボリュームを立方体とし，一辺の長さを 2000 から 100 まで変化させ，クリッピングによる描画速度を比較した．

(4) 画素数を 200 × 200 から 25 × 25 まで変化させ，解析コストを比較した．

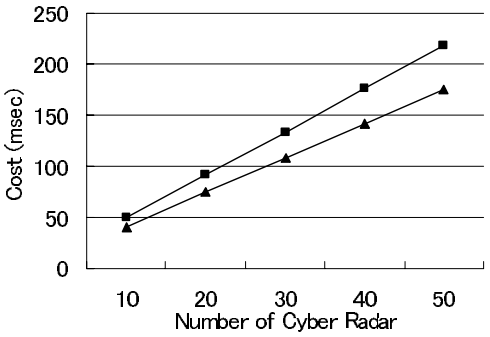
フレームレートを 100 msec 前後に調整した結果，各モデルのポリゴン数は Low-end model から順に 13,074，26,898，46,194，100,626 となった．**図 7** の各グラフの右端は通常描画の描画コストを示している．これと比較すると，探査距離画像（Draw Target）とマスク距離画像（Draw Sensor）の描画コストの合計は，60%から 10%程度にまで減少することが分かった．画素ごとの解析コストは画素数の増大にともなって大きくなるが，Low-end model 以外では数 msec から 10 msec 程度となった．



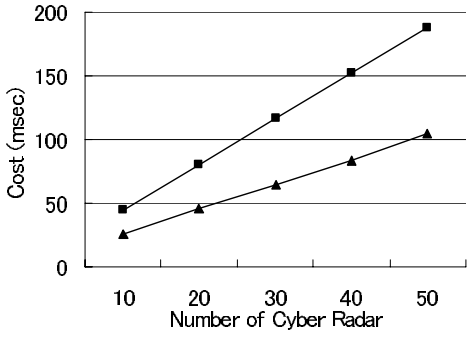
(a) Low-end model: Normal Drawing 28.1 msec



(b) Middle-class model: Normal Drawing 19.6 msec



(c) Heigh-class model: Normal Drawing 10.0 msec



(d) Professional model: Normal Drawing 8.0 msec

Number of total polygons: 6406

図 8 *Cyber Radar* の数に対する衝突検出コスト
Fig. 8 Detection cost for number of *Cyber Rardar*.

4.3 複数の $CyberRadar$ による衝突検出コスト

立方体の中に球を配置し，衝突検出によって跳ね返りながら運動する球の数を変化させ，1 フレームあたりのコストを測定した（**図 8**）．球の数は 50 個で各ポリゴン数は 128 とし，そのうち *Cyber Radar* により衝突検出を行いながら反跳運動をする数を 10 から 50 まで変化させた．それぞれについて，画素数 50×50, 100×100 の場合を測定し，1 フレームあたりの時間を計測した．ポリゴン数は 6406，テクスチャおよび半透明オブジェクトは含まれていない．また衝突検出の際，マスク距離画像の再利用は行っていない．

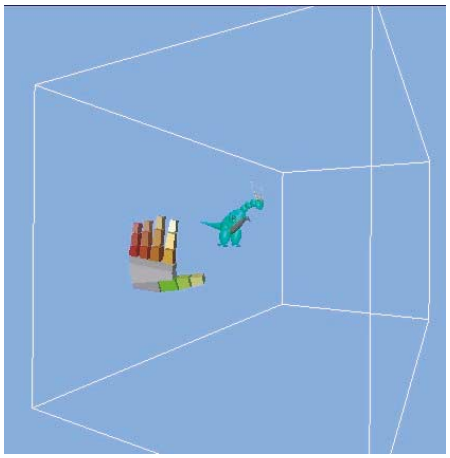
Cyber Radar の数が 10 以内ならば，Low-end model でもアニメーションでの適用が可能と思われるが，20 個に達すると Middle-class model が，それ以上になると High-class model 以上の動作環境が必要であることが分かった．実験結果から，*Cyber Radar* の数と衝突検出コストは，ほぼ線形であることが確認できた．

4.4 グラスプ・シミュレーションへの応用

本論文で行ったグラスプ・シミュレーションは，以下のステップで構成される．

(1) ターゲットの検出（**図 9**）
ターゲットの ID 番号を指定する．
検出領域を広く設定する（**図 10** 左）．
回転しながら，ターゲットを検出する．
(2) ターゲットへの接近（**図 11**）
マスク距離画像から手のサイズを取得する．
検出領域を手のサイズに変更する（図 10 右）．
衝突検出を行いながら，平行移動する．
(3) ターゲットを掴む（**図 13**）

手全体および**図 12** に示す 17 の *Cyber Radar* を配置した．ターゲットに接触するまでは手全体をカバーする *Cyber Radar* が 1 つだけ検出を行う．続いて 4 指を曲げる（図 12 a，図 13 a）．衝突が検出されると，5 指それぞれに配置された *Cyber Radar* により，各指を独立して曲げる．指全体から先の部分へ，動かす関節を移動させる（図 12 b，c，d，e および図 13 b）．運動する部分の変化に応じて，機能する *Cyber Radar* を変化させていく．他の指や手のひらの検出により，ターゲットに接触しなかった指は，手への接触



Target Area Size : W 2000, H 2000, D 8000
Target Size : W 300 H 500 D 800
Hand Size : W 400 H 400 D 30

図 9 オブジェクト検出のためのビューボリューム
Fig. 9 Viewing volume for object detection.

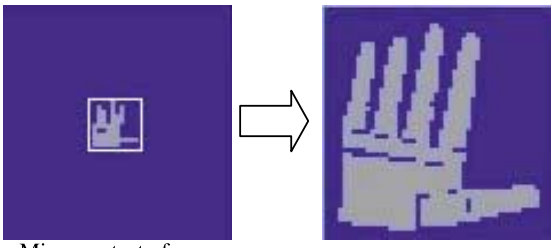
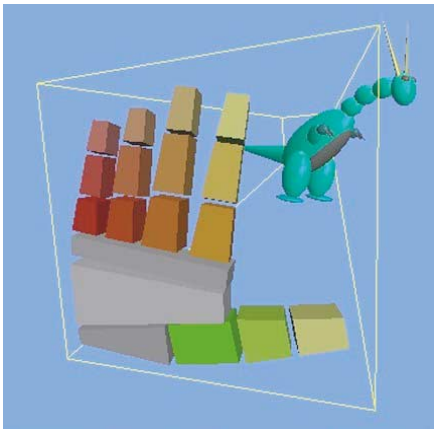


図 10 マスク距離画像によるビューボリュームサイズ変更
Fig. 10 Reset target area space size by *Mask Distance Scope.*



Target Area Size: W 440 H 440 D 1000

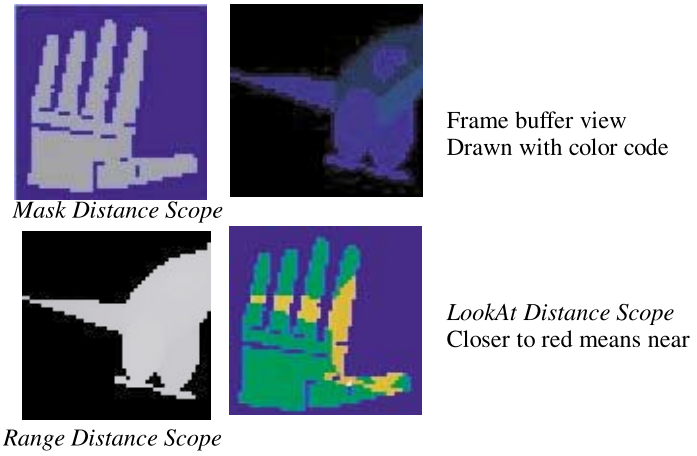
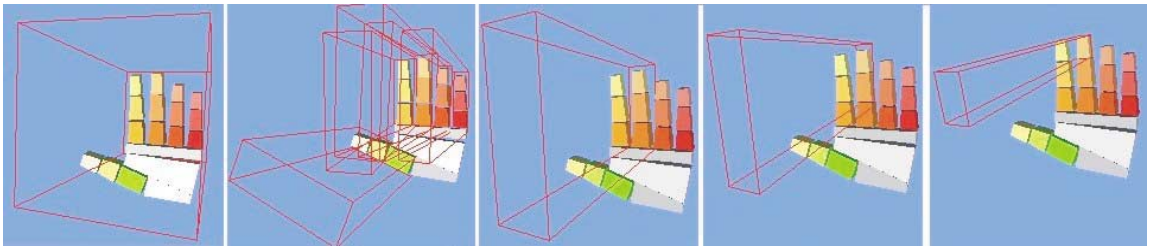
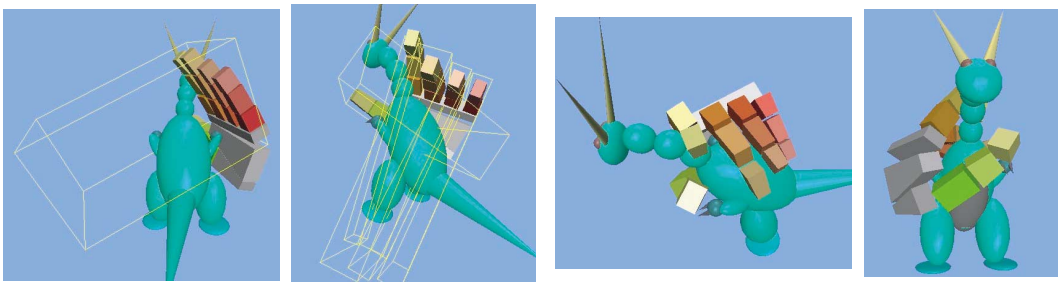


図 11 衝突検出
Fig. 11 Collision detection.



(a) (b) (c) (d) (e)

図 12 手に設定された *Cyber Radar*
Fig. 12 *Cyber Radars* of hand.



(a) (b) (c) (d)

図 13 グラスプシミュレーション
Fig. 13 Grasp simulation.

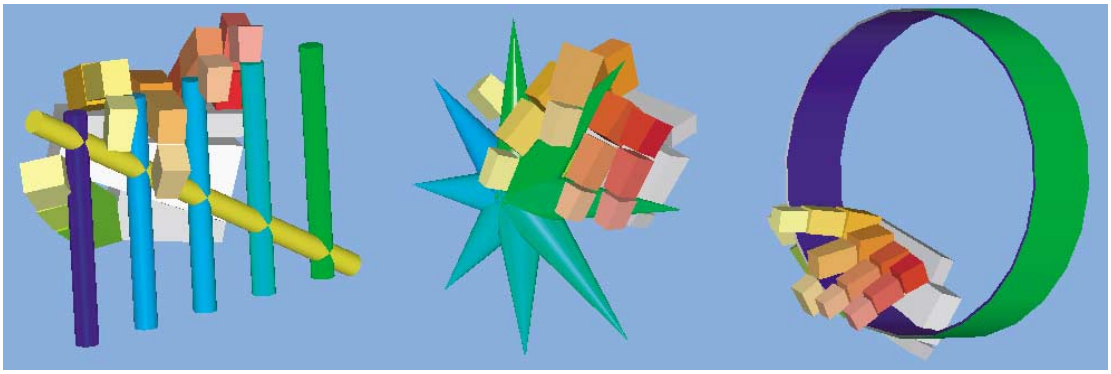


図 14 さまざまな形状オブジェクトによるシミュレーション
Fig. 14 Simulation by various models.

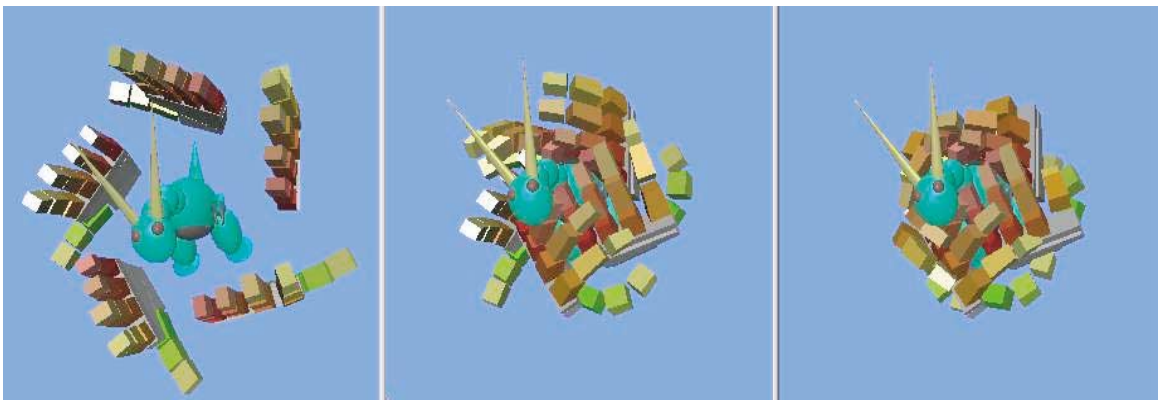


図 15 5 つの手によるシミュレーション
Fig. 15 Simulation of five hands.

で停止する．折り曲げる回転角は，最も接近した点の位置から回転軸までの距離とターゲットまでの距離の正接から求める．1 つのターゲットを 5 つの手により同時に掴む場合のフレームレートと衝突検出回数を，Professional model を用いて計測した（**図 16**）．最大 52 の *Cyber Radar* が同時に衝突検出を行っているが，このときのフレームの最大描画コストは 203 msec であった．相互に動きのある複雑な形状に対するリアルタイムアニメーションにおいて，*Cyber Radar* による衝突検出が有効に機能することを確認した．

4.5 評 価

Cyber Radar の処理コストおよびグラスプ・シミュレーションへの応用実験を行った結果，以下の点が明らかとなった．

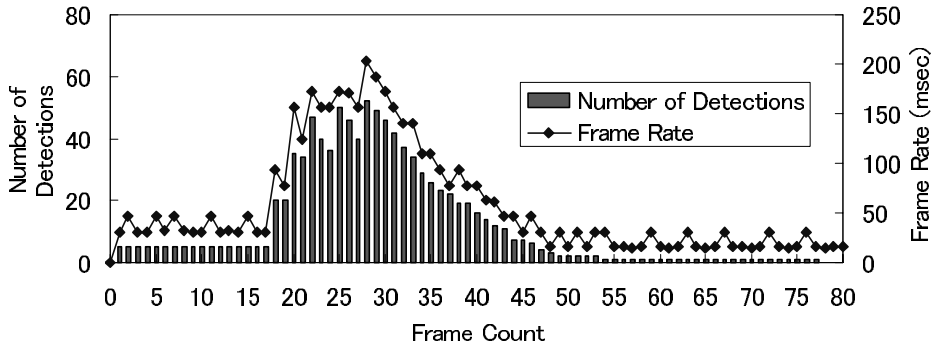


図 16 グラスプシミュレーションにおける検出回数とフレーム間隔
Fig. 16 Number of detections and frame rate of grasp simulation.

(1) 検出のための描画コストは，通常の描画コストの 10%〜60%の範囲であった．
(2) 解析コストは画素数の増大にともなって大きくなるが，Low-end model 以外では数 msec から 10 msec 程度となった．
(3) リアルタイムアニメーションを行う場合において同時作動可能な *Cyber Radar* の個数は，Low-end model では 10 未満，Middle-class model では 20 以内，High-class model では 40 以内であった．距離画像の画素数を抑えれば，Professional model では 50 以上を同時に作動させることが可能である．
(4) グラスプシミュレーションにおけるようなターゲットの形状の動きや変形に依存せずに衝突検出ができる．

衝突検出の従来手法では，形状の複雑さや変形によって計算量の増大が避けられない[2]．しかし *Cyber Radar* は，形状の複雑さや変形に依存せず，描画時間のオーダ以内で同一の精度の検出を行うことができることを確認した．

5. おわりに

Cyber Radar は，仮想空間を「見る」ことにより，オブジェクトおよび衝突を検出する．ここで「見る」とは，検出領域を描画し，そのとき作成された距離画像から，オブジェクトの存在と距離を検出することである．衝突検出では，検出を行うセンサオブジェクトを運動方向から見た距離画像を作成する．2 つの距離画像から相互の距離を求める照準距離画像を作成し，衝突の有無を判定し，衝突中心と衝突面の法線ベクトルの推定処理を行う．本手法は，(1) オブジェクトの形状を記述したデータ構造には依存せず，Z バッファ法が利用できる環境ならば適用可能である，(2) 形状の複雑さや動きに依存せず，描画時間のオーダで検出できる，(3) GPU の性能向上による処理コストの低減が期待できる，などの特徴を持つことから，リアルタイム CG アニメーションにおけるオブジェクトの探査および衝突検出において有効であるといえる．

参考文献

1) Lin, M. and Gottschalk, S.: Collision Detection between Geometric Models: A Survey, *Proc. IMA Conference on Mathematics of Surfaces* (1998).
2) 北村：衝突検出を利用した仮想物体操作，ヒューマンインタフェース学会誌，Vol.1, No.4, pp.25–33 (1999).
3) Gottschalk, S., Lin, M. and Manocha, D.: OBBTree: A Hierarchical Structure for Rapid Interference Detection, *Proc. ACM Siggraph '96*, pp.171–180 (1996).
4) van den Bergen, G.: Efficient Collision Detection of Complex Deformable Models using AABB Trees. *Journal of Graphics Tools*, Vol.2, No.4, pp.1–14 (1997).
5) Hubbard, P.M.: Approximating Polyhedra with Spheres for Time-Critical Collision Detection, *ACM Trans. Graphics*, Vol.15, No.3, pp.179–210 (1996).
6) Klosowski, J.T., Held, M., Mitchell, J.S.B., Sowizral, H. and Zikan, K.: Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs, *IEEE Trans. Visualization and Computer Graphics*, Vol.4, No.1 (1998).
7) Lin, M., Manocha, D., Cohen, J. and Gottschalk, S.: Collision Detection: Algorithms and Applications, Algorithms for Robotics Motion and Manipulation, Laumond, J.-P., Overmars, M. and Peters, A.K. (Eds.), pp.129–142 (1996).
8) Shinya, M. and Forgue, M.: Interference detection through rasterization, *Journal of Visualization and Computer Animation*, Vol.2,

pp.132–134 (1991).
9) Lee, S., Heo, J. and Wohn, K.: An efficient collision detection algorithm using range data for walk-through systems, *Proc. ACM Symposium on Virtual Reality Software and Technology*, pp.119–123 (1997).
10) Neider, J., Davis, T. and Woo, M.: *OpenGL Programming Guide Release 1*, p.371, Addison Wesley (1993).
11) De Gelas, J.: www.opengl.org /Professional Grade: OpenGL Accelerators Reviewed (Jan. 6, 2002).

(平成 13 年 7 月 16 日受付)
(平成 14 年 3 月 14 日採録)

山地　秀美（正会員）

昭和 30 年生．昭和 55 年埼玉大学教育学部，平成 2 年立教大学理学部卒業．平成 7 年日本工業大学非常勤講師．平成 13 年同大学情報工学科専任講師．コンピュータグラフィックス，バーチャルリアリティの研究に従事．IEEE 会員．

新藤　義昭（正会員）

昭和 29 年生．昭和 51 年電気通信大学卒業．同年富士通（株）入社．平成 5 年より日本工業大学工学部電気電子工学科専任講師．平成 7 年同大学情報工学科専任講師．平成 10 年情報工学科助教授．画像情報工学，コンピュータグラフィックス，バーチャルリアリティの研究に従事．特に，これらを活用した新たな情報メディアの開発や，情報処理演習における教授法や講師支援システムの開発に力を入れている．著書「OpenGL リアルタイム 3D プログラミング」（秀和システム），「バーチャルリアリティプログラミング」（NEC クリエイティブ）等．電子情報通信学会会員．