連続な直線が通るすべてのラスタ要素を求める 高速 DDA アルゴリズム

横 山 健一郎[†], 正会員 千 葉 則 茂[†], 田 山 典 男[†]

1. ま え が き

DDA (Digital Differential Analyzer) は直線 (実際は線分であるが, 慣習に従ってこう呼ぶ) のスキャン変換を求める, すなわち, デジタル直線を求めるアルゴリズムである. CRT への直線の描画に用いられるような DDA では, 必ずしも与えられた連続的な直線が通過するすべてのラスタ要素 (ピクセルやボクセル) をスキャン変換に含む必要はないが, ここではそれらすべてをスキャン変換に含むようなアルゴリズムについて考える. このような“探索向き” DDA は 3 次元医用画像などのボリュームレンダリングや¹⁾, レイトレーシング法の高速化手法である空間等分割法^{2)~4)}におけるボクセル空間の横断に重要な応用をもつ. CRT への直線描画のための高速 DDA としては, Bresenham のアルゴリズムが有名であるが, これは連続的な直線が通るすべてのラスタ要素を生成するアルゴリズムではない. また, 整数演算のみで実行でき, 高速であるのは, 直線の端点座標が整数のときであることを注意しておく.

これまでに, 探索向き DDA がいくつか提案されているが^{1)~4)}, 複数の文献により提案されている典型的な DDA (これを標準 DDA と呼ぶことにする)²⁾³⁾ など以外は, アルゴリズムの詳細⁵⁾や, 必要とされる時間計算量¹⁾が明らかでない. 筆者らは, この標準 DDA に対して, 必要とされる実数演算の回数が, 2 次元 DDA で 70%, 3 次元 DDA で 55% 少ないアルゴリズムの存在を示した. 全体の計算時間については, 実験

評価により, 2 次元 DDA で約 1.8 倍, 3 次元 DDA で約 1.5 倍の高速化となることを確認した.

2. 標準 DDA

本章では, 高速化の比較対象とする標準 DDA について説明をする. スキャン変換をするラスタ空間 (ディスプレイやボクセル空間) には, 直交座標系が定義されているとする. すなわち, 2 次元のときには xy 座標系が, 3 次元のときには xyz 座標系が定義されているとし, ピクセルやボクセルのそれぞれのラスタ要素は整数座標 (インデックス) で識別されるとする. ラスタ要素のそれぞれの軸方向の大きさは 1 とし, その整数座標はラスタ要素内の任意の点の実数座標の整数部分と一致するとする.

標準 DDA の方法を, 簡単のため 2 次元の場合を例にとり説明する (図 1 参照). また, 直線の方法は成分はいずれも正として説明する. 他の場合も同様である.

このアルゴリズムは, 現在のラスタ要素のインデックスが各座標軸方向について直線のどの位置まで有効であるかを示す直線上での距離 (任意の点からのもの

図 1 標準 2DDA
The reference 2DDA.

[†] 岩手大学 工学部情報工学科

“A Fast DDA Algorithm for Finding All the Raster Elements through Which the Given Continuous Line Segment Runs” by Kenichiro Yokoyama, Norishige Chiba and Norio Tayama (Faculty of Engineering, Iwate University, Morioka)

でよいが、通常は始点からのものを使う)、すなわち、直線が現在のラスタ要素から x, y のそれぞれの方向について初めて通過するラスタ要素の境界までの距離 t_x, t_y をもち、その比較により次に増加させるべきインデックスを決定するというものである。

図1のように、 x, y 方向に1ラスタ要素分進む場合の直線上の距離をそれぞれ dx, dy とする。 t_x と t_y を比較することにより、直線が現在のラスタ要素のどの境界を通過するかを知ることができる。つまり、2つの距離の小さい方の軸方向に直交する境界を次に通過することがわかる。したがって、 t_x と t_y の小さい方の軸方向のインデックスが次に増加させるべきインデックスとなる。

以下に、標準3DDDAのアルゴリズムを示す。以降で示すアルゴリズムについてもそうであるが、ここで扱うアルゴリズムは、必ずしも直線が通過するラスタ要素を過不足なくちょうど生成するとは限らないことを注意しておく。すなわち、ラスタ空間を斜めに走る直線が、あるラスタ要素からその頂点(格子点)を通り26近傍のラスタ要素へ突入する場合でも、その頂点を共有する余分な6近傍のラスタ要素を生成することにより、得られるデジタル直線はすべて6連結となるようにアルゴリズムを統一している(2DDDAでは4連結である)。ここで、「26近傍」や「6連結」という用語は、2次元ラスタ空間における用語である。「8近傍」や「4連結」の3次元ラスタ空間への拡張である。

次の3DDDAの記述より、新たなインデックスを1つ生成するためには、繰り返しのループを構成するための演算を除いて、比較が2回、整数の加算、実数の加算がそれぞれ1回ずつ必要であることがわかる。

```
{標準3DDDA:
  i, j, k      : x, y, z それぞれの方向のインデックス
  dx, dy, dz   : x, y, z それぞれの方向に1ボックス分進む
                 ときの直線上の距離
  tx, ty, tz   : インデックス i, j, k がそれぞれ x, y, z 方
                 向に最大に有効な直線上の距離
}
スキャン変換すべき直線に応じて, dx, dy, dz, tx,
ty, tz, i, j, k の初期値設定を
行う;
write(i, j, k);
while (i, j, k)が直線の終点の整数座標と不一致 do
  begin
    if tx <= ty
    then if tx <= tz
        then begin i := i+1; tx := tx+dx end
        else begin k := k+1; tz := tz+dz end
    else if ty < tz
        then begin j := j+1; ty := ty+dy end
        else begin k := k+1; tz := tz+dz end;
    write(i, j, k)
  end;
```

3. DDA の高速化

ここでは、直線の傾きによって場合分けを行うことにより、DDAの実数演算の回数を減らすことができることを示す。このアルゴリズムを“改良DDA”と呼ぶことにする。標準DDAではインデックスが変化するたびに実数演算が必要とされるが、改良DDAでは、直線方向ベクトルの各成分のうち絶対値が最大の方向のインデックスが変化する際には実数演算を必要としない。すなわち、最も多くインデックスの更新が行われる方向について実数演算が不要となるため、高速化が期待できる。

簡単のため、まず2次元の場合を例にとり説明する。さらに、図2のような場合、すなわち直線方向ベクトルの成分はいずれも正とし、かつ x 成分が y 成分より大きいとして説明する(この場合、 x 方向を基準方向と呼ぶ)。

y 方向の1ラスタ要素分の変化に対する基準方向の変化量を dX_j とすると、 x 軸上の距離で dX_j 進むまでは無条件に x 方向のインデックス i を増加させることができる。したがってこの方法では、基準方向のインデックスの増分計算のための実数演算は必要ない。以下に、改良3DDDAを示す。

```
{改良3DDDA:
  i, j, k      : x, y, z それぞれの方向のインデックス
  dXj, dXk     : y, z 軸それぞれの方向に1ボックス分進む
                 ときの x 軸方向の増分
  Xj, Xk       : インデックス j, k が有効な範囲での x 座標
}
スキャン変換すべき直線に応じて,
dXj, dXk, Xj, Xk, i, j, k の初期値設定を行う:
{scan conversion}
write(i, j, k);
while i < e do
  begin
    if Xj < Xk
    then
      begin
        if Xj < i+1 then
          begin
            j := j+1; Xj := Xj+dXj; write(i, j, k);
            if Xk < i+1 then
              begin
                k := k+1; Xk := Xk+dXk; write(i, j, k)
              end
            end
          end
        else {Xk <= Xj}
          if Xk < i+1 then
            begin
              k := k+1; Xk := Xk+dXk; write(i, j, k);
              if Xj < i+1 then
                begin
                  j := j+1; Xj := Xj+dXj; write(i, j, k)
                end
              end;
            i := i+1;
            write(i, j, k)
          end;
```

図 2 改良 2 DDDA
The improved 2 DDDA.

ここでも、直線の方方向ベクトルの成分はいずれも正とし、かつ x 成分が y, z 成分より大きいとして説明する。他の場合も同様に書ける。

4. 高速化の評価

本章では、標準 DDA と改良 DDA の速度評価を行う。まず、実数演算の回数について理論的な評価を行い、次に初期値の計算や整数演算等を含めた全体の計算時間を評価するために、DDA のインプリメントを行い、計算時間を測定した実験結果を示す。評価は、ある始点 0 から長さ r の直線をランダムな方向に引いた場合の平均的な速度で行う。

4.1 理論評価

(1) 2 DDDA について

以下のように評価できる。

- 標準 2 DDDA の場合：標準 3 DDDA から類推されるように、長さ r の直線を $0 \leq \theta \leq \pi/4$ の範囲でスキャン変換する場合の実数演算の回数は、インデックスの変化の回数と一致するので、次式で表される量に比例することがわかる (図 3 参照)。

$$\int_0^{\pi/4} (r \cos \theta + r \sin \theta) d\theta = r$$

- 改良 2 DDDA の場合：改良 3 DDDA から類推されるように、 x 方向が基準方向であるとするとき、 y 方向が変化するとき以外は実数の加算が必要ないことがわかる。したがって、実数演算の回数は以下の量で見積れる。

$$\int_0^{\pi/4} r \sin \theta d\theta \approx 0.293r$$

これより、標準 2 DDDA に比べ、実数演算の回数が約 30% に減少していることがわかる。

図 3 2 DDDA の理論評価
Theoretical estimation for 2 DDDA.

図 4 3 DDDA の理論評価
Theoretical estimation for 3 DDDA.

(2) 3 DDDA について

以下のように評価できる。

- 標準 3 DDDA の場合：アルゴリズムより、次式で表される量で評価できる。直線の方方向ベクトルのそれぞれの成分のうち、 x 成分が最も大きいような直線の終点が存在する領域は図 4 の斜線部である。ここでは、積分計算の簡単のため、実線で囲まれた領域について評価する。この余分な領域 (y 成分が最大となる領域) について、 x 成分が最大という条件のもとでの改良 DDA を評価することは、改良 DDA にとって不利となるので、高速化の主張には差し支えないことに注意されたい。したがって、以下のように実数演算の回数が見積れる。

図 5 計算時間
Running time.

$$\int_0^{\pi/4} \int_0^{\pi/4} (r \sin \theta + r \cos \theta \sin \phi + r \cos \theta \cos \phi) d\theta d\phi \approx 0.937r$$

・改良 3DDDA の場合：アルゴリズムの記述より，以下の量で見積れる．改良 3DDDA については，直線の y, z 軸への投影の積分で評価できる．

$$\int_0^{\pi/4} \int_0^{\pi/4} (r \sin \theta + r \cos \theta \sin \phi) d\theta d\phi \approx 0.437r$$

これらの結果より，実数演算の回数は約 46% に減少していることがわかる．

4.2 実験評価

標準 DDA と改良 DDA を，初期値の計算，整数演算，さらに比較やループを構成するための比較や分岐命令を含めた計算時間全体について評価するため，これらのアルゴリズムをインプリメントし実験評価を行った．実験結果を図 5 に示す．使用した計算機はワークステーション NWS-1460 (ソニー製) で，言語は C である．計算時間は，理論評価の積分範囲と同様に，半径 r の円周もしくは球面上に向け球の中心から 100 本の直線を引くのに要する時間を測定したものである．

結果より，改良 DDA アルゴリズムは標準 DDA ア

ルゴリズムに比べて，2 DDDA で約 1.8 倍，3 DDDA で約 1.5 倍の高速化が達成されていることがわかる．

5. む す び

複数の文献により提案されている典型的な標準 DDA に対して，必要とされる実数演算の回数が，2 次元 DDA で 70%，3 次元 DDA で 55% 少ないアルゴリズムを考案した．全体の計算時間については，実験評価により，2 次元 DDA で約 1.8 倍，3 次元 DDA で約 1.5 倍の高速化となることを確認した．改良 DDA は実数演算の回数が少ないので，高速であるというだけでなく，浮動小数点演算による誤差の影響を受けにくい良好な DDA となっていると考えられる．この点についての検討も興味深い．

(1991 年 6 月 20 日受付)

〔参 考 文 献〕

- 1) 田山，清水，千葉，太田原：“切出し立体画像を高速に生成するボクセル追跡法”，信学論，D-II，J72-D-II，9，pp. 1332-1340 (1989)
- 2) J. Amanatides and A. Woo: “A Fast Voxel Traversal Algorithm for Ray Tracing”, Eurographics '87, pp. 3-10 (1987)
- 3) J. G. Cleary and G. Wyvill: “Analysis of an Algorithm for Fast Ray Tracing Using Uniform Space Subdivision”, Visual Computer, 4, pp. 65-83 (1988)
- 4) A. Fujimoto, T. Tanaka, and K. Iwata: “ARTS: Accelerated Ray-Tracing System”, IEEE CG&A, 6, 4, pp. 16-26 (1986)

よこやまけんいちろう
横山 健一郎 平成 2 年，岩手大学工学部情報工学科卒業．同年，同大学大学院修士課程に進学し，現在に至る．コンピュータグラフィックスの研究に従事．

ちば のりしげ
千葉 則茂 昭和 59 年，東北大学大学院博士課程修了．同年，同大学助手．昭和 61 年，仙台電波高専助教授．昭和 62 年，岩手大学工学部助教授．平成 3 年，同教授となり，現在に至る．アルゴリズム，コンピュータグラフィックスの研究に従事．工学博士．正会員．

たやま のりお
田山 典男 昭和 43 年，東北大学大学院修士課程修了．同年，日本信号(株)に入社．昭和 47 年，岩手大学工学部講師，現在，同助教授．多値論理，3 次元並列画像処理計算機，コンピュータグラフィックスの研究に従事．工学博士．