2009－CG－134　(12)
2009／2／16

# 奥行き情報を用いた画像合成による特殊相対論的可視化とその応用

鶴岡修平†　　河口洋一郎††

物体が相対論的大きさを持つ場合，すなわち，光の進む距離に対して十分に大きい天文学的なサイズの物体を可視化する場合，遠くの領域には過去が見えるという時間遅れの現象が生じる．また，見ている形状が運動している場合その形は歪んで観察される．本研究では，2 次元画像とその奥行き情報を用い，それらを合成することによって，時間遅れを再現する手法を提案する．本手法は実装がごく簡単であり，既存の可視化手法と共に用いる事も可能である．また大容量のメモリを搭載した GPU を用いたリアルタイムレンダリングが可能である．

# Special Relativistic Visualization using Depth Information and Its Application

SHUHEI TSURUOKA†　YOICHIRO KAWAGUCHI††

When visualizing astronomical-sized objects, because they are too large in relation to the speed of light, the phenomenon of seeing the past in distant space will occur when observing them. For example, when we stand at the edge of Galaxy and see it, the shape of it will be distorted because the speed of light is too slow. When the objects move at high speed, the distortion is very large. This paper proposes a method to produce time difference visualization, using two-dimensional image series and deep information based on a time series. The method is extremely simple, has the characteristic of being easy to implement, and can be effectively used with existing visualization techniques. The proposed method requires a massive amount of memory, but by using high-speed and high-capacity memory, animations can be generated sufficiently in real time.

## 1. はじめに

天文学的なサイズの物体を観察した場合，光の進む距離に対して物体の大きさが大きい場合，遠くの領域に過去が見えるという現象が生じる．例えば，銀河系の端に立って，その反対側を眺めたら，10 万年昔の姿が見えるだろう(Fig.1)．

また，物体が運動している場合には，その事により，物体の形自体も歪む．運動が早ければ早いほど，その歪みも顕著になる．

特殊相対論的現象を再現するために，従来から、レイトレーシングを拡張した手法が提案されている [Hsiung et al. 1989][Weiskopf 2000]．しかし，この手法では，計算量が多いといった，古典的なレイトレーシングと同様の問題がある．

Time-buffer メソッド[Hsiung et al. 1990]は Z-buffer 法と似た手法だが，レンダリングアルゴリズム自体を変更しなくてはならないため，既存の可視化技術と組み合わせる手間が生じた．

本研究では，通常の，時間遅れを考慮しないレンダリング画像を用い，その奥行き情報を組み合わせることで，時間遅れを擬似的に再現する手法を提案する．この手法のメリットは，実装がごく簡単であり，既存の手法と共に用いることが出来るという点である．また，合成の作業のコストは，形状の複雑さに関係しないという特徴がある．

## 2. 手法

特殊相対論的現象を理解するために，従来からミンコフスキーダイアグラムが使われてきた[Lorenz et al. 1952]．

この図は，空間を 1 次元で表現し，時間軸と併せて，2 次元のグラフとして表現する．時間軸の $\tau$ とは，$\tau = ct$ である(c は光速)．

このダイアグラムにおいて，一般的な可視化について考える。通常の相対論的で無い可視化とは，時間軸を合わせた 4 次元時空間を，時間軸に垂直な平面で切断した面であると言うことが出来る (Fig.2)．

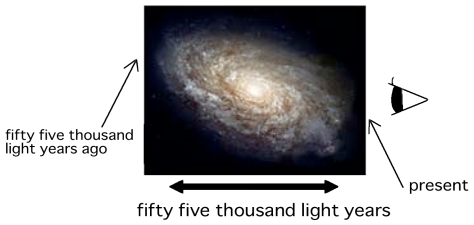


図 1：NGC4414 は，直径約 5,6000 光年である．
Figure 1: NGC 4414 is about 55 thousand light years in diameter (Photo credit: Hubble Space Telescope NASA / ESA [NASA].)

† 東京大学大学院 学際情報学府
The University of Tokyo, Graduate School of Interdisciplinary Information Studies.
†† 東京大学
The University of Tokyo

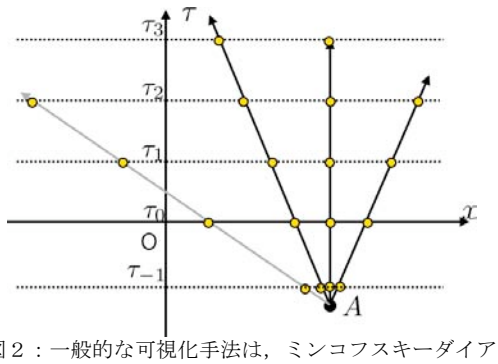


図２：一般的な可視化手法は，ミンコフスキーダイアグラム上で，時刻一定の線（波線）で表される．

Figure 2: A conceptual diagram of a conventional method in a Minkowski diagram. Dotted line is constant τ.

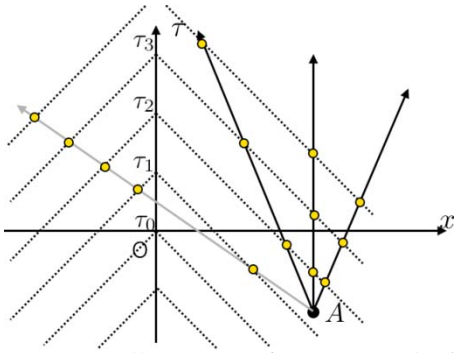


図 3: 特殊相対論的には，レンダリング画像は時間軸に 45 度傾いた線で表される．

Figure 3: A conceptual diagram of a time-space visualization method in a Minkowski diagram. (along dotted line).

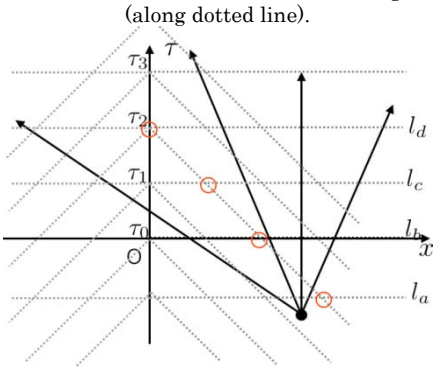


図４：従来の可視化画像によって時間遅れを再現するには，赤丸部分を求め，合成すればよい．

Figure 4: A time-space ray-tracing diagram approximated by combining conventional ray tracing images. This method correct red circles to constitute approximate visual image

対して，時間遅れを含む可視化とは，時間軸に対して 45 度傾いた平面で切断された面を可視化する事であると考えることが出来る (Fig.3)．　ここで，観察者は，τ＝０の線上にいるので，時間軸に垂直な線上での，時間軸からの距離は，時刻が一定の状況における奥行きと同値である．この事から，通常の固定時間の画像と奥行き情報を適切に合成することで，特殊相対論的可視化を再現することが可能であることがわかる(Fig. 4)．

本手法の特徴は，アルゴリズムがきわめて単純であること，既存の手法と併用することが可能であることである．

提案する手法は，4 次元時空間の時間遅れの現象を，2 次元の画像とそれらの奥行き情報をもとに合成する手法である．時間軸におけるサンプリングの間隔（本研究では，時間解像度と呼ぶ）が細かいほど，より正確な画像を得られる(Fig. 5)．この時間解像度がある程度大きい場合，ジャギーやモアレのパターンが現れる．

本手法では，奥行き情報は画像データとして蓄積されるため，視点から見えない位置にある奥行き情報は欠落する．この点で，正確な結果を得ることが出来ない．

また，本手法は，レイトレーシングなどの手法と比較して，大量のメモリを必要とする．大容量のメモリを搭載した GPU を用いることで，リアルタイムで映像を生成することが可能である．

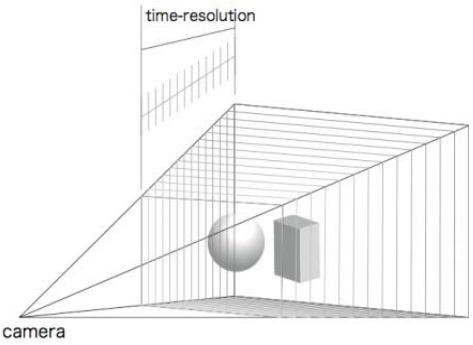


図５：空間のサイズと時間解像度を決定する．

Figure 5:Determine the size of the time-space, and how many steps are required for light to pass through.

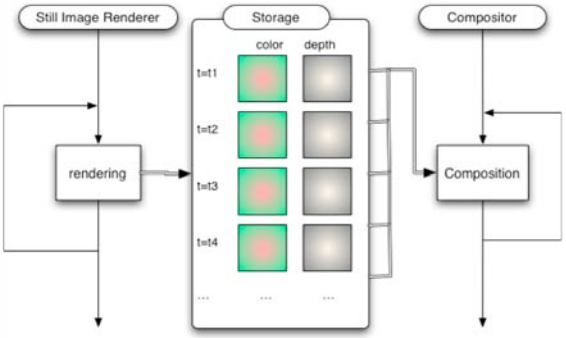


図６：本手法のフローチャート

Figure 6:The flowchart

## 3. 実装

本手法のフローチャートを Fig. 6 に示す．

本手法は，大きく分けて，以下の３つに区分される．

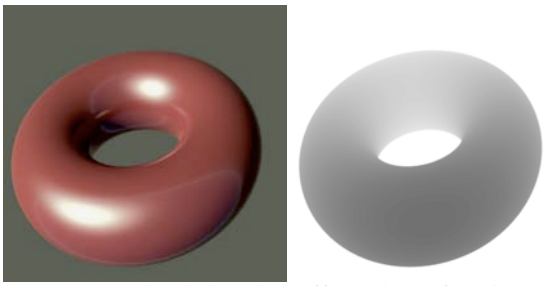

図７：カラー画像（左）と奥行き情報（右・白色領域が最も遠く，黒い領域が最も近い領域）．
Figure 7:Color Image (left) and depth information (right). In depth information, white area is farthest, and black area is nearest from camera.

- 固定時間におけるレンダリング
- データの蓄積
- データの合成

本研究では，実装に，これらの全ての過程において GPU を用いた．それぞれの過程は，区分可能であり，それぞれを別々のハードウェア，ソフトウェアにて実装する事が可能である．また、レイトレーシング[Witted, 1980]や Z-バッファ法においては，全く追加のコストをかけずに奥行き情報を生成する事が可能であり，既存のレンダリングソフトウェアと併用する事が可能である．

## 3.1. 固定時間画像のレンダリング

始めに，通常の固定時間のレンダリング画像を生成する．それぞれの時刻については，t, t+Δt, t+2Δt, t+3Δt… とする．

表１：画像解像度と必要メモリ量
Table 1: Image Resolution and required memory size.

| Resolution | VGA | XGA | FullHD |
|---|---|---|---|
| Width | 640 | 1024 | 1920 |
| Height | 480 | 768 | 1080 |
| Memory Size (time-resolution 256) | 600 MB | 153 6MB | 4.0 GB |
| (time-resolution 512) | 1200 MB | 3072 MB | 8.0 GB |

表２：画像サイズと fps.(時間解像度は 256)
Table 2: Image size and frames per second (fps) at execution.

| Image Size | Memory (MB) | Fps (frame/second) |
|---|---|---|
| 400*400 | 312.5 | 51.3 |
| 500*500 | 488.3 | 51.3 |
| 600*600 | 703.1 | 3.73 |
| 700*700 | 957.0 | 1.95 |

この際に，空間の大きさと光速について考慮する必要がある．光速は，c=299,792,458 m/s であるが，ここにおいては簡単のため，扱う空間の大きさを決定し，そこから，その空間をどれだけのタイムステップで光が貫通するかを決定する．ここから自分の扱う空間における光の速度を求めることが出来る．本手法ではこのタイムステップの数を，”時間解像度”と呼ぶ．１フレームの最終結果を得るためには，この時間解像度と同じ数の画像と奥行き情報を要する（Fig. 7）．

## 3.2. データの蓄積

画像データと奥行き情報はメモリに蓄積される．それぞれのデータは 2 次元画像データとして蓄積される．本研究では，OpenGL を用い，画像データは RGBA のカラーバッファを，奥行き情報はデプスバッファの内容をそのまま用いた．それらの画像は，テクスチャとして保存され，Cg（C for graphics）によるシェーダプログラムから参照される．画像データのピクセルフォーマットは 32bit（RGBA 4bit ずつ），奥行き情報は 32bit （float)を用いた．本手法では，大量のメモリを必要とするが，画像サイズと必要メモリ量を表1に示す．必要メモリ量が GPU の VRAM 上に収まる場合に限り，ハードウェアアクセラレーションが有効になり，効率的な処理が可能になる．現在では 1G 以上のメモリを搭載した GPU も一般的になり，XGA 程度の解像度でも十分にリアルタイムで生成可能である．

## 3.3. 時空間の合成

時間解像度の数と同数の画像データが蓄積された後に合成が行われる．それらのデータは画像データであるので，レンダリングの際の用いたカメラのパラメータを元に，画像上の奥行き情報から，実際の空間内の座標，あるいは，実際の空間内における奥行き情報を復元する必要がある．カメラの投影行列の逆行列を用いる事で，画像上の位置と奥行き情報から，座標が求まる．

以下の説明は，OpenGL のデプスバッファを用いている場合について考える．

投影法が平行投影か斜投影の場合は，奥行き情報から得られたデプス値（0.0 から 1.0 にクリッピングされているとする）を d とすると，実際の空間における奥行き（カメラからの距離）D は下記式で表される．

$$D = zNear + (zFar - zNear) * d \qquad (1)$$

（zNear,zFar はそれぞれ，カメラから近い面と遠い面のクリッピング面までの距離）

また，透視投影の場合，ある画素(u, v)のデプス値が d だとすると，実際の空間における座標値 P は，下記，式（2）によって表される．ここで，u, v は，0 から 1 にクリッピングされたスクリーン上の座標値であり，また，d も 0 から 1 にクリッピングされているとする．これらを正規化デバイス座標系に変換し，-1 から 1 の範囲に直した後，投影行列の逆行列を掛けることで，眼点座標系に変換する．眼点座標系では，カメラの位置は原点であるので，眼点座標系における位置の絶対値が，実際の奥行き D である．

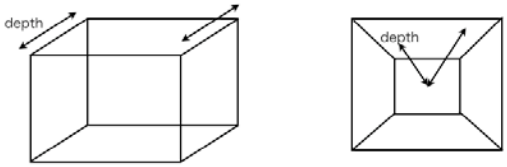


図８：斜投影（左）と透視投影（右）
Fig 8: Oblique projection (left) and perspective projection (right).

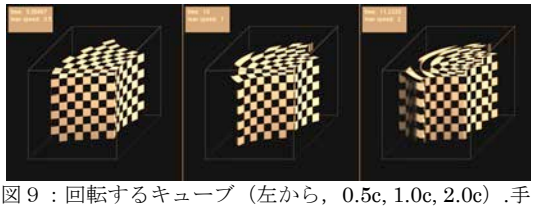

図９：回転するキューブ（左から，0.5c, 1.0c, 2.0c）.手前の面が右から左に回転.（斜投影）
Fig 9: A rotating cube. (From the left, the maximum speed of the surface is 0.5c, 1.0c and 2.0c). The front surface rotates from left to right (Oblique projection).

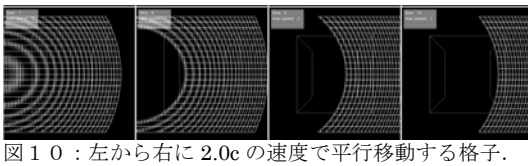

図１０：左から右に 2.0c の速度で平行移動する格子.
Fig 10: A grid that moves from left to right at a speed of 200% of the speed of light in a translational motion. The projection method used is transparent projection.

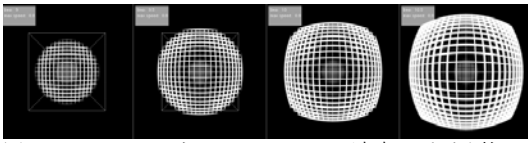

図１１：カメラに向かって 2.0c のス速度で近づく格子（透視投影）
Fig 11: A grid that approaches the camera at a speed of 200% of the speed of light (Perspective projection).

$$P = M_{proj}^{-1}\left\{\begin{pmatrix} -1.0 \\ -1.0 \\ -1.0 \\ 1.0 \end{pmatrix} + 2.0\begin{pmatrix} u \\ v \\ d \\ 0.0 \end{pmatrix}\right\} \qquad (2)$$

この後，アニメーションの時間解像度と光速を考慮し，若い時刻のデータから新しい時刻のデータの順に，画素を重ねてゆくことで時間遅れの画像を合成する．画像の合成は，一つの結果画像を得るのに，時間解像度の数だけ合成を繰り返す．この工程には，フラグメントシェーダを用いた．

この際，ジャギーを防ぐため，それぞれの画素は，ガウス分布にしたがった透明度を与えられて合成される．時刻 t における時間遅れの画像を得る場合，時刻 t-Δt の光は，cΔt だけ遠方にあるはずである．この時，時刻 t-Δt における画像中の，奥行き D であるような画素に対して，以下の式(3)によるアルファ値αを適用して合成した．

$$\alpha = \exp(-\{|\, c\Delta t - D \,|\cdot f\}^2) \qquad (3)$$

（$f$ は，経験的な値を用いる.）

時間解像度が小さい場合，Δt の間隔が大きくなるため，ジャギーを防ぐ事は難しい．また，そもそも物体の運動速度が大きい場合には正確な結果を得る事が難しいため，時間解像度を大きくする必要がある．

## 4. 実験と結果

本章では，生成された結果について述べる．今回は，OpenGL2.0，Cg（C for graphics）を用いた．また，ハードウェアは，GeForce 8800 Ultra (786 MB) を用いた．このメモリ量は，時間解像度 256，あるいは 512 の時，400x400 の画像サイズにおいて，十分なメモリ量を満たし，リアルタイムの映像生成が可能である．

下の例では，物体の回転を扱っているが，通常，回転の速度は，角速度などを用いて表すが，ここでは角速度の変わりに，物体の回転中心から最も遠い位置の速度の大きさで代用する．簡単のためこの距離は 1 とする．また以下では，物理的には起こり得ないが，形状の変化を強調するため，超光速（速度が 1.0c を超える）についての見え方も扱っている．

### 4.1. 投影手法

形状の見た目は，投影法によって顕著に変わる．斜投影は，作図表現において良く使われる投影手法である．斜投影は，人の目が通常見る見え方と比べると顕著に違うために，映像表現にはあまり使われないが，視点からの奥行き情報の差を観察しやすいため，本研究における時間遅れの画像を得る場合には，透視投影や平行投影に比べて，時間遅れを観察しやすい（Fig8）．透視投影は，奥行きが視点から球面状に分布するため，形状の変化が球面状に観察される Fig.10 ，Fig.11 は，運動する格子である．形状の歪みの変化は、球面状のパターンを形成している。

### 4.2. 一般的なポリゴモデルの見え方

本手法においては，形状の複雑さは，固定時間のレンダリングの工程にのみ影響するため，画像の合成過程における計算負荷は変化しない．このため，固定時間のレンダリングが可能な範囲においては，どんなに複雑なシーンでも一定時間での合成が可能である．

Fig.12 and Fig.13.に，大量のポリゴンを持つ形状での回転の様子を示す．数万～数十万ポリゴンの形状でもリアルタイムで映像を生成することが可能であった．ここにおける計算効率とは，固定時刻でのレンダリングにおける計算コストと，一定サイズの画像データを合成するコストであり，画像

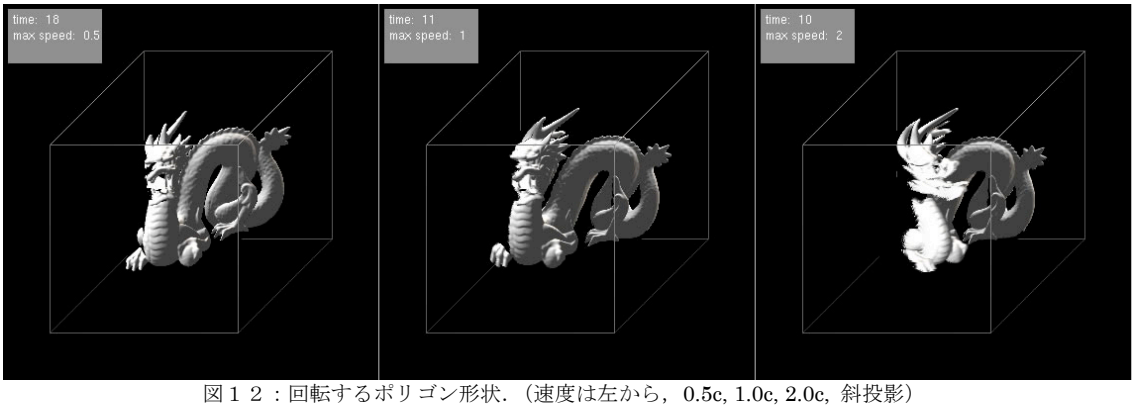


図１２：回転するポリゴン形状．（速度は左から，0.5c, 1.0c, 2.0c，斜投影）

Figure 12: Polygonal shape rotating along the y-axis. From the top, the maximum speed is0.5c, 1.0c and 2.0c (Oblique projection).

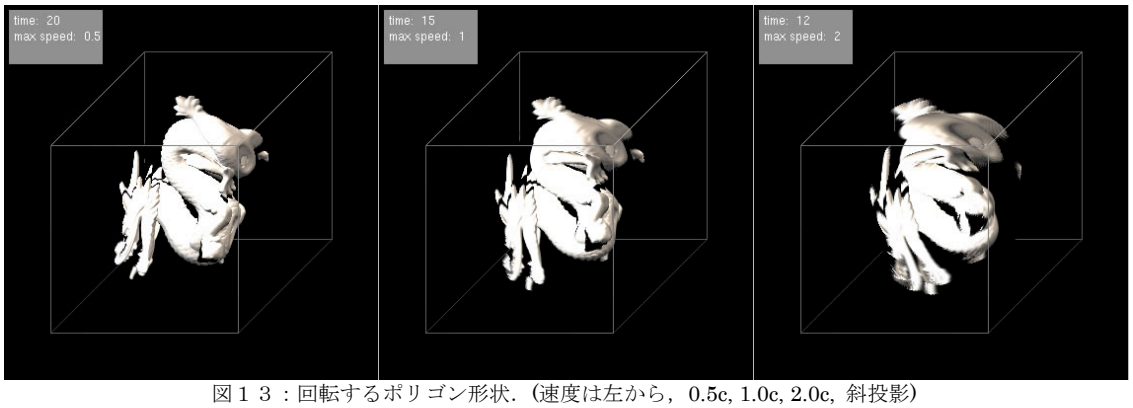


図１３：回転するポリゴン形状．(速度は左から，0.5c, 1.0c, 2.0c，斜投影)

Figure 13: Polygonal shape rotating along the x-axis. From the top, the maximum speed is 0.5c, 1.0c, and 2.0c (Oblique projection).

データの解像度が一定である限り，合成にかかる計算コストは変わらない．OpenGL を用いたアニメーションの生成の fps を表２に示す．GPU のメモリに載る場合，リアルタイムアニメーションが可能であるが，それ以上のメモリ量を要する場合，合成時間が大幅に増える．

## 4.3. 他手法との併用

本手法は，画像データと奥行き情報が生成出来る様々な手法と併用可能である．その例として，仮想的な反射を考慮した回転の様子を Fig.15 と Fig.16.に示す．OpenGL の環境マッピングを用いて，テクスチャ画像を反射させている．実際には，特殊相対論的な反射では，反射する前の光線についても時間遅れを考慮しなくてはならないが，本手法では再現する事は出来ない．

また，既存のレンダリング手法のみならず，レンジファインダーなど，距離を計測出来る機器を用いることで，実世界の奥行き画像を用いて，時間遅れを含む映像を生成する事も可能である．

## 5. 課題

本手法は，固定時間のレンダリングにおいて，カメラから見える範囲の奥行き情報しか用いる事が出来ず，カメラから見えない部分の情報は欠落している．従ってこの欠落した部分において，特殊相対論的には本来見えるはずの部分が見えなくなってしまい，4 次元的な影(Fig.15)となって，黒い領域が生じてしまう．これは，本来のカメラとは違う方向からの画像情報と奥行き情報を追加で用いることで，有る程度影を防ぐ事が可能である．また，デプスピーリングを用いて，奥行き情報を複数枚用意する方手法を用いる事で，完全な奥行き情報を構築出来るため，原理的には影を完全に防ぐ事が出来るが，一般の場合において，有限回のデプスピーリングで奥行き情報を構築する事は出来ないため，より大量のメモリが必要となる．

また，画像データを圧縮し，より大量のデータを扱うことで，より高精度な画像の生成が可能となるが，その際の，データの圧縮，伸張における計算コストが増えるため，効率的なデータの蓄積手法が望まれる．

## 6. まとめ

本研究では，固定時間のレンダリングにおける画像データと奥行き情報の合成によって，時間遅れを含む映像を生成する手法を提案した．本手法は，実装がきわめて単純で，既存の手法とも併用出来る．

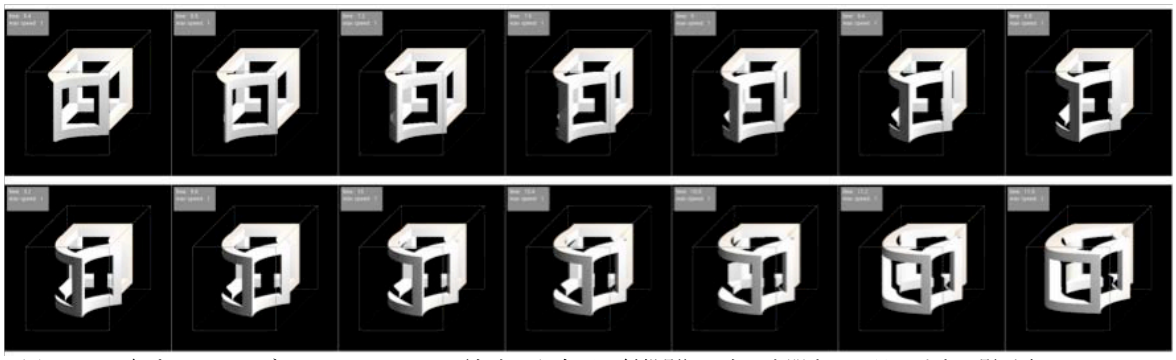

図１４：回転するキューブのアニメーション（左上から右下，斜投影）４次元空間上での見かけ上の影が生じている．
Figure 14: Animation of wire cube rotating with Y-axis. (From left to right, top to bottom, oblique projection). Four-dimensional shadows appear.

図 15：擬似的な反射を考慮した，回転する形状．
Fig 15: Rotating object with Y-axis taking into account hypothetical reflection.

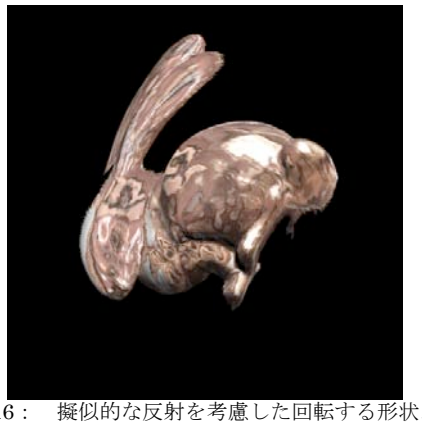

図 16：　擬似的な反射を考慮した回転する形状．
Fig 16: Rotating object with Z-axis taking into account hypothetical reflection.

手法の欠点は，大量のメモリを要する事であるが，高速で大容量なメモリを用いる事で，画像サイズと時間解像度によっては，リアルタイムでの映像生成が可能である．

## 7. 謝辞

形状データには，Stanford Computer Graphics Lab の提供する形状データを用いた．

This research is supported by CREST of JST (Japan Science and Technology).

## 参考文献

**N**ational **A**eronautics and **S**pace **A**dministration (NASA) homepage, http://www.nasa.gov/

H.A.LORENZ, A.EINSTEIN, H.M.H. 1952. *The Principle of Relativity*. Dover, New York.

Turner Whitted, 1980, An improved illumination model for shaded display. In *Communications of the ACM*, Vol23, No6, pp.343-349.

P-K. Hsiung and ANDDUNN, R.H.P. 1989. Visualizing relativistic effects in spacetime. In *Supercomputing '89: Proceedings of the 1989 ACM / IEEE conference on Supercomputing*, ACM, New York, NY, USA, 597–606.

Ping-Kang Hsiung, Robert H. Thibadeau, and M. Wu. T-buffer: Fast visualization of relativistic effects in spacetime. *Computer Graphics*, 24(2): 83-88, Mar. 1990

Hsiung, P.-K.; Thibadeau, R.H.; Cox, C.B.; Dunn, R.H.P.; Wu, M.; Olbrich, P.A., Wide-band relativistic doppler effect visualization. *Visualization '90, Proceedings of the First IEEEConference on*, Issue 23-26 Oct 1990, 83–92, 465–7.

WEISKOPF.D, 2000, Fast visualization of special relativistic effects on geometry and illumination. In *SI3D '90: Proceedings of the 1990 symposium on Interactive 3D graphics*, Snowbird, Utah, United States