

## 3D シーンのための Multi-View 投影を用いた 広角透視投影における歪みの修正

亀田則行<sup>†</sup> 金森由博<sup>‡</sup> 西田友是<sup>†</sup>

広い視野が求められる映画やゲームなどの 3D アプリケーションでは、カメラの視野角を広げた投影が行われる。しかし通常の透視投影では、人間の目で実際の 3D シーンを視認する場合と異なり、物体の歪みが避けられない。

そこで本稿では、通常の透視投影の描画結果に近づけつつ、3D シーンの広角透視投影における歪みを低減する手法を提案する。これを実現するため、各物体について、画面の中央に透視投影し、さらに相似変換（回転、平行移動、一様スケールリング）を適用する。提案法は実装が容易であり、既存のリアルタイムアプリケーションに容易に組み込める。

### Reducing Distortions in Wide-Angle Perspective Views of 3D Scenes Using Multi-View Projection

NORIYUKI KAMEDA<sup>†</sup> YOSHIHIRO KANAMORI<sup>‡</sup>  
TOMOYUKI NISHITA<sup>†</sup>

In 3D applications that require wide field-of-views, projections are performed with widened viewing angles of cameras. However, unlike the way a human being perceives a 3D scene, we cannot avoid distortions in resultant images when using ordinary wide-angle perspective projections.

In this paper, we propose a method that reduces such distortions while minimizing the differences between the results of ordinary perspective projections. This is accomplished by projecting each object to the center of the screen and applying similarity transformation (i.e., rotation, translation and uniform scaling). Our method is easy to implement and easy to integrate into existing real-time applications.

<sup>†</sup> 東京大学  
The University of Tokyo  
<sup>‡</sup> 筑波大学  
University of Tsukuba

### 1. はじめに

広角画像は広い範囲の景観を把握するのに大変有効である。広角画像を得るには広角レンズを用いて撮影する他に、一般的なレンズで撮影した写真を繋ぎ合わせて合成するソフトウェアや、最近登場したパノラマ撮影機能を備えたデジタルカメラが使用でき、広角画像は身近なものになっている。また、広角で撮影すると遠いものがより小さく近いものがより大きく誇張され、迫力が増す、という効果もある。広角の表現は、静止画像だけでなく、映画や主観視点のゲーム(レースゲームやシューティングゲームなど)といった、広い視野が求められる 3D アプリケーションでも有用である。

しかし 3D シーンを広角で投影すると、本来直線的な構造物が曲がったり、物体が引き伸ばされたりといった不具合が生じる。この理由は、既存のあらゆる投影法が平面への投影時に歪みを避けられず、広角にするとその歪みが顕著になるためである[1]。一方、人間の目は広い視野角を持っているが(最大垂直視野角が 125°)、歪みが補正されて知覚されるので、実際に人間の目で 3D シーンを見ると歪みは認識されない。

現在一般的に用いられている投影法である透視投影では、投影前の直線を投影後も直線に保つことができ、画面中央から 30~40°程度の視野角内であれば、物体形状の歪みは少なく、人間の目にとって自然な描画結果を得ることができる[2,3]。しかし、視野角を広げると物体が画面の端に向かって引き伸ばされてしまう。投影の歪みは画面中央から離れるほど大きくなるため、例えば、近年普及が進んでいるワイド画面のアスペクト比(HDTV では 16:9) では特に、画面の両端に近いほど歪みが顕著となる。

本稿では、透視投影による投影結果とできるだけ近づけつつ、人間の認知の仕方でできるだけ近づくよう、3D シーンの広角透視投影による歪みを軽減する手法を提案する。提案法は、壁や床といった平面的で投影面積の大きいものは歪みが認識されにくく、手前にある比較的小さな物体は歪みが目立ちやすいであろう、という推測に基づき、前者には通常の透視投影を適用し、後者には多視点投影(Multi-view 投影)を用いる。すなわち、後者の各物体に対し個別にカメラを向け、そのカメラの画面中央に物体を投影した後、それらを 1 つの画像に合成する。これはちょうど、画家が絵を描くときに、物体ごとに鉛筆などで寸法を測りながら描く行為に相当する。実現方法としては、透視投影によって直線を直線のまま保ちつつ、物体の形が歪まないよう相似変換（回転、一様スケールリング、平行移動）を適用する。提案法はシェーダ言語を用いて容易に実装でき、座標変換に修正を加えるのみなので、既存のリアルタイムレンダリングのアプリケーションに容易に組み込める。

## 2. 関連研究

コンピュータグラフィックスの分野において広角画像の歪みを低減するための試みは、Zorin と Barr によって初めて行われた[1]。彼らは、投影による歪みを2つの誤差関数で評価した。すなわち、(1)直線が直線のまま保たれるか(投影された曲線の曲率がゼロに近い)、(2)投影が相似変換に近くなるか(等角性を保つか)、について考え、1つの関数による大域的な投影では、この両者を同時に満足する投影は存在しないことを示した。彼らはさらに、透視投影とステレオ投影とを線形補間することで、両者のトレードオフを図った。彼らの手法は3次元物体の投影だけでなく、後処理として画像を補正することも可能である。しかしながら、彼らの手法は(1)と(2)の両方を満たすことはできないので、投影される個々の物体に注目すると、ある程度歪みが残っていることが観察される。

広角画像に写った個々の被写体に着目して、選択的に目立つ歪みを低減する手法がいくつか提案されている。Zelnik-Manor [4]らは、画像から前景物体を切りだして遠景と分離し、前景・遠景とに別々の投影法を適用して歪みを低減した。前景物体には、各物体に個別のカメラを正対させて画面の中央に投影したような歪みのない投影像を作る(Multi-view 投影)。遠景には、壁などの継ぎ目で投影面を分け、複数の投影面に分けて投影することで歪みを低減する(Multi-plane 投影)。最終的に前景・遠景を1つの画像に再度合成することで、結果画像が得られる。また、画像を分離せず、ユーザが歪みを低減したい部分を指定した制約を満たすよう、画像を変形する手法も提案されている[5,6]。これらはすべて静止画が対象で、手作業やユーザ入力が必要な上、計算時間がかかるので、3Dシーンのリアルタイムレンダリングには不向きである。

提案法は、Zelnik-Manor ら[4]の Multi-view 投影と同様に、物体ごとに個別のカメラで撮影したような投影像を1枚の画像に合成する。

提案法は、個々の被写体の歪みを低減しつつ、3Dシーンの広角透視投影を実現する初めての手法である。提案法と同様に、多視点で撮影した画像を1枚の画像に合成する手法は、非写実的レンダリング(Non-Photorealistic Rendering; NPR)の分野において多数提案されてきた。ただしその目的は、絵画的表現を模倣すること[7,8]や、多くの視覚的情報を1つの画像に凝縮すること[9]などにあり、投影によって直線が曲がったり物体が変形したりすることは許容される。詳しくは調査論文[10]を参照されたい。

## 3. 提案手法

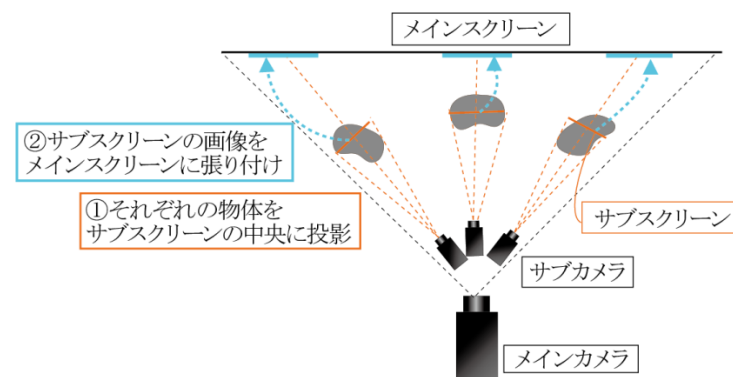


図1 提案手法の概念図(上から見た図)。

先行研究は2Dの画像を入力としているのに対し、本稿では、3Dシーンを対象として透視投影における歪みを修正する手法を提案する。そのためのアイデアとして、3DシーンにZelnik-Manor ら[4]と同様な Multi-View 投影を適用する。すなわち、物体1つにつき1つのカメラ(サブカメラ)を仮想的に用意し、それぞれの物体をサブカメラに対応する、サブカメラの注視方向に垂直なスクリーン(サブスクリーン)中央に投影する。これによりサブスクリーン上には透視投影による投影形状の歪みが解消された物体の投影像が得られる。最後に、これらのサブスクリーンに投影された歪みのとれた画像を、1つのカメラ(メインカメラ)から見えるスクリーン(メインスクリーン)上に合成することで、全体として、歪みを低減する(図1)。以上のアイデアを実現させるための具体的なアルゴリズムは以下である。なお、サブカメラ、サブスクリーンは概念上のものでアルゴリズム上では使用せず、物体をメインスクリーン中央に平行移動することに対応させる。さらに相似変換(回転、平行移動、一様スケールリング)を行うことで、物体の向き、位置、大きさを通常の透視投影の結果に近づける。

- (I) 物体の向きを通常の透視投影の結果に近づけるための、物体の回転。
- (II) 視点座標系においてメインスクリーンの中央に物体を平行移動(サブカメラを設置することに対応)し、物体をメインスクリーンの中央に透視投影。
- (III) 物体の大きさを通常の透視投影の結果に近づけるための、物体の一様なスケールリング。
- (IV) メインスクリーン中央に配置された物体をもとの位置に戻すためのスクリーン上での平行移動。

以下、(I)～(IV)それぞれについて説明する。

### 3.1 物体の回転

変換をする際に、スクリーン中央への平行移動を行うことで、視点から見える物体の角度が異なってしまう(図 2). まず、通常の透視投影の場合の物体の向きと比べて物体の角度が合うよう回転を施す。

物体の重心(全頂点の座標の平均で近似)を  $\mathbf{G}$ ,  $\mathbf{G}$  から視点に向かう単位ベクトルを  $\mathbf{v}_g$ , 注視方向を表す単位方向ベクトルを  $\mathbf{v}_{gaze}$  とする. ここで  $\mathbf{v}_g$  が  $-\mathbf{v}_{gaze}$  に重なるように回転する回転行列を  $\mathbf{R}$  とすると(図 3),

$$\mathbf{v}'_e = \text{Trans}(\mathbf{G}) \mathbf{R} \text{Trans}(-\mathbf{G}) \mathbf{v}_e \quad (1)$$

となる. ここで  $\mathbf{v}_e$  は視点座標系における変換前の物体の頂点座標,  $\mathbf{v}'_e$  は視点座標系における変換後の物体の頂点座標,  $\text{Trans}(\mathbf{x})$  は平行移動ベクトル  $\mathbf{x}$  だけ平行移動する行列を表す.

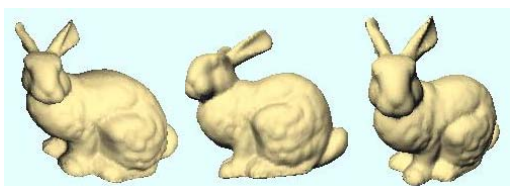


図 2 回転の有無による投影像の見え方の比較. (左)通常の透視投影. (中)回転なし. (右)回転あり. 回転を加えることで、通常の透視投影の見え方に近づけることができる.

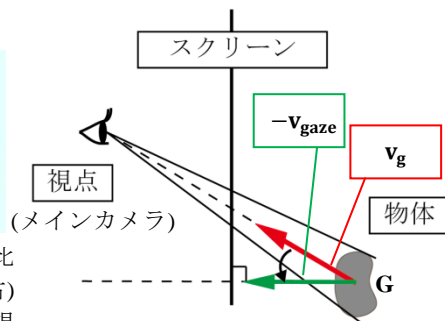


図 3 透視投影の見え方と合うように物体を回転.

### 3.2 スクリーン中央への透視投影

物体をスクリーン中央に配置するため、視点座標系において平行移動する(図 4). これは、サブカメラを設置し、サブスクリーン中央に物体を表示することに対応する。

注視方向ベクトル  $\mathbf{v}_{gaze}$  に沿った視線上に重心  $\mathbf{G}$  を正射影した点を  $\mathbf{C}$  とする. 平行移動量  $\mathbf{v}_{GC}$  は以下のように書ける.

$$\mathbf{v}_{GC} = \mathbf{C} - \mathbf{G} = (\mathbf{G} \cdot \mathbf{v}_{gaze}) \mathbf{v}_{gaze} - \mathbf{G} \quad (2)$$

さらに、 $\mathbf{G}$  から  $\mathbf{C}$  へ物体を平行移動する.

$$\mathbf{v}''_e = \text{Trans}(\mathbf{v}_{GC}) \mathbf{v}'_e \quad (3)$$

ここで  $\mathbf{v}''_e$  は視点座標系における変換後の物体の頂点座標である.

物体をスクリーン中央に平行移動後、スクリーンに透視投影する. これ以降の座標変換は、スクリーン平面上で行う.

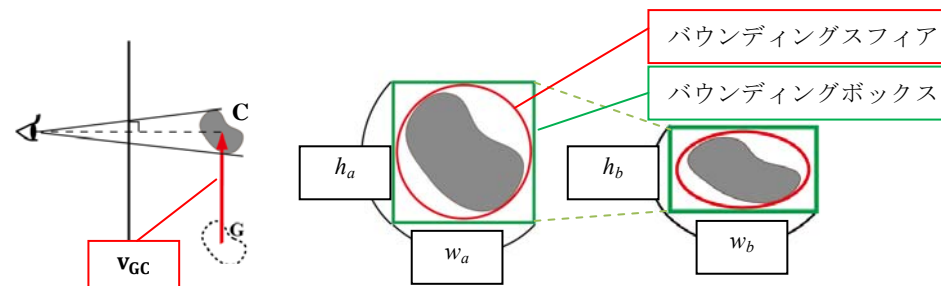


図 4 スクリーン中央への透視投影.

図 5 スクリーン上での投影像の違い. (I)(II)による変換後の投影像(左)と通常の透視投影像(右).

### 3.3 スクリーン上での一様スケーリング

物体をスクリーン中央に配置した際に歪みが修正されることに伴い、物体のサイズが通常の透視投影時と異なってしまう. これを修正するため、一様なスケーリングを施す.

視点座標系で物体のバウンディングスフィアを計算し、そのバウンディングスフィアに対するスクリーン上でのバウンディングボックスを計算する(図 5). バウンディングスフィアの半径を  $R$ , 視点座標系における中心座標を  $(x, y, z)$  とすると、スクリーン平面上でのバウンディングボックスの 4 頂点の座標  $(b_{x,screen}, b_{y,screen})$  は、次のようになる[11].

$$b_{x,screen} = \frac{z_{screen}(xz \pm R\sqrt{x^2 + z^2 - R^2})}{z^2 - R^2}, \quad b_{y,screen} = \frac{z_{screen}(yz \pm R\sqrt{y^2 + z^2 - R^2})}{z^2 - R^2} \quad (4)$$

(I)(II)(III)による変換後の透視投影、通常の透視投影のバウンディングボックスの大きさをそれぞれ  $w_a \times h_a$ ,  $w_b \times h_b$  とする(図 5). ただし(I)(II)(III)による変換後のバウンディングスフィアの投影像は真円であるため、 $w_a = h_a$  である. スケーリング変換は

$$\mathbf{v}'_s = \text{Scale}(\alpha) \mathbf{v}_s \quad (5)$$

と書ける. ここで  $\mathbf{v}_s$  は変換前のスクリーン平面上の物体の頂点座標,  $\mathbf{v}'_s$  はスクリーン上の変換後の物体の頂点座標である. ただし倍率  $\alpha$  は

- (i)  $\alpha_{ave} = (w_b + h_b) / (w_a + h_a)$
- (ii)  $\alpha_{max} = \max\{w_b, h_b\} / w_a$
- (iii)  $\alpha_{min} = \min\{w_b, h_b\} / w_a$

について実験した. 結果を 4 節に示す.

### 3.4 スクリーン上での平行移動

最後に、スクリーン上での物体の位置が変化しないように、通常の透視投影結果の位置に物体を戻すため、スクリーン平面上において平行移動する。

この平行移動について、以下の2通りの実験を行った。

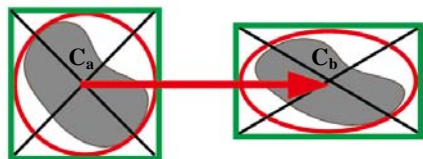


図 6 バウンディングボックスの中心を一致させる平行移動. (I)~(III)による変換後の投影像(左)と通常の透視投影像(右).

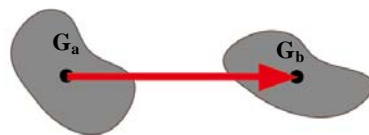


図 7 物体の重心を一致させる平行移動. (I)~(III)による変換後の投影像(左)と通常の透視投影像(右).

(i) バウンディングボックスの中心を一致させる

変換後のスクリーン平面上のバウンディングボックスの中心と変換前のバウンディングボックスの中心をそれぞれ  $C_a$ ,  $C_b$  とし,  $C_a$ ,  $C_b$  が重なるように平行移動する(図 6). (4)式を用いると,

$$C_a = (0, 0), \quad C_b = \left( \frac{xZZ_{screen}}{z^2 - R^2}, \frac{yZZ_{screen}}{z^2 - R^2} \right) \quad (6)$$

である. 平行移動変換は以下のように書ける.

$$v_s'' = \text{Trans}(C_b) v_s' \quad (7)$$

ここで  $v_s'$  はスクリーン平面上における変換後の物体の頂点座標である.

(ii) 物体の重心を一致させる

変換後と変換前のスクリーン平面上の物体の重心をそれぞれ  $G_a$ ,  $G_b$  とし,  $G_a$ ,  $G_b$  が重なるように平行移動する(図 7). 平行移動変換は以下のように書ける.

$$v_s'' = \text{Trans}(G_b - G_a) v_s' \quad (8)$$

以上の2通りの方法を用いた実験結果を4節に示す.

以上の手順により, 歪みの修正された, 通常の透視投影結果と近い描画結果を得ることができる.

## 4. 実験結果

提案法をC++, OpenGL, GLSLを用いて実装し, 2.13GHz Intel Core(TM)2 CPU, 2GB RAM, NVIDIA GeForce 7900 GTX GPU を搭載したPC で実験を行った. 以下, 結果画像は特に断りがない限り, 視野角は90°, および画面サイズは800×450である.

表 1 に平行移動および倍率を変化させて行った実験結果を示す. シーン下部の平面(ワイヤフレーム表示)は常に通常の透視投影で描画し, それ以外の前景物体に Multi-view 投影を適用した. それぞれ左下のティーポットにのみバウンディングスフィアとスクリーン上のバウンディングボックスを表示している. 黒いバウンディングボックスは通常の透視投影像の, 赤いバウンディングボックスは提案手法による投影像のバウンディングボックスである.

実験結果からわかるように, 通常の透視投影と比べて, 提案手法は明らかに物体の歪みが修正されている. すなわち, 実空間での観測結果により近い歪みの少ない描画結果を得ることができた.

各列の上中下段の結果より, 3.3 節で述べた, 倍率  $\alpha$  を3つの場合に変化させた結果を比較することができる. 我々は,  $\alpha = \alpha_{min}$  の場合が最も通常の透視投影に近い結果であると考ええる. 透視投影による物体の歪みは「伸びる」ことにより起こるため, バウンディングボックスの長辺は伸ばされた結果によるものである. 従って短辺の方がより実際の観測結果に近い距離を保っていると考えられる. また, 比率  $\alpha$  の計算方法の違いによる描画結果の違いは, 物体が比較的遠くにある右の物体にはさほど大きくは現れず, 左の物体のように近くにある場合には顕著に差が表れることがわかる. これは遠くのもの, 小さなものに比べ, 近くのもの, 大きなものの方が透視投影による歪みが大きく, その結果バウンディングボックスの辺の長短の差が大きくなることに起因する. あまりにも大きな物体や近くの物体を扱うには制限があることも同時に示唆する.

また, 各行の左右2つの結果から, 3.4 節で述べた, 平行移動をバウンディングボックスの中心を基準に行う場合と物体の重心を基準に行う場合を比較することができる. 透視投影による物体の投影像はスクリーンの縁に向かって外方向に引き延ばされるため, バウンディングボックスの中心は物体の重心よりも外側にある場合が多い. 従ってバウンディングボックスの中心を基に平行移動した投影像は, よりスクリーン外側に物体を押し出す結果となった. 結果として物体の重心を基に平行移動を行う方が, 物体間の相対的な距離を保つことがわかった.

以上を踏まえ, 我々は,  $\alpha = \alpha_{min}$  かつ重心を基に平行移動した投影像が最も良い結果であると考え(表 1 の該当のマスを黄色で表示)が, これについてはユーザーテスト



などを行いさらに検証し、適切な方法を決定していく必要がある。

図8に視野角を変化させた際の結果について示す。視野角を $125^\circ$ に広げた場合にも物体の形状を保つことができた。しかし、視野角を広げると歪みは大きくなり、その結果、提案法によりスケールされた物体が非常に大きく投影されてしまう場合があった。

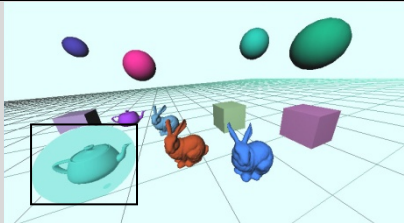
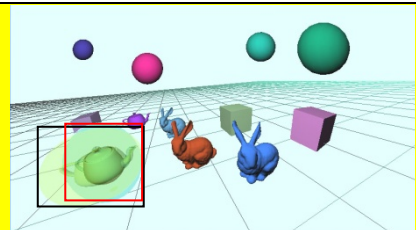
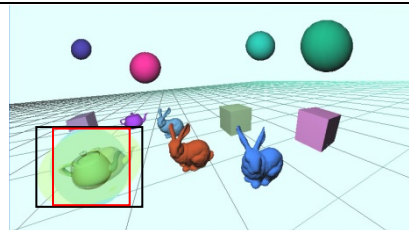
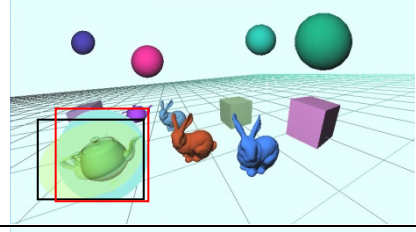
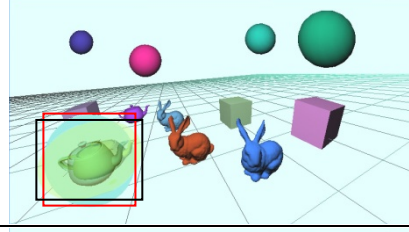
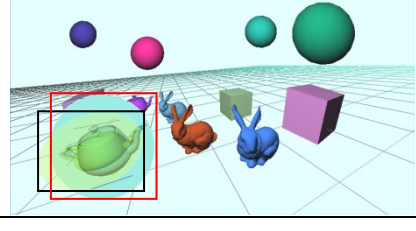
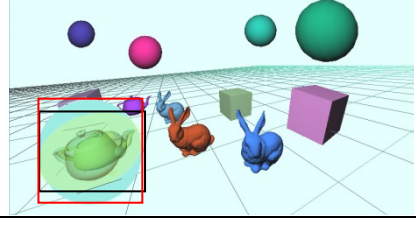
また、図9に示すように、物体間の接触がある場合、一方の物体がもう一方の物体の中にめり込んだり、あるいは接触関係を保たず宙に浮いたりしてしまうなどの問題がみられた。これは、相似変換の際に接触関係を考慮せずに座標変換を行うために起きる問題であると考えられる。

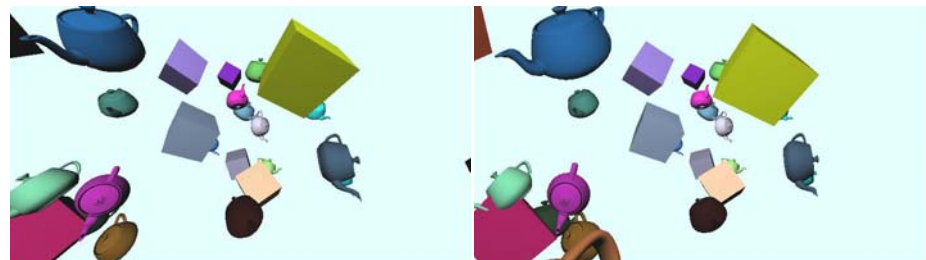
### 5. まとめと今後の課題

本稿では、画面の中央への透視投影と相似変換により、通常の透視投影における歪みを軽減する手法を提案した。提案法により、広い視野角においても物体の歪みは軽減され、より自然な描画結果を得ることができた。

今後の課題としては、物体間の接触がある場合に、相似変換に制約を加え、接触関係が保たれるように改善することが必要である。また、今回は影がある場合を扱っていないため、そのような場合を実装し、影の位置や大きさについても妥当性のあるようなアルゴリズムを提案したい。

表1 倍率 $\alpha$ と平行移動の方法の違いによる投影像の比較(通常の透視投影: 160fps, 提案手法: 159fps)。

通常 の 透視 投影 像		
	重心を一致させる	バウンディングボックスの中心を一致させる
$\alpha$		
$\alpha_{min}$		
$\alpha_{ave}$		
$\alpha_{max}$		



(a) 視野角 90°



(b) 視野角 125°

図8 視野角 90°, 125°の際の投影像の比較. (左)通常の透視投影像. (右)提案手法による投影像.

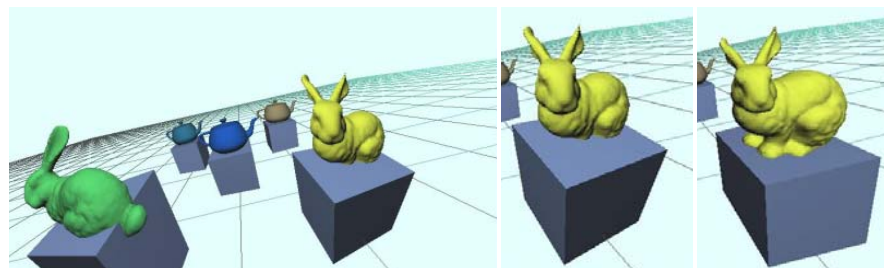


図9 相似変換によって物体がめり込んだ様子. (左)提案法による投影像(全体). (中)提案手法による投影像(拡大). (右)通常の透視投影像(拡大).

### 参考文献

- [1] D. Zorin and A. H. Barr. Correction of geometric perceptual distortions in pictures. In Proc. of SIGGRAPH 1995, pp. 257-264, 1995.
- [2] M. H. Pirenne. Optics, Painting & Photography. Cambridge University Press, 1970.
- [3] D. Vishwanath, A. R. Girshick, and M. S. Banks. Why pictures look right when viewed from the wrong place. Nature Neuroscience 8, 1401-1410 (2005).
- [4] L. Zelnik-Manor, G. Peters, P. Perona. Squaring the Circle in Panoramas. In Tenth IEEE International Conference in Computer Vision, 2005. Vol. 2, pages 1292-1299, 2005.
- [5] R. Carroll, M. Agrawal, and A. Agarwala, Optimizing content-preserving projections for wide-angle images. ACM Trans. Graph., 28(3):1-9, 2009.
- [6] J. Kopf, D. Lischinski, O. Deussen, D. Cohen-Or and M. F. Cohen, Locally Adapted Projections to Reduce Panorama Distortions. Computer Graphics Forum (Proc. of EGSR 2009), Vol. 28, No. 4, pp. 1083-1089, 2009.
- [7] P. Coleman and K. Singh, Ryan: rendering your animation nonlinearly projected. In NPAR 2004, pp. 129-138, 2004.
- [8] 吉田謙一, 高橋成雄, 藤代一成, 岡田真人: 「2次元投影図上の見えの操作に基づいた非透視投影の設計」, 画像電子学会誌, Vol.37, No.4, pp.412-418, 2008.
- [9] S. Takahashi, N. Ohta, H. Nakamura, Y. Takeshima, and I. Fujishiro, Modeling Surperspective Projection of Landscapes for Geographical Guide-Map Generation, Computer Graphics Forum, Vol. 21, No. 3, pp. 259-268, 2002.
- [10] J. Yu, L. McMillan and P. Sturm, Multiperspective modeling, rendering, and imaging. ACM SIGGRAPH Asia 2008 courses, Article No. 50, 2008.
- [11] Y. Kanamori, Z. Szego and T. Nishita, GPU-based Fast Ray Casting for a Large Number of Metaballs. Computer Graphics Forum (Proc. Of Eurographics 2008), Vol. 27, No. 3, pp. 351-360, 2008.