# 首振りディスプレイエージェントの目を描画するための計算モデル

# A Computational Expression to Draw Planar Eyes for a Bobblehead Display Agent

西澤 良真　尾関 基行*　岡 夏樹
Ryoma NISHIZAWA, Motoyuki OZEKI, and Natsuki OKA

京都工芸繊維大学
Kyoto Institute of Technology

**Abstract:** A bobblehead display agent is a display (LCD monitor, touch panel, tablet PC, or etc.) attached to a pan/tilt platform or a robot arm. By drawing CG eyes on the display, the display could function as agent's head like that of a humanoid robot with movable eyes. However, the line of sight is not correctly conveyed to a viewer when the viewer looks at the display from an angle. This problem becomes pronounced in the case of planar eyes. To solve the problem, we propose a novel computational expression to draw planer eyes that could correctly point the viewer in the right direction. This paper graphs displacements of black eyes calculated by our method and two existing methods and shows the advantages and disadvantages of our method.

## 1　はじめに

メールやスケジュール，ソーシャルネットワーク，ニュース，音楽，映画など，様々な情報をネットワークに繋がれたコンピュータを介してやりとりする時代になった．今後もネットワーク家電やスマートグリッドなどの普及により，家庭内にも情報ネットワークが張り巡らされていくと予想される．現在，これらのサービスを享受・制御するためのメディアハブとなっているのは PC やタブレット端末，スマートフォンなどであるが，ネットワークサービスが複雑になるに伴って，情報機器を使い慣れた若者であってもこれらを使いこなすことは難しくなってくる．

そこで期待されているのが，ネットワークサービスやインターネット情報とユーザとの間を仲介してくれるエージェントの登場である．ユーザはエージェントとのインタラクションを介して，メールやニュースなどのインターネット情報をやりとりしたり，ネット家電やスマートグリッドの情報を受け取って操作する．エージェントは単なる受け身のユーザインタフェースではなく，ユーザの様子を認識したり行動パターンを学習することで，その時々にユーザに役立つ情報を提供してくれる．

*連絡先：京都工芸繊維大学 情報工学部門
〒 606-8585 京都市左京区松ヶ崎橋上町
E-mail: ozeki@kit.ac.jp

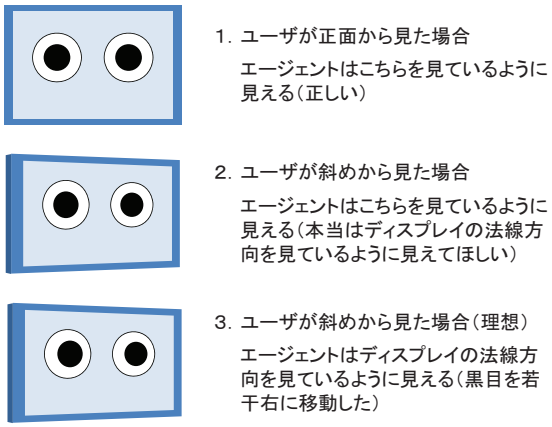



図 1: 視線方向のズレの例

そのようなエージェントの形態として，我々は，タブレット端末などのディスプレイを Pan/Tilt 制御可能な雲台に乗せた「首振りディスプレイエージェント」に着目している．ディスプレイが物理的に動くことで，CG エージェントには難しい実世界への参照・指示が可能となる．更に，ディスプレイ上に顔を描画することで，視線や表情を容易に変えられるロボット頭部が廉価に実現できる．

しかし，ディスプレイが首を振ることによって，ユーザがディスプレイを斜めから見ることになってしまう．

ディスプレイは，本来，ほぼ正面から見ることが前提とされており，3D グラフィックスであってもその前提に従って描画されている（3 次元的にユーザ視点から見える絵は計算できるが，その絵はディスプレイ正面から見ることを前提に描画される）．この問題は，ディスプレイ上に 2 次元的に目を描画する場合，より顕著に現れる．紙などにこちらを見ている目の絵を描いて，それを左右に回転してみるとわかるが，紙を多少回転してみても，その目は常にこちらを見ているように見えるのである（図 1）．

そこで本研究では，ディスプレイ上に 2 次元的に描いた目の視線方向が，ディスプレイを斜めから見たときにも（見かけ上）正しい方向を見ているように描画するための計算モデルを提案する．以降，本稿では，首振りディスプレイの位置づけと問題点を 2 章で述べ，2 次元的に描いた目の視線方向を（見かけ上）正しく見せるためのアプローチを 3 章で述べる．4 章で二つの従来手法を取り上げた後，5 章で提案手法の計算式とその計算結果の例をグラフで示す．

## 2　首振りディスプレイエージェント

### 2.1　エージェントとしての位置づけ

現在 主に扱われているエージェントの形態としては，身体を伴ったロボットやディスプレイ内に描画された CG エージェント，音声のみで存在するエージェント（身体は端末本体）などが挙げられる．これらの中でまず普及が進むと考えられるのは，タブレット端末やスマートフォンの上で動作するエージェントであろう．音声だけであれば既にエージェント化は進みつつあり，CG エージェントを実現するために十分なグラフィックス性能も備えている．近年では，端末に搭載されたカメラを使ってユーザの顔の位置を認識し，ユーザ視点に合わせて CG を動かすことで運動視差を再現するアプリも発表されている [1]．この技術を応用すれば，（ユーザを含む）実世界とのやりとりが可能な CG エージェントも近いうちに登場すると予想される．

しかし，実世界とのインタラクションを考えたとき，フィジカルに動く部分を持たないというのは大きな欠点である．ユーザに何かしら働きかけるのに，画面内での描画では気づかれないことや分かりにくいことも多く，一方，音声は無視しづらいためにユーザに負担をかける．何より，実世界に存在する対象をフィジカルに参照・指示できないことは，ユーザとの直感的なインタラクションを実現するにあたって致命的な制約となる．人は，手腕を使った指示以外にも，視線の向き・頭部の向き・体の向きの情報を受けて巧みにコミュニケーションを行っている．手腕を使ったポインティングははっきりと明示的に行われるものなので，音声やディスプレイ表示でも代用できる．一方，視線・頭部・体の向きは，誰に対して，何について，どういう心的姿勢で話しているのかを暗に示すものであり，身体性による表現でなければ伝達が難しい．

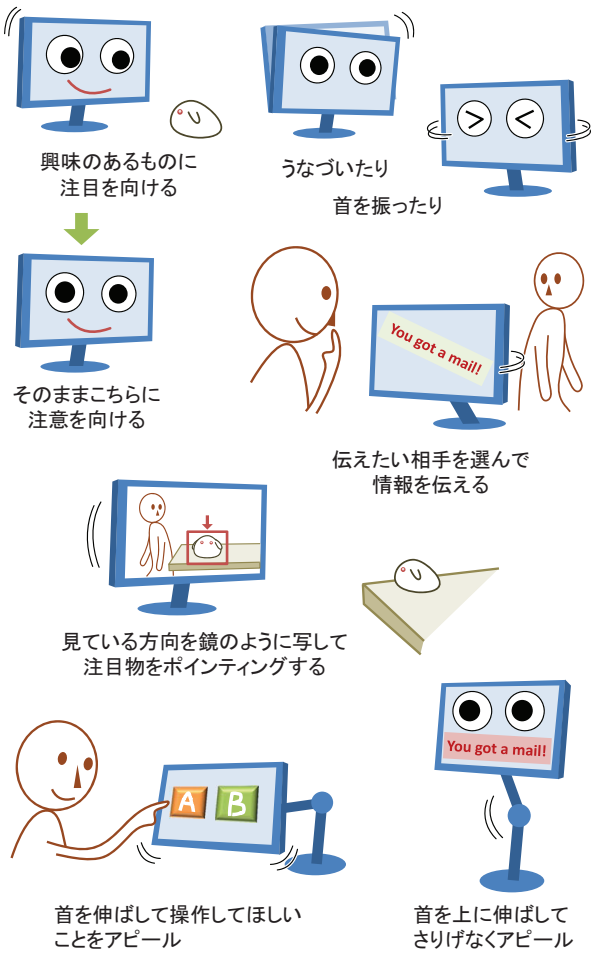



図 2: 首振りディスプレイエージェントで実現可能なインタラクションの例

一方，動く眼球や頭部，手腕を備えたロボットはこの問題を回避しているが，現時点ではコストが高い．動く眼球と頭部を備えたロボットの価格は，足がなくて動けないタイプですら，タブレット端末が 20 台購入できるほどである．将来的には，家や会社内を自由に移動できる人型ロボットが自動車程度の価格で手に入るようになることが望ましいが，それでも，風呂場やトイレといった狭いスペースにはユーザと一緒に入れないし，家族が同時に使いたいと思っても人型ロボットを複数台購入できる予算的・スペース的な余裕のある家庭は多くないだろう．

そこで我々が注目しているエージェント形態が，冒頭でも述べた「首振りディスプレイエージェント」である．

これは，タブレット端末やスマートフォンを Pan/Tilt 制御可能な雲台に乗せたものである．ディスプレイに顔を描画することでディスプレイ本体を頭部とみせかければ，ロボット頭部のように実世界への参照・指示が可能となる．予算が許すのであれば，ロボットアームの先にディスプレイを取り付け，ピクサーの動くランプ（Luxo Jr.）のように，より生き物らしい表現もできる．

首振りディスプレイエージェントが可能とするインタラクションの一例を図 2 に示す．普段持ち歩いているタブレット端末やスマートフォンを雲台やアームに取り付けて使ってもよいし，通常のディスプレイやタッチパネルを組み込んで廉価なロボットの頭部として使ってもよい．コスト的にもスペース的にも，家や会社の各所に置いておくことが可能である．我々は，1 台の人型ロボットが家中を動き回る未来よりも，各所に置かれたエージェントが適材適所の役割を果たす未来のほうがずっと早く訪れると予想している．

## 2.2 要件

我々が「首振りディスプレイエージェント」に求める要件を以下に挙げる．

**要件 (1)** ディスプレイ本体が首を振り，フィジカルな頭部としての役割を果たす

**要件 (2)** ディスプレイ本体（頭部）とディスプレイに描画された目によって，実世界にあるものを参照・指示できる

**要件 (3)** 特別なディスプレイを使わず，タブレット端末など，通常のディスプレイで実現できる

まず，要件 (1) が満たされなければ，ユーザの方向を向いてくれるだけの便利なディスプレイになってしまう．CG エージェントとは異なり，ディスプレイは実世界にフィジカルに存在するものであり，ディスプレイの向いている方向は実世界の誰から見ても一意である．詳しくは後述するが，多視点 3D ディスプレイを用いないかぎり，CG エージェントの目や頭部はこの性質を持たない．ディスプレイがエージェントの頭部と見做されることにより，ディスプレイの向きがエージェントの注意や興味の向いている方向であるとユーザに暗に伝えることができる．実世界とのインタラクションにおいて，CG エージェントに対するフィジカルエージェントの最大の利点はそこにある．

ディスプレイ本体を頭部と見做してもらう（ディスプレイの淵を顔の淵と見做してもらう）ためには，CG の顔がディスプレイの中から覗いているような表現ではなく，ディスプレイの表面に 2 次元的な目や口が貼り付けられたような表現が望ましい．そうすることで，薄いディスプレイを横や斜めから見たときにディスプレイの背後に仮想的な空間を想像しなくてもよく，顔のパーツまで含めて「四角い平面頭のエージェント」として実世界に矛盾なく存在できる．

目を 2 次元的に描画することには，他にも，漫画的な表現・装飾がしやすいという利点がある．ロボットに比べれば手軽に表情を作り出せるものの，表情が変化する様子などを 3D グラフィックスで描画するには手間と技術が必要になる．かなり精巧にアニメーションを作り込まなければ不気味の谷に陥る可能性があり，逆に，顔を精巧にすればするほどディスプレイ本体との一体感が失われ，顔がディスプレイ枠から覗いているような印象をユーザに与える．一方，顔を漫画的に表現できるのであれば，パーツを作ることもアニメーションを作ることも比較的用意であり，不気味の谷に陥る危険性も少ない．また，表情の変化にアニメーションを使わず，表情をパタパタと切り替えても違和感は少ない．そのためにも，目は 2 次元的に描画できることが望ましい．

要件 (3) は，首振りディスプレイエージェントの位置づけとして妥当な要件である．要件 (2) については次節以降で詳しく取り上げる．

## 2.3 問題点

2.1 節で述べたように，ロボットと CG エージェントの中間的存在としてうまく機能しそうな首振りディスプレイエージェントであるが，2 次元的に描かれた “目” に関しては，ディスプレイが首を振る（つまり，ユーザがディスプレイを斜めから見る）ことによって次の二つの問題が生じる：

1. 同時に一つのユーザ視点から見た場合にしか，エージェントの視線方向が正しく示せない
2. 既存の CG の描画方法では，ディスプレイを斜めから見たときに，黒目の位置が意図した方向からズレたり，白目が歪んだりする

前者は，ビデオ会議システムを多人数で使用する際にもよく問題となる [2]．この問題は，裸眼の多視点 3D ディスプレイを使用することで解決されると考えられるが，視点数の多いものは現時点では大変高価であり，将来的にもタブレット端末やスマートフォンに採用されるかどうかわからない．それに対して我々は，首振りディスプレイエージェント専用のインタラクションモデルを導入することで，通常のディスプレイでも複数のユーザとのコミュニケーションを成立させることができると考えているが，これについては別の機会に発表したい．

後者の問題は，前節で挙げた要件 (2) が満たせないということであり，目を 2 次元的に描画するという制約を無くさないかぎり，多視点 3D ディスプレイを用いても解決しない．これは，平面上に描画された目の視線方向を人が認知する際，幾何学的な正しさよりも，白目と黒目の位置関係をより重視して判断するためだと考えられる．

例えば，真正面を向いた目が描画された平面（黒目は白目の左右中央に描かれている）を回転したとき，その目は幾何学的[1]には平面の法線方向を向いている状態になる（後述の従来手法 A）．しかし，平面を回転する角度が浅い場合，それを見た人は「この目はこちらを向いている」と感じる．これは，平面を回転しても，相変わらず黒目が白目の左右中央にあるためである．この場合，描画された目が平面の法線方向に向いているように見せかけるためには，黒目の描画位置を平面を回転した方向に少し寄せる必要がある（この例については既に図 1 に示した）．

同様の問題が，ユーザがディスプレイを斜め方向から見ている状態で，エージェントがユーザの方向を見たときにも生じる．例えば，ユーザが斜め 45° の位置からディスプレイを見ているとすると，エージェントの黒目は 45° だけユーザの方向に移動する[2]．しかし，前述したようにユーザは自分の方向を見ているか否かを「黒目が白目の左右中央に近いか否か」で判断しているため，この場合はユーザのいる位置よりも深い角度（45° 以上の角度）にエージェントの視線が向いているように感じる．よって，この場合には黒目位置を白目の中央付近に寄せる必要がある．

## 3　アプローチ

前述の問題を解決するには,「2 次元的に描かれた目」から「3 次元的な視線方向」への写像を何らかの方法で求める必要がある．アプローチとしては，(1) 実際に 2 次元的に描画された “目” を見たときに人がどの方向を見ていると感じたかを細かく調べ，その分布を近似する関数を求める方法と，(2) 概ね正しいと思われる計算式を予め用意しておき，その中の 1〜2 個のパラメータをユーザに合わせてフィッティングする方法がある．我々は (1) の方法をまず試みたが，2 次元的に描かれた “目” から受ける方向性は人によって異なる可能性が大きいことがわかったため，ユーザへの個別適応が容易である (2) のアプローチをとることにした．

前述した「見かけ上の目のズレ」の修正方向を鑑みると，平面に穴を空け，そこに裏から球形の眼球をはめ込んだ状態を仮定するとうまく説明づけられる．例えば，平面の法線方向に黒目を向けておき，そのまま平面を横に回転すると，黒目の位置が白目の左右中央よりも回転した方向に少し寄る．この見え方は，CG の仮想空間内でディスプレイ面に埋まった眼球を想定し，それを仮想空間内のユーザ視点に置いた仮想カメラで撮影した “見え方” を描画すればほぼ再現できる．この方法は，ユーザの視点に合わせて CG を変化させることで運動視差を再現する既存のシステムでも利用されている（後述の従来手法 B）．

しかし，仮想空間内で 3 次元的に計算された見え方をそのままディスプレイに描画してしまうと，ディスプレイを斜めから見たときに白目が歪んでしまう．これは，そもそも目を斜めから見たときの歪みまで計算に含んだ見え方を，実際にディスプレイを斜めから見ることで二重に歪めてしまうためである．また，平面から飛び出た半球を斜めから見ると満月が欠けたような形に見えるが，表現したいのはあくまで 2 次元的に描かれた目なので，月のように欠けるのはおかしい．白目は平面上に描かれた円もしくは楕円であり，ユーザがディスプレイをどこから見ていても，ディスプレイに描画する形は円か楕円のままでなければならない．

以降では，まず二つの従来手法を 3 次元の眼球モデルを使って説明し，その後，提案手法について説明する．なお，要点を絞って説明するために以下の簡易化を行うが，これによって本提案の本質的な主張が変わることはない．

- 透視投影ではなく平行投影で近似する．
- 輻輳は考えない（輻輳有りの場合については最後に計算式のみを示す）．
- ディスプレイの回転方向は水平方向のみを考える．
- 計算式では黒目の中心位置と白目の淵の位置のみを求める．

また，次章以降の計算式で使用する記号は以下のとおりである．図 3 に概要を示す．

$\theta$： ユーザとディスプレイを結んだ直線からディスプレイの法線方向までの角度

$\phi$： ディスプレイの法線方向からエージェントの視線方向までの角度

$R$： 眼球の半径

$r$： 眼球の中心とディスプレイ面の距離

なお，$\theta$ と $\phi$ についてはディスプレイの法線方向を 0° とし，ディスプレイから見て反時計回り（左回り）を正とする．

---

[1] 平面に描画された目の 3 次元的な方向性について，そもそも幾何学的に正しいといえる答えはないかもしれないが，ここでは「単純に考えればそうなる」という程度に解釈されたい．

[2] 平面に描かれた目が斜め 45° を向いている状態をどう計算するかも問題であるが，4 章の従来手法 A の考え方が最も基本的と考えられる．

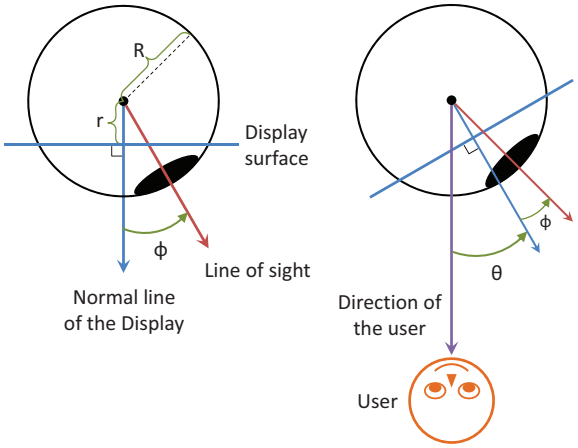



図 3: 計算式で使用するパラメータ

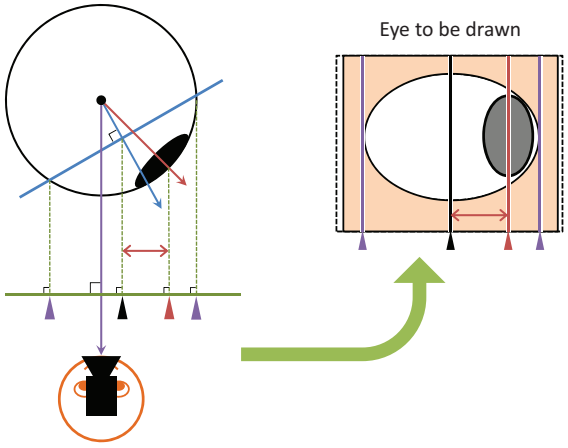



図 5: 従来手法 B

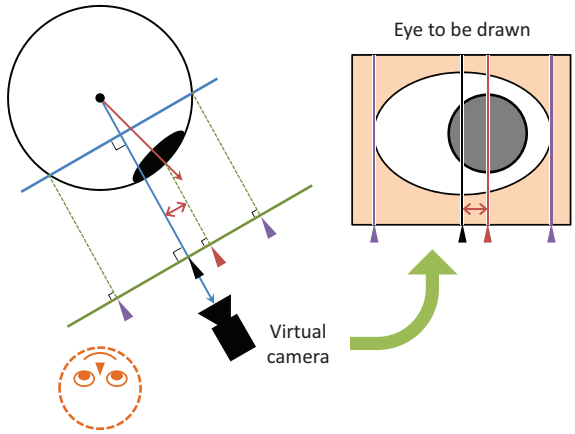



図 4: 従来手法 A

# 4 従来手法

## 4.1 従来手法 A

従来手法 A は，2.3 節で引き合いに出した “目” の計算方法である．ユーザが常にディスプレイを真正面から見ていると仮定して，エージェントの “目” の見え方を計算する．CG エージェントと人がインタラクションするシステムで，運動視差を再現しないものの多くはこの手法を用いていると考えられる [3]．ユーザがディスプレイを正面から見ている場合にはこの手法でも問題は生じない（ディスプレイを正面から見ている場合には従来手法 B も提案手法も計算結果は同じになる）．

図 4 に模式図を示す．ディスプレイ面に眼球が埋め込まれた状態の仮想 3 次元空間を想定し，仮想カメラをディスプレイの正面に置く．視線方向 $\phi$ へ眼球を動かした上で，仮想カメラからの 2 次元的な目の見え方をディスプレイにそのまま描画する．白目中心からの黒目の移動量は以下の式で計算できる．

$$(R - \frac{r}{cos\phi})sin\phi + rtan\phi$$

この式からわかるように，ユーザがディスプレイを見る角度 $\theta$ は計算の中に含まれず，ユーザがディスプレイをどこから見ても同じ “目” を描画する．この考え方を首振りディスプレイエージェントに適用する場合，ディスプレイを回転しても，ディスプレイの法線方向とエージェントの視線方向の関係は変わらないものとして考える．例えば，$\phi = 30°$ で計算した “目” を描画したディスプレイを $-30°$ 回転させると，エージェントはこちらを向いていることになると考える．しかし，2.3 節で例に挙げたとおり，実際にはそのように見えない．

一方，従来手法 A の長所は，ディスプレイを回転させても「2 次元的に描画された目」としての整合性を保つことにある．つまり，ユーザがディスプレイを見る角度 $\theta$ に因らず，ディスプレイに描画される白目の形や直径は同じになる[3]．これはディスプレイ表面に目が描かれていると考えれば正しい．

## 4.2 従来手法 B

従来手法 B は 3 章で引き合いに出した “目” の計算方法である．ユーザの視点（顔の位置など）を検出して運動視差を再現するシステムが用いている方法である[4]．ユーザがディスプレイを見る角度が浅い場合には，この方法でも問題はほとんど露呈しない．

図 5 に模式図を示す．ディスプレイ面に眼球が埋め込まれた状態の仮想 3 次元空間を想定し，仮想カメラ

[3] もちろん，ユーザはその “目” が描画されたディスプレイを斜めから見るので，最終的にユーザの網膜に映る “目” はその分だけ歪んでいる．

[4] 運動視差を再現するシステムはまだ少なく，詳しい情報は得られていないが，インターネット上で動画が見つかったものを確認するかぎりでは従来方法 B が用いられていると考えられる．

を実際のユーザの視点に置く．視線方向 $\phi$ へ眼球を動かした上で，仮想カメラからの 2 次元的な “目” の見え方をディスプレイにそのまま描画する．白目中心からの黒目の移動量は以下の式で計算できる．

$$(R - \frac{r}{cos\phi})sin(\phi + \theta) + rtan\phi cos\theta$$

この式からわかるように，ユーザがディスプレイを見る角度 $\theta$ の値によって黒目の位置が変化し，実際にロボットや人の眼球を見た場合に近い “目” を描画できる．$\theta$ を変えながら数点のキャリブレーションデータを取得し，眼球とディスプレイとの距離を表す r を調整することによって，2 次元的に描画された “目” の計算式を求めることができると考えられる[5]．

一方，従来手法 B の欠点は，ユーザがディスプレイを見る角度が深くなるにつれて白目が歪んでいくことである．従来手法 B では，白目中央から白目の淵までの距離は以下のように計算される．

$$\sqrt{R^2 - r^2}cos\theta \tag{1}$$

従来手法 A の場合，白目中央から白目の淵までの距離は $\sqrt{R^2 - r^2}$ で計算されるが，従来手法 B ではそこに $cos\theta$ が掛けられる．仮想空間内での見え方としてはこれで正しいのであるが，これをそのままディスプレイに描画してしまうと，そのディスプレイを更に角度 $\theta$ から見ることになるため，歪みが大きくなる．これを従来手法 B の問題点 (1) と呼ぶことにする．

また，従来手法 B では，$\theta$ が大きくなると白目とディスプレイの境界が眼球の膨らみに隠れてしまい，仮想カメラには眼球の膨らみが白目の淵として投影される．この場合の白目中央から白目の淵までの距離は以下のように計算される．

$$R - r|sin\theta| \tag{2}$$

このように，従来手法 B では白目中央から白目淵までの距離が左右で異なってしまい[6]，平面に目を描画したようには見えなくなる．これを従来手法 B の問題点 (2) と呼ぶ．

## 5 提案手法

提案手法では，提案手法 B の問題点 (1)，(2) を次のようにして解決する．

[5] 2 次元的に描画したときの白目の幅は CG 設計者から与えられるため，$r$ が決まれば $R$ は決まる．よって，決定すべきパラメータは $r$ のみとなる．

[6] 白目中央から白目の淵までの距離だけでなく，白目の淵のラインを投影すると，$\theta$ が大きくなるに従って，満月から上限の月に変化するように欠けていく．

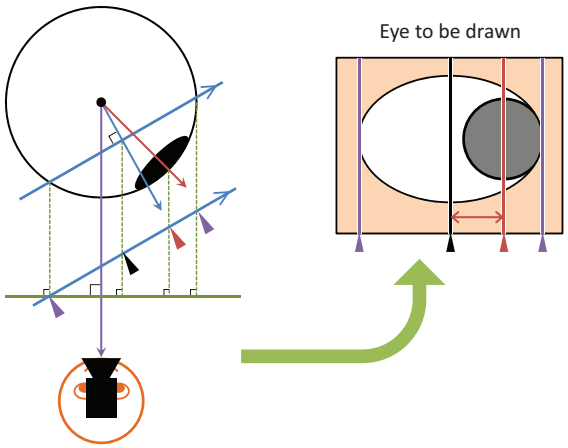



図 6: 提案手法

1. 仮想カメラからの見え方をそのままディスプレイに描画するのではなく，更にディスプレイ面に投影したものを描画する．

2. 白目の淵の位置は，常にディスプレイ面と眼球の境界を投影したものとして算出する（境界が眼球の膨らみに隠れても境界が透けて見えると考える）．

これらの対策により，白目の淵は従来手法 A と同様，ユーザがディスプレイを見る角度 $\theta$ によって変化しなくなる．また，$\theta$ の変化に対する黒目の移動量は，従来手法 A よりも従来手法 B に近い曲線を描く．

図 6 に模式図を示す．ディスプレイ面に眼球が埋め込まれた状態の仮想 3 次元空間を想定し，仮想カメラを実際のユーザの視点に置く．視線方向 $\phi$ へ眼球を動かした上で，仮想カメラからの 2 次元的な目の見え方をディスプレイ面に投影したものをディスプレイに描画する．

白目中央からの黒目の距離量の計算式は以下のようになる．

$$\frac{(R - \frac{r}{cos\phi})sin(\phi + \theta) + rtan\phi cos\theta}{cos\theta}$$

また，白目中央から白目の淵までの距離の計算式は以下のようになる（提案手法 A と同じ）．

$$\sqrt{R^2 - r^2}$$

以下，各手法を使って計算した黒目の移動量をプロットした例を示す．ここでは，エージェントの視線方向として重要な次の二つの場合を取り上げる．

- エージェントが正面（ディスプレイの法線方向）を見ている場合
- エージェントがユーザの方向を見ている場合

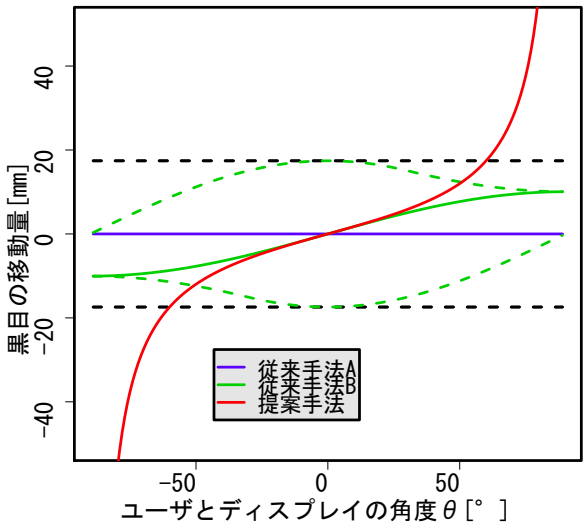



図 7: エージェントが正面を見ているときの $\theta$ と黒目移動量の関係（$r = 0.5R$ の場合）; 点線は白目の淵を示す

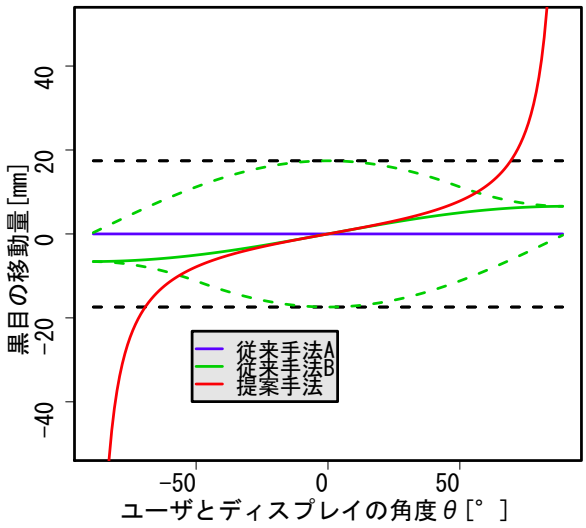



図 8: エージェントが正面を見ているときの $\theta$ と黒目移動量の関係（$r = 0.75R$ の場合）

エージェントが正面を見ていることを正しく表現できなければ，ディスプレイをエージェントの頭部と見做してもらっても，その頭部が指し示す方向が不正確になる．また，エージェントの頭部がユーザのほうを向いていない状態でユーザと目を合わせる行為は，注意がまだユーザに向いていることを示したり，ユーザに返答などを求めていることを伝えたりといった役割を果たす．

### エージェントが正面を見ている場合

まず，エージェントが正面を見ている場合のグラフを図 7 と図 8 示す．図 7 は眼球を半径の 1/2 だけ飛び出させた場合（$r = 0.5R$），図 8 は $r = 0.75R$ となるよう眼球を半球になるまで飛び出させた場合である．点線は白目の淵を表す（黒は従来手法 A と提案手法，緑は従来手法 B）．

これらは各計算式に $\phi = 0$ を代入することで計算でき，式を変形するとそれぞれ次のようになる．

$$\begin{aligned} \text{従来手法 A} &: 0 \\ \text{従来手法 B} &: (R-r)sin\theta \\ \text{提案手法} &: (R-r)tan\theta \end{aligned}$$

図 7 から分かるように，従来手法 A の場合，ユーザがディスプレイを見る角度 $\theta$ に因らず，常に黒目は白目中央に位置する．一方，従来手法 B では黒目の位置は $\theta$ に比例して大きくなるが，白目（緑の点線）も縮んでしまう．提案手法は，$\pm 40°$ の範囲では従来手法 B に近いラインを描き，且つ，白目は変形しない．しかし，$\theta$ が $\pm 40°$ を越えたあたりで従来手法 B から大きく逸れはじめ，$\pm 60°$ 付近で白目を飛び出てしまう．図 7（$r = 0.5R$）と図 8（$r = 0.75R$）の結果を比べると，ディスプレイ面に眼球を埋め込ませるほど，黒目が白目から飛び出す角度が大きくなっていることがわかる．

### エージェントがユーザの方向を見ている場合

次に，エージェントがユーザの方向を見ている場合のグラフを図 9 と図 10 示す．図 9 は眼球を半径の 1/2 だけ飛び出させた場合（$r = 0.5R$），図 10 は $r = 0.25R$ になるよう眼球を飛び出させた場合である．

これらは各計算式に $\phi = -\theta$ を代入することで計算でき，式を変形するとそれぞれ次のようになる．

$$\begin{aligned} \text{従来手法 A} &: -Rsin\theta \\ \text{従来手法 B} &: -rsin\theta \\ \text{提案手法} &: -rtan\theta \end{aligned}$$

図 9 から分かるように，従来手法 A の曲線は従来手法 B の曲線より傾きが大きい（ユーザの位置をオーバーシュートしてしまう）．提案手法は，先ほどの例と同様，$\pm 40°$ の範囲では従来手法 B に近いラインを描き，且つ，白目は変形しない．こちらの場合でも $\theta$ が $\pm 40°$ を越えたあたりで黒目の位置が従来手法 B から大きく逸れはじめ，$\pm 60°$ 付近で白目を飛び出てしまうが，こ

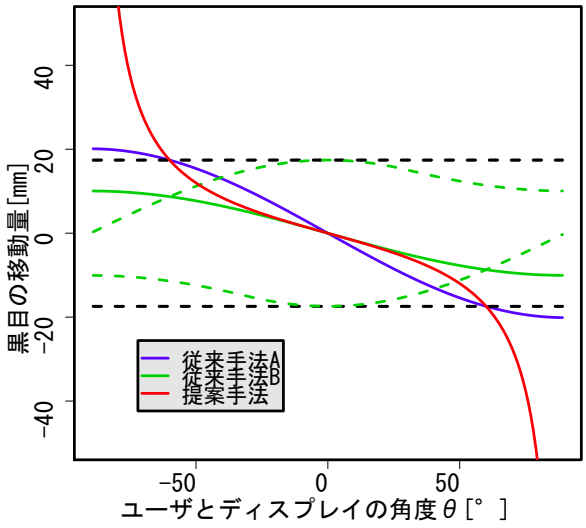



図 9: エージェントがユーザの方向を見ている場合の $\theta$ と黒目移動量の関係（$r = 0.5R$ の場合）

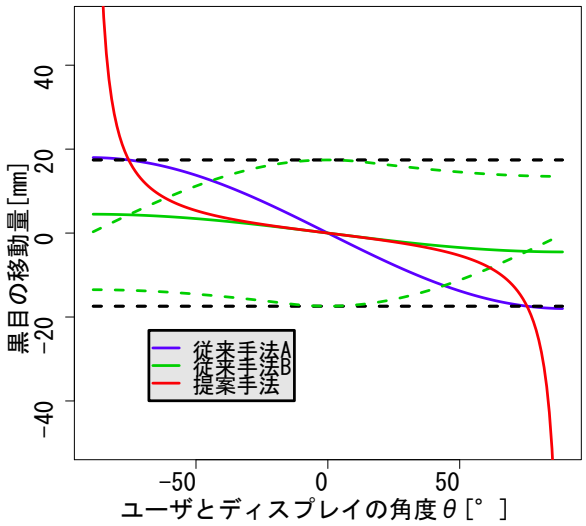



図 10: エージェントがユーザの方向を見ている場合の $\theta$ と黒目移動量の関係（$r = 0.25R$ の場合）

れはいずれの手法でも同様である．図 9（$r = 0.5R$）と図 10（$r = 0.25R$）の結果を比べると，いずれの手法でも，ディスプレイ面から眼球を飛び出させるほど，黒目が白目から飛び出す角度が大きくなっていることがわかる．これは，ディスプレイが正面を見ている場合と逆の方向に $r$ を変化することになる．

### 実装例

図 11 に提案手法を用いてエージェントの目の描画した例を示す（$r = 0.5R$，$\theta = 20°$）．参考までに，従来手法 A の結果も示す．これらはユーザ方向を見る場合，つまり $\phi = -20°$ とした場合であるが，ここまでに議論してきた通り，従来手法 A は黒目がユーザを通り越えた方向を見ているように感じられる．

また，図 12 に上と同様の条件で，エージェントがディスプレイ面の法線方向を見るようにしたものを示す．（この図ではディスプレイ面の法線方向に銀杏の葉を配置した．）こちらに関しても，提案手法・従来手法 A の 2 つのモデルを用いた場合を示したが，従来手法 A では銀杏の葉よりもユーザ寄りの方向を向いているように感じられる．提案手法では銀杏の葉よりも少し行き過ぎた方向を見ているように感じられるが，それは r を調整すれば修正できる問題である．

### 輻輳を加味した場合

最後に，輻輳を加味した場合の計算式を以下に記載しておく．$\alpha$ は輻輳角であり，左右の “目” の輻輳角を求めて個別に計算する．

$$\frac{(R - \frac{r}{cos\phi})sin(\phi - \alpha + \theta) + rtan\phi cos(\alpha - \theta)}{cos(\alpha - \theta)} \quad (3)$$

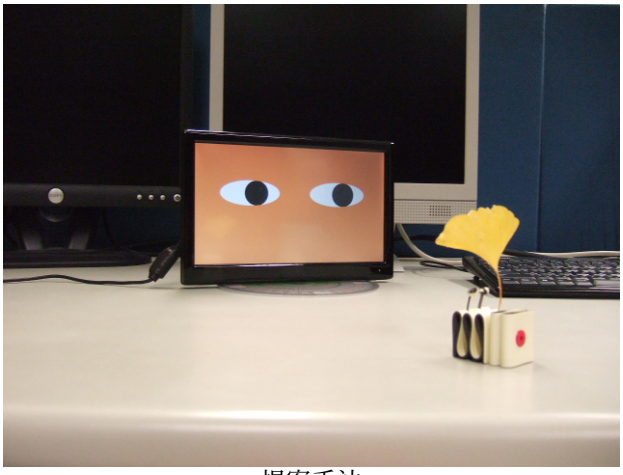
## 6 まとめ

本稿では，首振りディスプレイエージェントのための “目” の計算モデルを提案した．ディスプレイがエージェントの頭部としての役割を果たすためには，エージェントの目がディスプレイ表面に 2 次元的に描かれているように見せるのがよい．しかし，2 次元的な “目” による 3 次元的な実世界への参照・指示方向を求める計算モデルはこれまでに提案されていない．本研究では，ユーザの視点を検出して運動視差を再現する従来の手法（従来手法 B）を改変し，2 次元的な目を描画しても整合性が保たれる計算モデルを提案した．ユーザがディスプレイを斜めから見る角度 $\theta$ を横軸として黒目の移動量をグラフにプロットした結果，従来手法 B に近い黒目の移動量のカーブを描きながら，従来手法 B のように白目が歪まないことを示した．

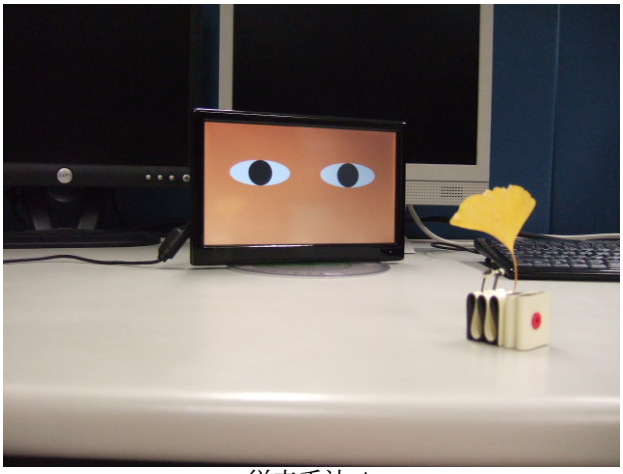
提案手法

従来手法 A

図 11: ユーザ方向を見る場合
$r = 0.5R$，$\theta = 20°$，$\phi = -20°$



提案手法



従来手法 A

図 12: ディスプレイ面の法線方向を見る場合
$r = 0.5R$，$\theta = 20°$，$\phi = 0°$

ただし，提案手法では，比較的狭い範囲（$\pm 60°$）で黒目が白目の範囲から出てしまうことが明らかになった．この問題は，仮想空間上での眼球中心とディスプレイ平面の距離 $r$ を調整することで改善することはできる．しかし，エージェントが正面を見ているときとユーザを見ているときで，問題が軽減される $r$ の変化方向が逆であった．この問題への対策の一つとして，$r$ を $\theta - \phi$ の関数として定義し，ユーザとディスプレイとエージェントの視線方向の関係に応じて $r$ を変更することが考えられる．

もう一つの対策として，従来手法 B に基づいた別の計算モデルを構築することが考えられる．我々が既に考案しているのは，従来手法 B の白目の淵の位置が（従来手法 A と同じ）直線を描くように引き伸ばしてやる方法である．この方法であれば，少なくともエージェントが正面を向いている場合には，黒目が白目の範囲を飛び出してしまうことは起こらない．

今後は，この二つの対策を実装し，実際に幾人かの実験協力者のキャリブレーションデータを集めて，2 次元的に描画された目であっても実世界への参照・指示がうまく成立することを示していく予定である．

## 参考文献


[1] http://dailynewsagency.com/2011/05/29/idesktop-vr/.

[2] S.M.Anstis, J.W.Mayhew, and Tania Morley. The perception of where a face or television ‘ portrait ’ is looking. *The American Journal of Psychology*, 1969.

[3] 室彰浩, 佐藤隆夫. CG 顔を用いた視線方向知覚の検討. 電子情報通信学会技術研究報告. MVE, マルチメディア・仮想環境基礎, 2005.