



VRビギナーズ別冊

# 「0グラムの読書をしたい」

・デモの技術系解説と参考企業など

FreePaper (無料)

# 0グラムの読書をしたい

## —— はじめに

以前からコミケでVRやMR、ホログラムを使ったデモをやってきました。その動機は、

「本を手に持ちたくない」

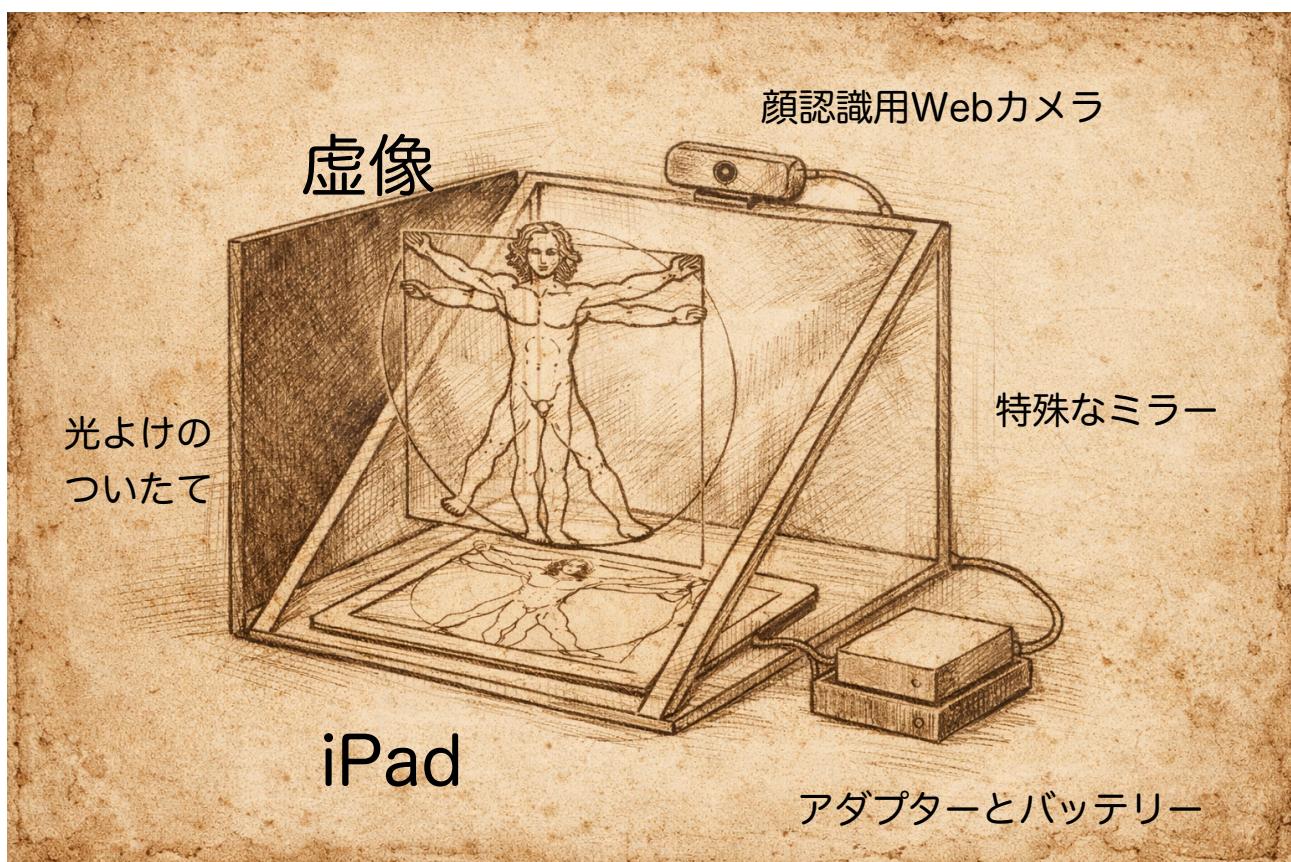
です。本は重いし硬くて、頭の上にうっかり落とすと大惨事です。弊サークルの同人誌ならまだしも、ハードカバーの小説となると流血ものです。

というわけで、前回は空中にホログラムで浮かべた本を、指で操作することで立体を感じてもらっていました（今も大学では展示しています）。しかし、昨今、閲覧者の動きに合わせて絵を動かすことで立体を感じさせるデモがとても多いですよね。というわけで、今回はそれをホログラムと合わせたものを用意しました。（ここまで来るとすでに本とあまり関係がないですが！）

さて、どう見えるでしょうか？

## —— 顔認識で体験するのホログラムデモについて

この展示は、iPadの画面に映る映像と、ミラーを使った光学的な仕組み、そして顔認識の技術を組み合わせた実験的なデモです。見る人の「首の向き」や「立ち位置」によって、同じ映像でも奥行きの感じ方が変わる、という点を体験的に確かめることを目的にしています。



iPadに表示された映像を特殊なミラーで反射させ、その反射像（虚像）をミラー前方にひん曲げて、「外」=あたかも空中に浮かんでいるように見せています。特殊なミラーはとても高価なので、試したい人はアクリル板などでやりましょう。三角柱の「内」に表示されてしまいますが、雰囲気はわかるかと思います。

映像は固定されていません。iPadのカメラで閲覧者の顔を検出し、顔の位置に応じて表示内容をリアルタイムに変えています。

その結果、頭を少し左に動かすと、虚像もそれに合わせて見え方が変わります。

前に近づけば、対象は大きく見え、後ろに下がれば、少し奥に引っ込んだように感じられます。実際には、空間の中で何かが動いているわけではありません。「自分の位置」と「画面の描き方」の関係が、そう錯覚させているだけです。

この感覚は、VRゴーグルのような没入型の体験とは少し違います。

目を覆う装置はなく、あくまで「現実の空間の中」にある展示物を見ているのに、視点だけが計算によって補正される。その中間的な感覚が、このデモの面白さです。

この方式の良いところは、「他人から何をやっているか確認できるところ」です。いわゆるVR機器は業務用から民生用まで新旧腐るほど試してきましたが、最大の不満点は怪しい変態に見えるところです。AppleのVision Proなんて、MacBookと合わせてスタバで使ってみましたが、付いてきた院生も引いていました。やはり、隣人から「正常」に見えることが重要なのです。

---

## 中で動いているもの

### —— GitHubに置かれたHTMLの中身

このデモは、専用アプリではなく、Webページとして作られています。

そのため、GitHubにアップロードされたHTMLファイルを、ブラウザで開くだけで動作します。昔のように私が管理している学内サーバーを使っているわけではないので、サーバー落ちなどの責任問題からも回避できます。現代技術素晴らしい。

中では、いくつかの技術が組み合わされています。

- ・Webカメラから映像を取得する仕組み
- ・顔の位置を検出する顔認識ライブラリ
- ・3D空間を描画するJavaScriptライブラリ
- ・顔の位置と、仮想的なカメラ位置を結びつける簡単な計算

といっても、数式や難解な処理が詰め込まれているわけではありません。

「顔が画面のどこにあるか」「前より近づいたか、離れたか」という情報を、そのまま視点の移動量に変換しているだけです。

このHTMLは、GitHub Pagesで公開されています。

#### DEMOページ

<https://wasuke.github.io/iPadBox/>

#### リポジトリページ

<https://github.com/wasuke/iPadBox/>

URLを開けば、誰でも同じデモを体験できます。インストールは不要で、コードもすべて見ることができます。学生向けに授業で使うときは、

- ・数値を少し変えると、見え方がどう変わるか
- ・顔認識を切ったら、錯覚はどこまで残るか
- ・iPadではなくPCや別のカメラを使ったらどうなるか
- ・より適したコンテンツは何が考えられるか
- ・博物館での応用プラン

といった改造を課題にしています。

## —— 今っぽい話題とのつながり

最近のARやVR、そして空間コンピューティングと呼ばれる分野では、「何を表示するか」よりも、「どこから見ているか」を重視する流れが強くなっています。

Apple Vision Pro や各種ARデバイスも、実は同じ発想を使っています。世界を直接変えるのではなく、「見る人の位置」を正確に測り、その位置に合わせて映像を調整することで、現実と仮想を自然につなげています。

この展示は、その考え方を、できるだけ小さく、分解した形で見せているものです。高価なデバイスがなくても、「視点」と「計算」だけで、見え方はここまで変わる、という例であります。

## —— おわりに

このデモは、完成品というより「途中のスケッチ」に近いものです。ダヴィンチの手稿が、発明そのものではなく、考え方の跡だったように、ここにあるのも「考え方の見える形」です。

コードを読んで、少し変えて、壊して、また戻す。その過程そのものが、この展示の延長線上にあります。

## —— index.htmlの役割

index.html は「起動ランチャー」です。やることは3つです。

### ① カメラ入力の準備（選択）

- ・利用可能なカメラ（例：前面・背面・USB）を列挙します
- ・ユーザーが選んだカメラを、次ページで使えるように deviceld やモードとして保持します

※ iPad+USBカメラの場合、ブラウザが返す deviceld を使って指定します。

### ② コンテンツの準備（選択）

- ・「空っぽの部屋（市松・格子）」か「3Dモデル（GLB / STLなど）」を選びます
- ・選択されたモデルのパス（例：models/sample.glb）を保持します

### ③ viewer.html へ遷移（URLパラメータを付与）

- ・選択結果を URL クエリに詰めて viewer.html を開きます

例：

- ・空っぽ部屋：viewer.html?camMode=front
- ・モデル：viewer.html?camMode=usb&deviceld=...&model=models/sample.glb

index.html は、描画や顔認識はしません。「設定を作って、viewer.html に渡す」だけです。

## —— viewer.htmlの役割

viewer.html は「本体（レンダラー+トラッキング+錯覚表現）」です。大きく5つの仕事があります。

### ① URLパラメータを受け取り、設定を確定

- ・camMode (front/back/usb)
- ・deviceld (USBカメラ指定用)
- ・model (GLBなどのパス)
- ・model=\_\_EMPTY\_ROOM\_\_ のような指定なら空部屋モード

### ② カメラ起動 → 映像ストリーム取得

- ・getUserMedia() でカメラ映像を取得し、<video> に流します
- ・以後、フレームごとに動画から画像を取り出して顔認識に渡します

### ③ 顔認識（FaceLandmarker）

- ・MediaPipe の FaceLandmarker を初期化します

- ・各フレームで顔のランドマークを取得します
- ・複数人が映った場合は、最初にロックした顔に近い候補を追い続けます（ロック追跡）

#### ④ 顔の2D位置 → 仮想カメラ制御へ変換

- ・ランドマークから
  - ・顔の中心（画面内のx,y）
  - ・顔の大きさ（近い/遠いの指標）
- を計算します
- ・それを元に、三次元空間のカメラ位置とFOV（ズーム量）を更新します
  - ・左右・上下：覗き込み方向
  - ・近づく：ズーム（起動時の顔サイズを基準）

さらに急にガタつかないように、スムージング（指数移動平均）を入れています。

#### ⑤ three.js 描画（箱+照明+モデル+虚像っぽい見え）

- ・箱（内側の壁）は市松模様+明るいパッチのテクスチャ
- ・照明は強めのスポットライト（白飛びOK）で立体感を出す
- ・モデルがあれば中央に配置し、ゆっくり回転させる
- ・最終的に、毎フレーム renderer.render(scene, camera) で描画します

## —— 大まかなデータフロー（全体）

「設定」→「映像」→「顔情報」→「カメラ」→「描画」の一本道です。

### A. ページ間 (index → viewer)

- 1.ユーザーがカメラとコンテンツを選ぶ
- 2.index.html が viewer.html?... というURLを組み立てる
- 3.viewer.html が URLパラメータを読み取って起動条件を確定

### 渡されるデータ

- ・camMode (front/back/usb)
- ・deviceId (必要なら)
- ・model (GLBの相対パス、または空部屋指定)

### B. viewer 内（リアルタイム処理）

- 1.カメラ映像（videoフレーム）を取得
- 2.FaceLandmarker に渡す
- 3.顔ランドマーク（点群）を得る
- 4.点群から「顔中心」と「顔サイズ」を計算
- 5.それをカメラ位置・FOVへ変換（スムージング込み）
- 6.three.js が箱・照明・モデルをそのカメラ視点で描画

リアルタイムに流れるデータ

- ・VideoFrame (カメラ映像)
- ・FaceLandmarks (顔の点群)
- ・HeadCenter (x,y)
- ・FaceSize (近接度)
- ・CameraPose (カメラ位置)
- ・FOV (ズーム)

## —— まとめ

- ・index.html : 設定を選ばせて viewer.html に渡すページ
- ・viewer.html : 顔認識→視点計算→three.js描画を行う本体
- ・データフローは「URL設定 → カメラ映像 → 顔ランドマーク → カメラ制御 → 描画」の一本線です

## —— もっと付録

説明を飛ばした「特殊なミラー」ですが、どうしても欲しい人は以下で検索してください。

---

### 1) ミラー／反射アレイ系 (Parity Mirror系)

- ・Parity Innovations (パリティ・イノベーションズ) : Parity Mirror (空中にフルカラー像) / 空中タッチなどインタラクションも展開

---

### 2) 空中結像プレート系 (ASKA3D系)

- ・アスカネット (Asukanet) : ASKA3D (特殊パネルで等距離の空中に結像させる“空中結像プレート”)

※同社はエアリアルイメージング事業として整理して発信しています

- ・IMUZAK :マイクロレンズアレイ方式で「空中に飛び出す像」を作る技術を公開 (3D Air Floating Images)
- ・TOWA (東和) : 微細レンズを統合した「floating image technology (浮遊像)」を事例として紹介
- ・OMRON Components (オムロン系部品) : マイクロプリズム／マイクロレンズで光線を制御し、aerial images (空中像) などに言及 (部品・技術として)

---

3) 霧・ミスト“スクリーン”系（空中に投影面を作る）

- ・FogScreen（フィンランド発）：薄い「乾いた霧」のスクリーンに投影し、通り抜け可能な“空中映像面”を提供
  - ・Panasonic（Silky Fine Mist の応用）：超微細ミストを使い、投影表示のメディアとしても実験・展示（サイネージ用途の紹介記事あり）
- 

4) “浮いて見える”系（方式は違うが近い体験を作る）

- ・HYPERVSN（Kino-mo系）：装着不要で“空中に浮いて見える”3Dビジュアルをうたうホログラフィック・ディスプレイ（回転ファン方式の系統）
- 

5) 研究色が強いが押さえておくと便利（企業研究）

- ・NTT：Mirror-Transcending Aerial Imaging System（MiTAI）を発表（鏡と空間をまたぐ見え方の研究）
- 

探索を広げるための検索キー（会社名＋方式）

- ・aerial imaging + plate + company
  - ・floating image + microlens array + company
  - ・dihedral corner reflector array + aerial display（Parity Mirror周辺の論文・特許系の入口になりやすい）
  - ・fog screen + projection + company
  - ・mist screen + projection + Panasonic
- 

発行日：2025/12/31

文責：中部大学人文学部メディア情報社会学科 桢和佑