加速度センサの更なる活用 ～入門から応用まで～ (Part2)

(株)バンダイナムコゲームス
コンテンツ制作本部
制作ディビジョン
技術部　技術サポート課
山口兼太郎
Kentaro_Yamaguchi@bandainamcogames.co.jp

BANDAI NAMCO Games

2/Sep '09

- 加速度センサを使った図形入力
- 重力方向に基づくコントローラの向き決定
- 加速度センサの誤差測定

加速度センサを使った図形入力

図形入力: コントローラの軌跡

コントローラを自由に動かす

加速度情報 → 軌跡を求める

ほぼ不可能

コントローラの動きに制約をつければ?

問題点と対策

向きが分からない

→ コントローラを平行移動

積分による誤差の集積

→ 始点と終点の位置関係を指定

例: 始点と終点が一致(閉じた図形)

未知数の個数 ＞ 条件の個数

→ 始点速度 または 終点速度を指定

(始点速度0, 終点速度0)

基本式

加速度センサ値(重力を含む)

$\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_{n-1}$

速度　　$\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{n-1}, \mathbf{v}_n$

位置　　$\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{n-1}, \mathbf{p}_n, \mathbf{p}_{n+1}$

$\mathbf{v}_{i+1} = \mathbf{v}_i + \Delta t\,(\mathbf{a}_i - \mathbf{g})$　　$\mathbf{g}$: 重力加速度ベクトル

$\mathbf{p}_{i+1} = \mathbf{p}_i + \Delta t\,\mathbf{v}_i$

すべてコントローラ座標系

原点は $\mathbf{p}_0$

条件式の追加

始点速度0 ($\mathbf{v}_0 = 0$)

始点の歪が小，終点の歪が大

終点速度0 ($\mathbf{v}_n = 0$)

始点の歪が大，終点の歪が小

両者の結果をブレンド

得られる結果

図形($\mathbf{p}_0$, $\mathbf{p}_1$,…, $\mathbf{p}_{n+1}$) = 3次元空間内の点列

重力加速度ベクトル $\mathbf{g}$

いずれも，コントローラ座標系

→ コントローラの向き(ピッチ，ロール)

→ 図形の向き(ピッチ，ロール)
(図形の上下方向が分かる)

まとめ

制約をつければ，加速度情報のみで，
図形入力は可能

- コントローラを平行移動
- 始点と終点の位置関係を指定
- 始点速度0，終点速度0

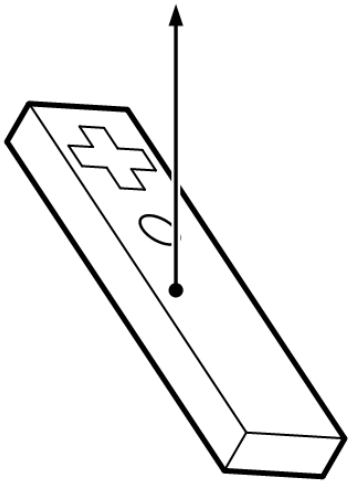
ただし・・・

- 短時間(2秒以内)限定
- 入力途中の形は分からない
- 入力にコツが要る

重力方向に基づく
コントローラの向き決定

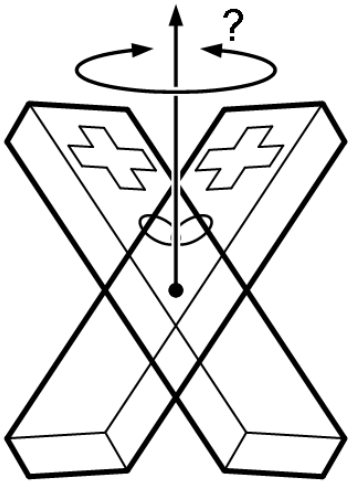
条件追加による向き決定

現実世界

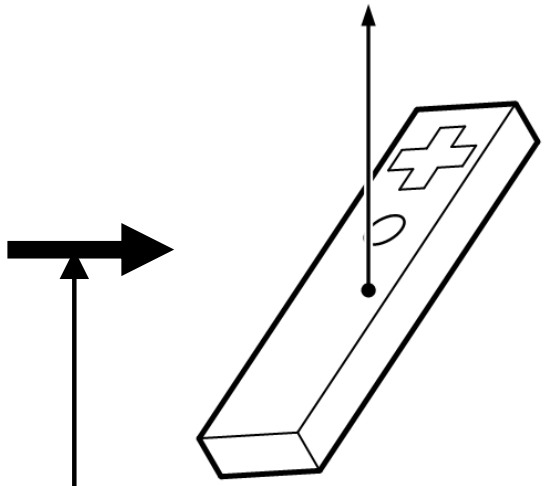


重力加速度を
検出(静止状態)

ゲーム内



分かるのは
上下のみ



向きが決まる

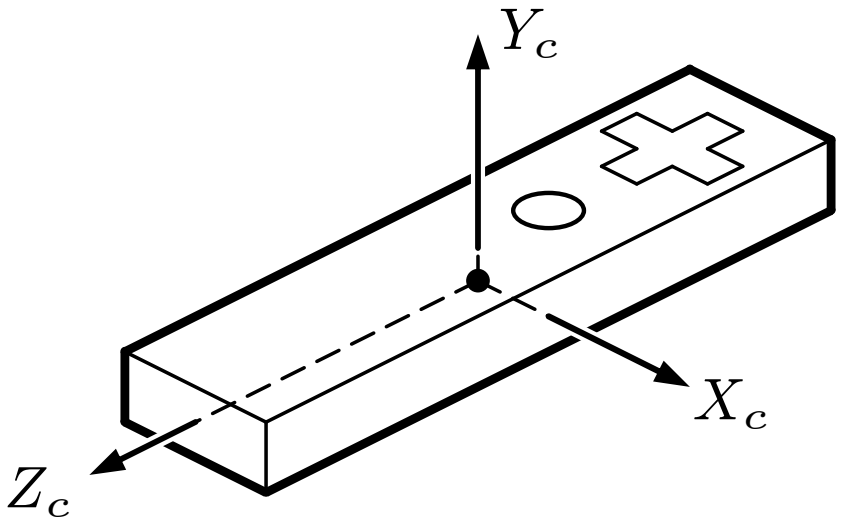
計算条件を追加

座標系

ワールド座標系
Xw, Yw, Zw

+Xw = 右
+Yw = 上
+Zw = 手前

コントローラ座標系
Xc, Yc, Zc



両者の向きが揃った状態 = 基準状態

アップベクトル $\mathbf{v}_u$

加速度センサが検出した「上向きベクトル」
=「アップベクトル」 $\mathbf{v}_u$
　(コントローラ座標系)
$|\mathbf{v}_u| = 1$

従来手法

左右方向を維持

+Xc を Xw Yw 平面の Xw ≧ 0 領域に入れる

前後方向を維持

+Zc を Yw Zw 平面の Zw ≧ 0 領域に入れる

いずれも 2方向でフリップ

フリップ現象は解消できるのか?

向き決め方法の分類

履歴非依存型

コントローラの向きは，現在の $\mathbf{v}_u$ だけで決まる。
(過去の $\mathbf{v}_u$ には依存しない)

履歴依存型

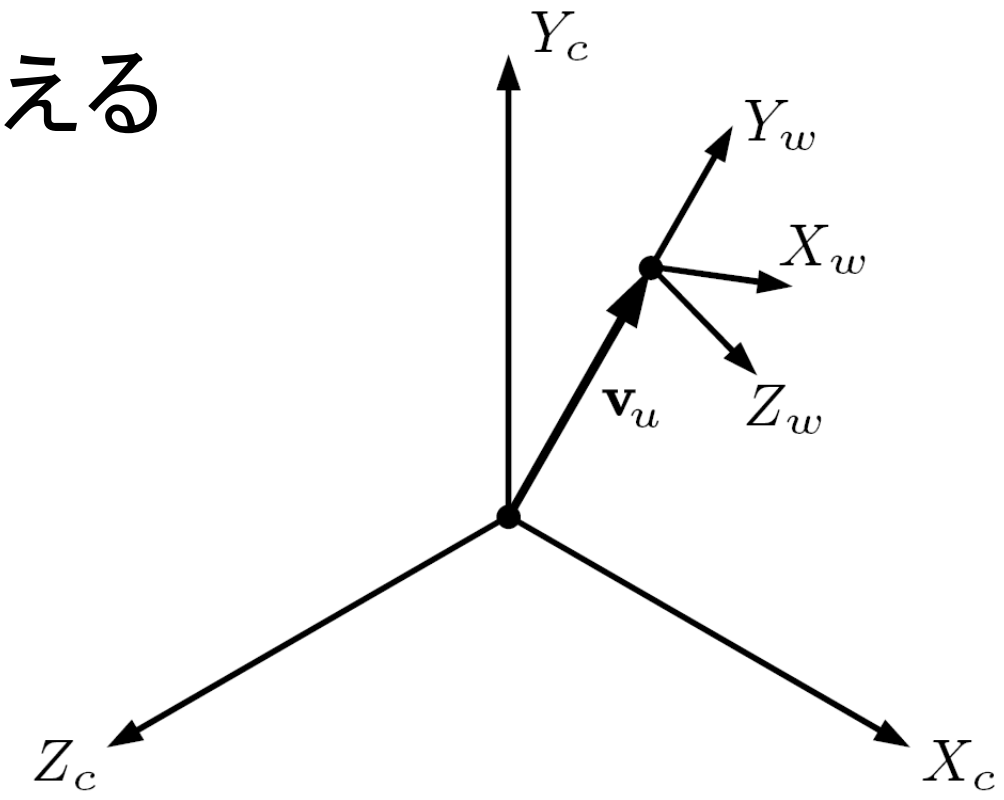
コントローラの向きは，現在の $\mathbf{v}_u$ だけでは
決まらない。
(過去の $\mathbf{v}_u$ に依存する)

履歴非依存型の特性

コントローラ座標系で
ワールドの向きをとらえる

$\mathbf{v}_u$ と Yw は一直線

Xw ⊥ $\mathbf{v}_u$
Zw ⊥ $\mathbf{v}_u$

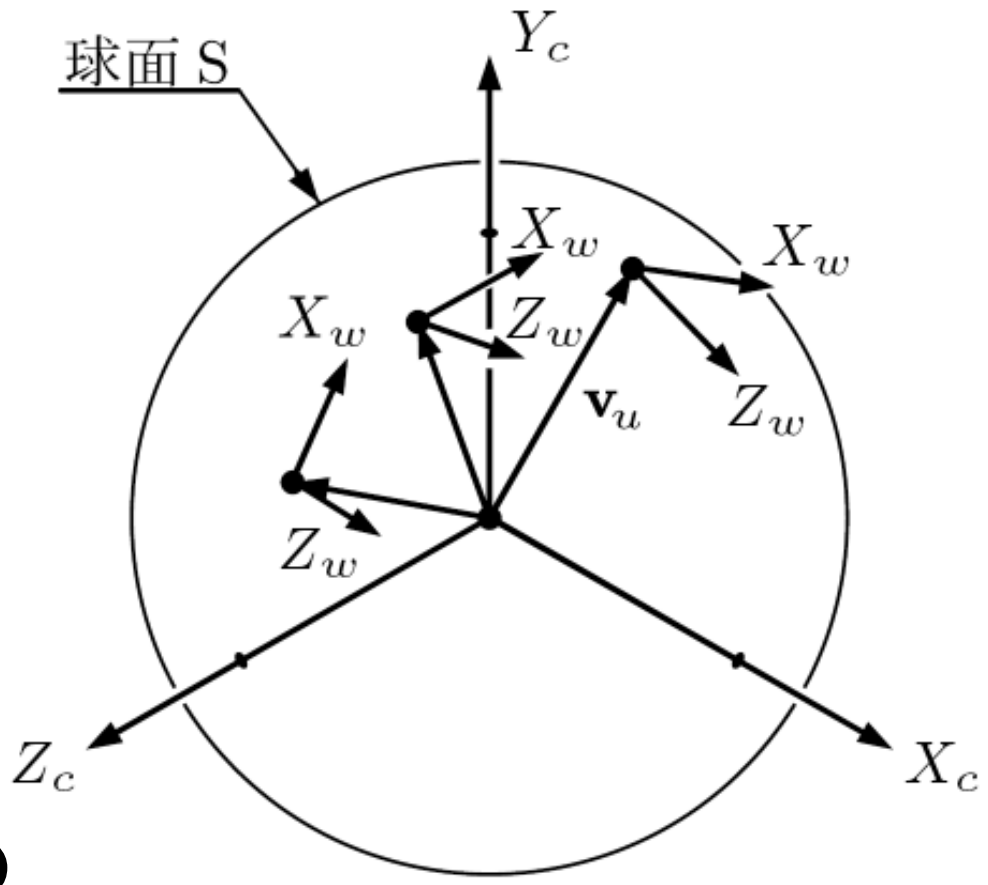


履歴非依存型の特性

$\mathbf{v}_u$ の終点は球面を作る

Xw, Zw は,
　球面S に接する

履歴非依存
↓
球面S上の各点が
それぞれ決まった方向の
Xw, Zw を持つ

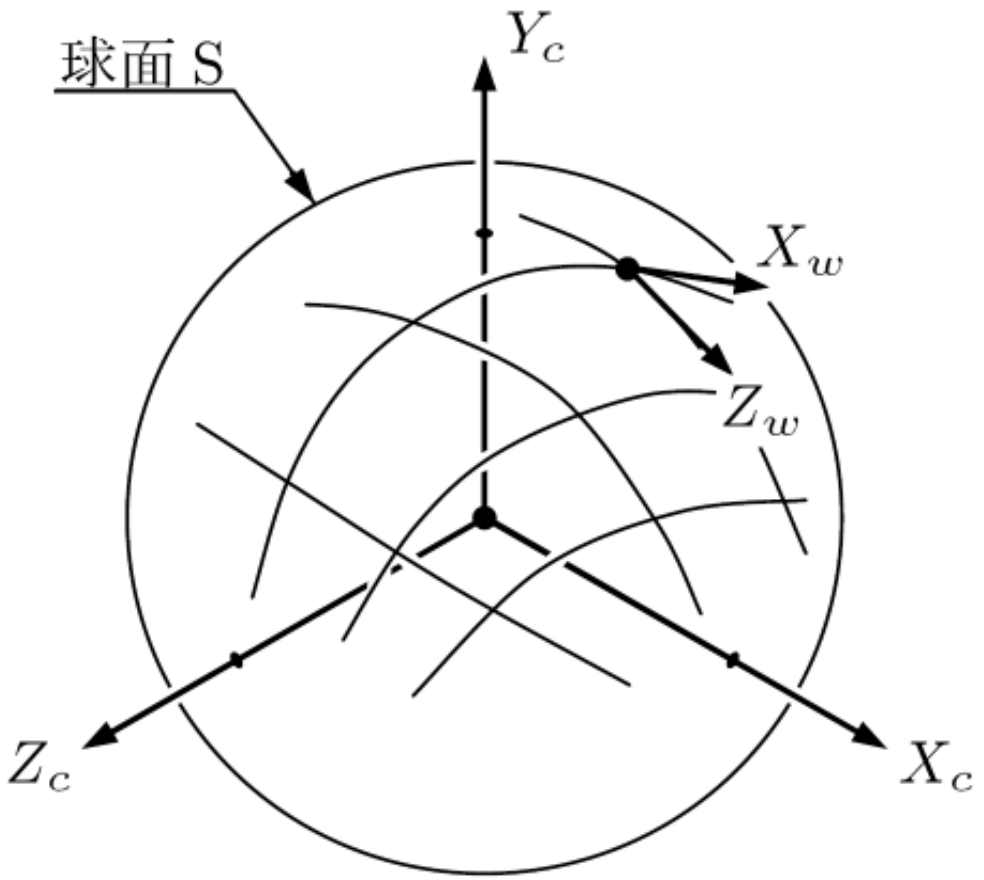


特性グリッド

各点における Xw, Zw の
向きを表す

コントローラ座標系に固定

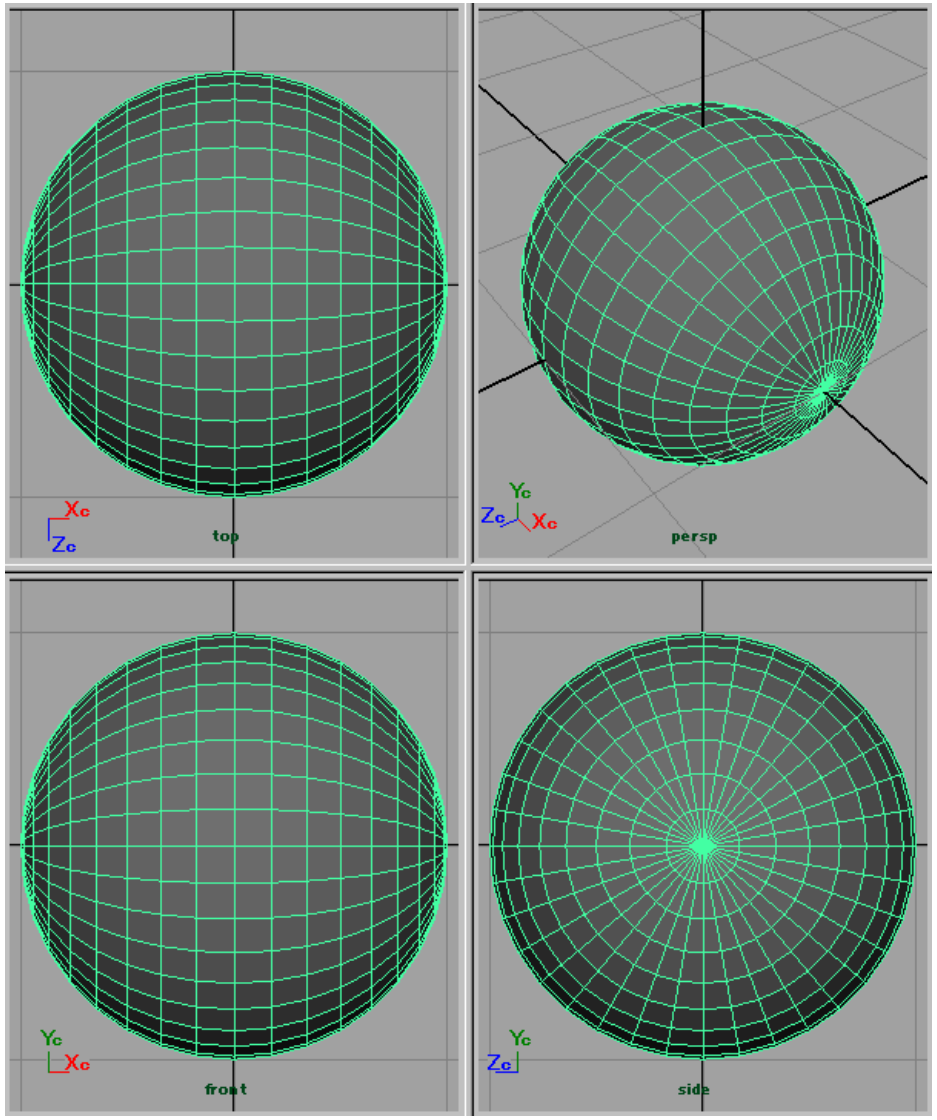
履歴非依存型の
向き決め挙動
↓
特性グリッド



従来手法の特性グリッド

従来手法
(左右方向を維持)

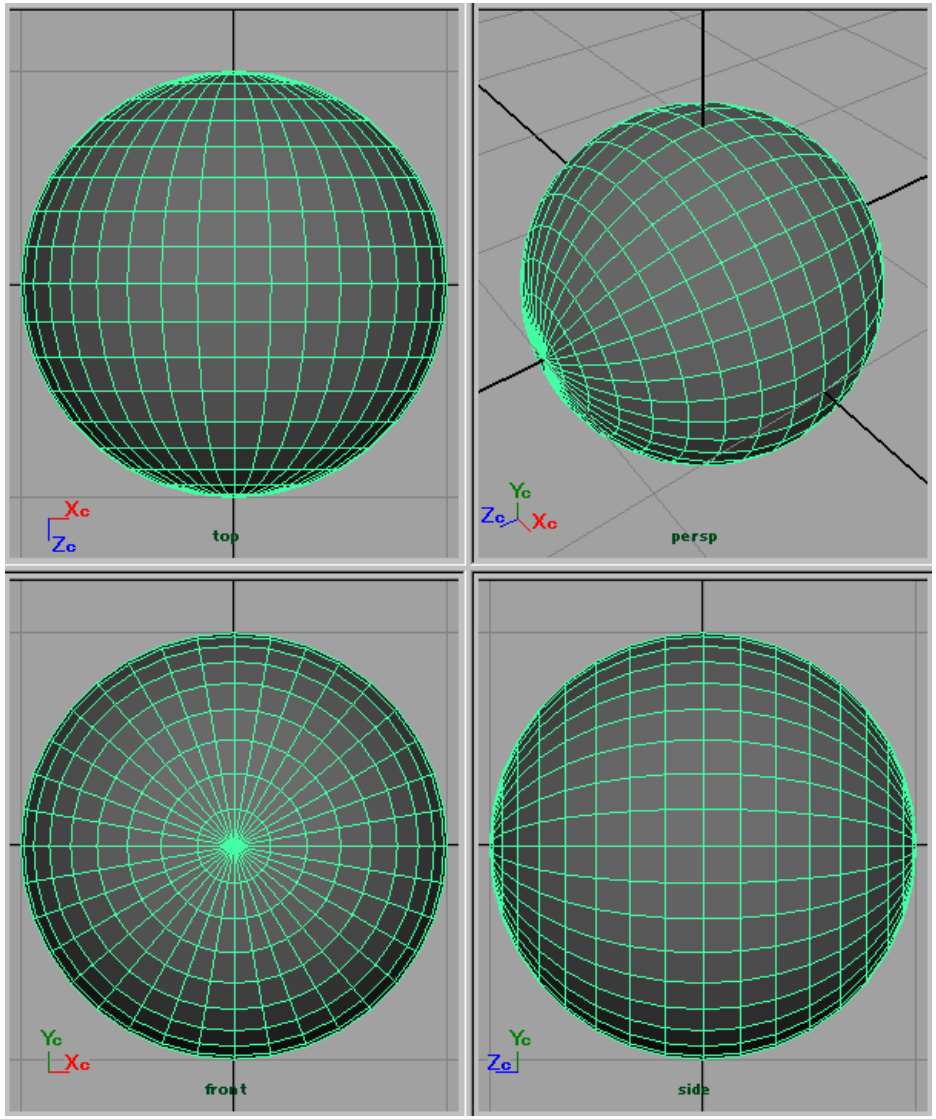
+Xc, -Xc 方向が
特異点(フリップ方向)



従来手法の特性グリッド

従来手法
(前後方向を維持)

+Zc, -Zc 方向が
特異点(フリップ方向)



フリップは回避可能か?

履歴非依存型では，フリップは回避できない

不動点定理から導かれる

「地球上には無風地点が存在する」

球の表面に沿った流れ
　⇒ 流速0 の点が必ずできる

Xw(または Zw)を流れと見なす
　⇒ 必ず特異点が存在する

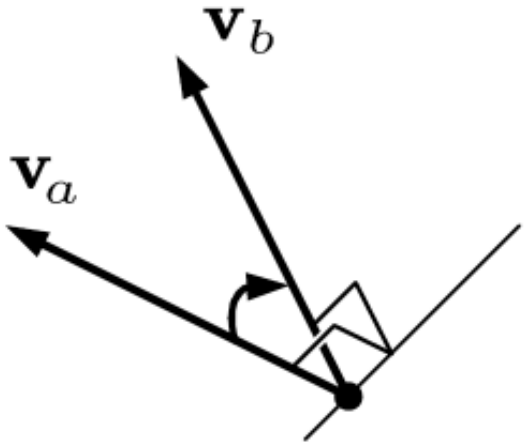
単純回転

$\mathbf{v}_a$ から $\mathbf{v}_b$ への単純回転 (定義)

$\mathbf{v}_a$, $\mathbf{v}_b$ のなす平面に垂直な軸の回りの回転
回転により $\mathbf{v}_a$ は $\mathbf{v}_b$ に重なる

$\mathbf{v}_a = \mathbf{v}_b \Rightarrow$ 無回転

$\mathbf{v}_a = -\mathbf{v}_b \Rightarrow$ 定義不能



履歴非依存型の改良

特異点を 1個にする
⇒ フリップ方向が 1方向になる

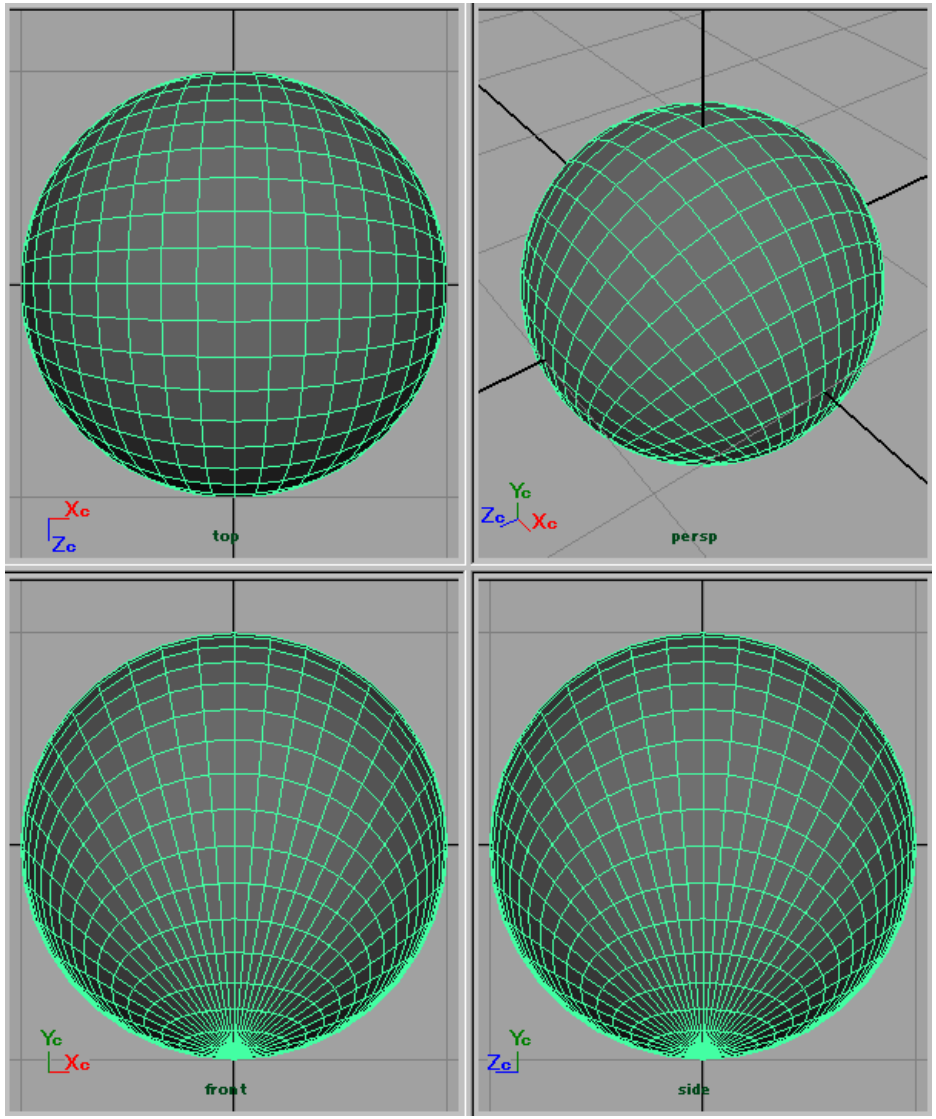
計算手順:

ワールド座標系にて，コントローラに以下の操作

1. 基準状態にセット
2. $\mathbf{v}_u$ から +Yw への単純回転を掛ける

履歴非依存型の改良

-Yc 方向が
特異点(フリップ方向)



履歴非依存型の改良

$\mathbf{v}_u$ が向く頻度が少ない方向 → $\mathbf{v}_r$
　$\mathbf{v}_r$ をフリップ方向にする

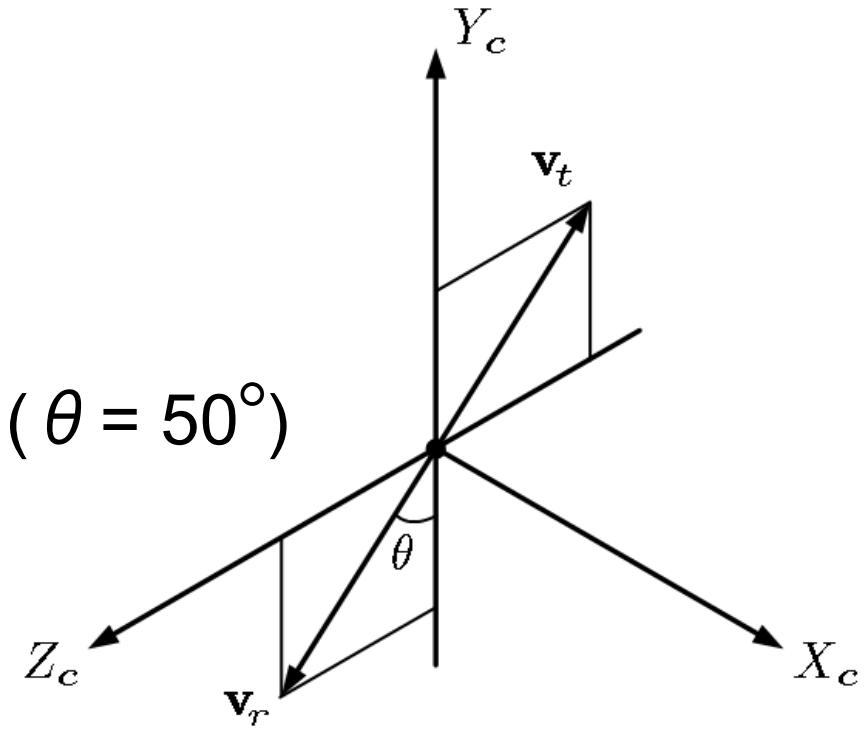
目標ベクトル $\mathbf{v}_t = -\mathbf{v}_r$

計算手順:

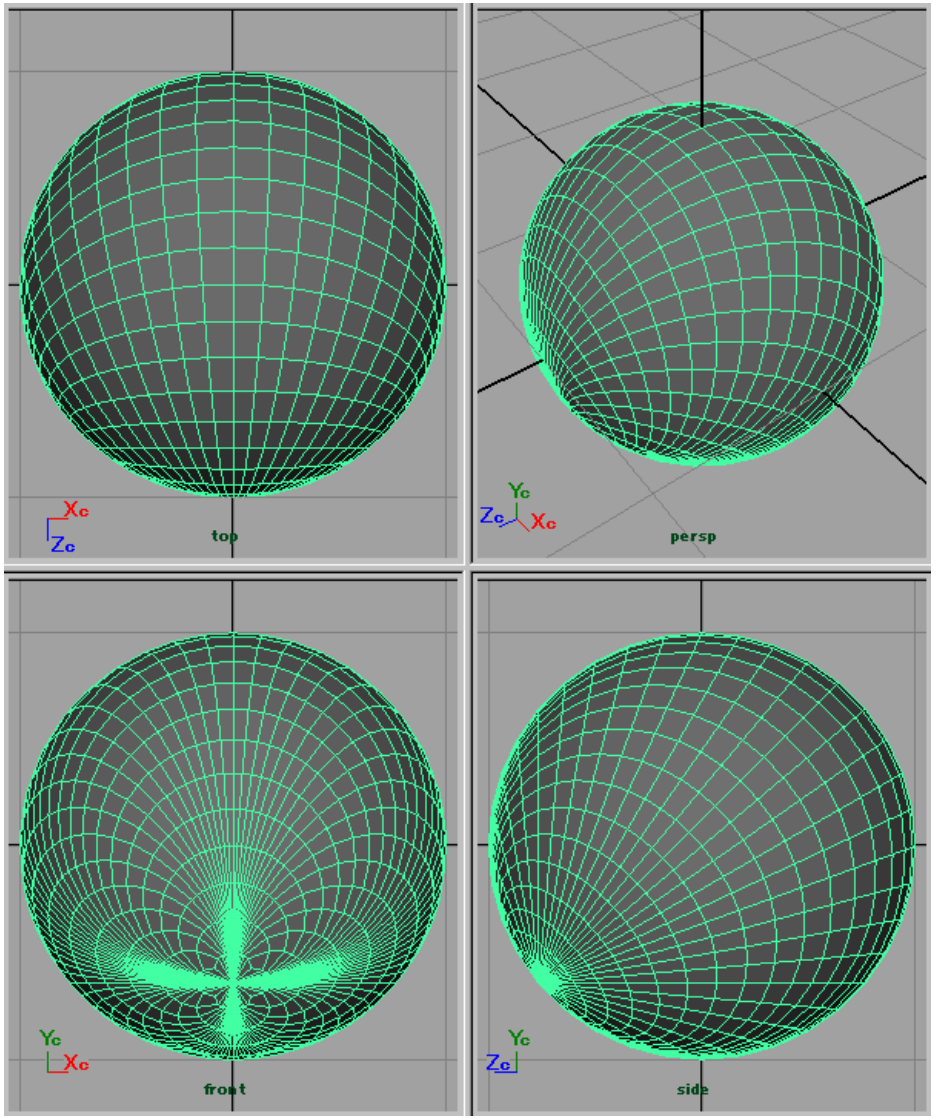
ワールド座標系にて，コントローラに以下の操作

1. 基準状態
2. $\mathbf{v}_u$ から $\mathbf{v}_t$ への単純回転を掛ける
3. $\mathbf{v}_t$ から +Yw への単純回転を掛ける

履歴非依存型の改良



($\theta = 50^\circ$)



$\mathbf{v}_r$ 方向が
特異点(フリップ方向)

履歴依存型

フリップを回避できる
(不動点定理の縛りを受けない)

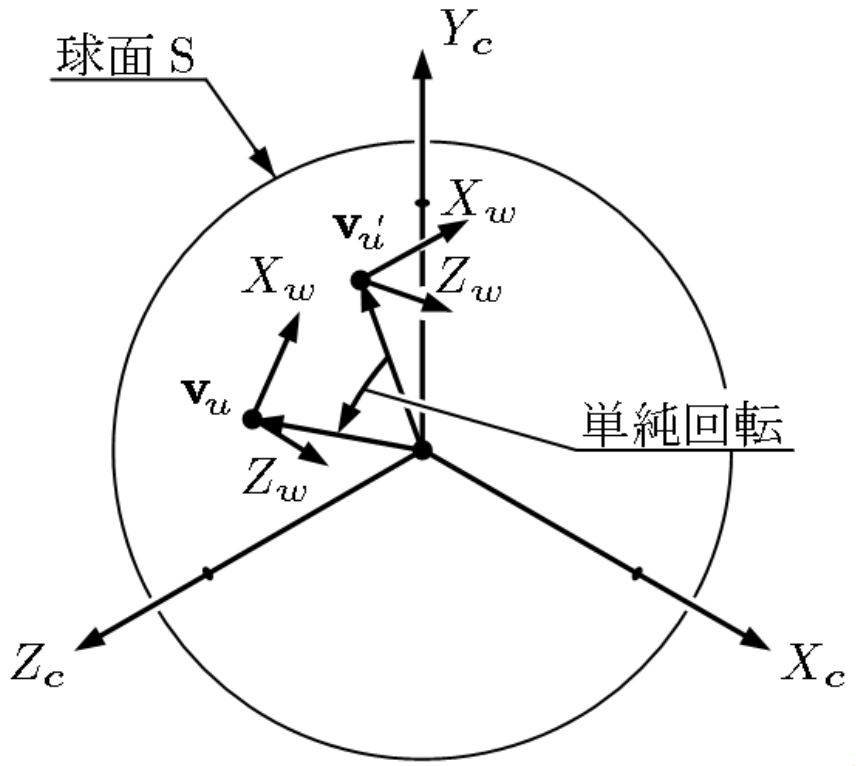
履歴依存型の例:

$\mathbf{v}_u$’：直前の時点の $\mathbf{v}_u$

ワールドに

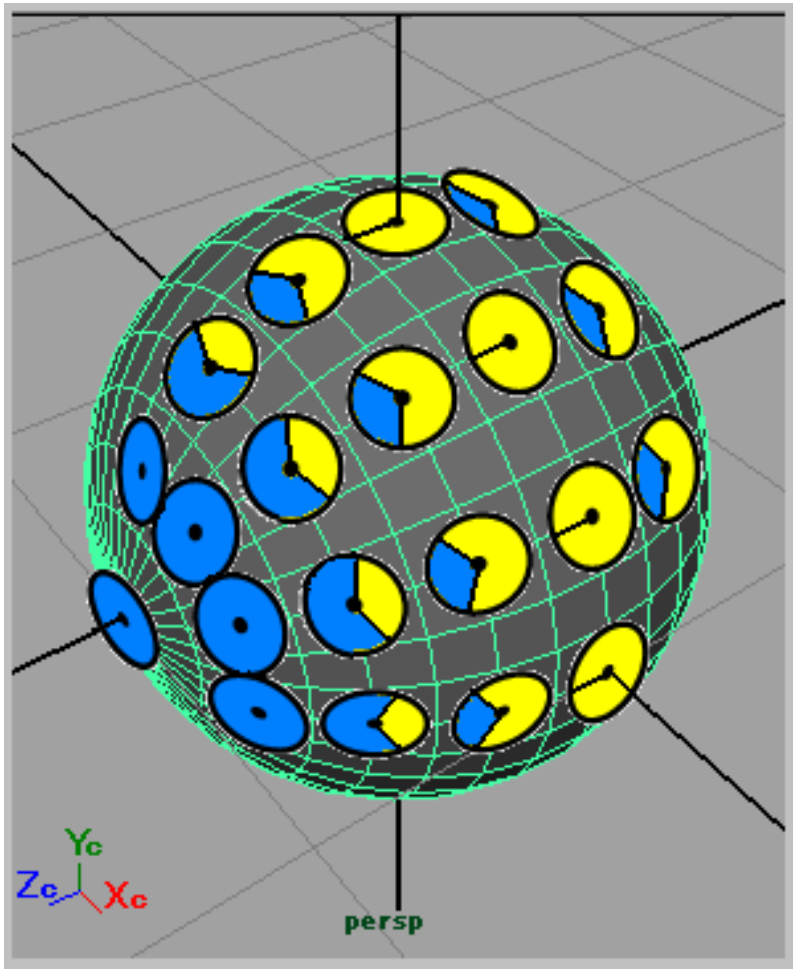
$\mathbf{v}_u$’ から $\mathbf{v}_u$ への単純回転
を掛ける

向きのずれが発生する



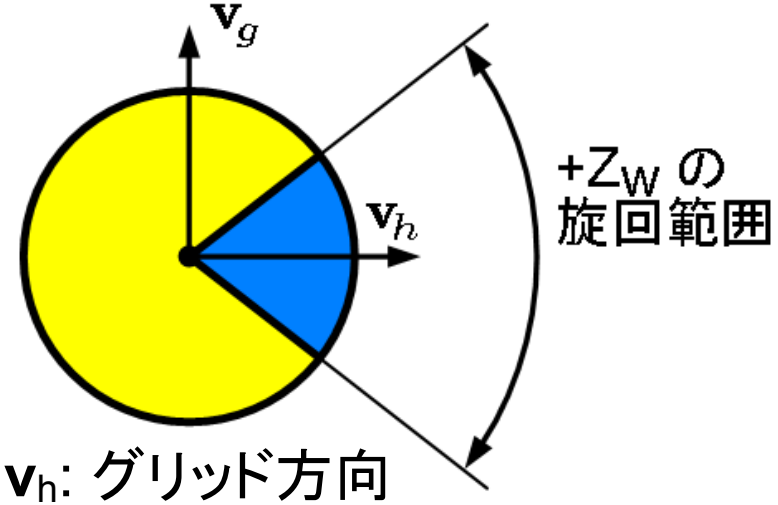
ハイブリッド型

「履歴非依存型」と「履歴依存型」の融合



球面S 上の各点にて，旋回範囲を設定(特性グリッドを基準)

旋回範囲: +Zw(または+Xw)が向くことのできる方向の範囲



$\mathbf{v}_g$, $\mathbf{v}_h$: グリッド方向

旋回範囲の設定

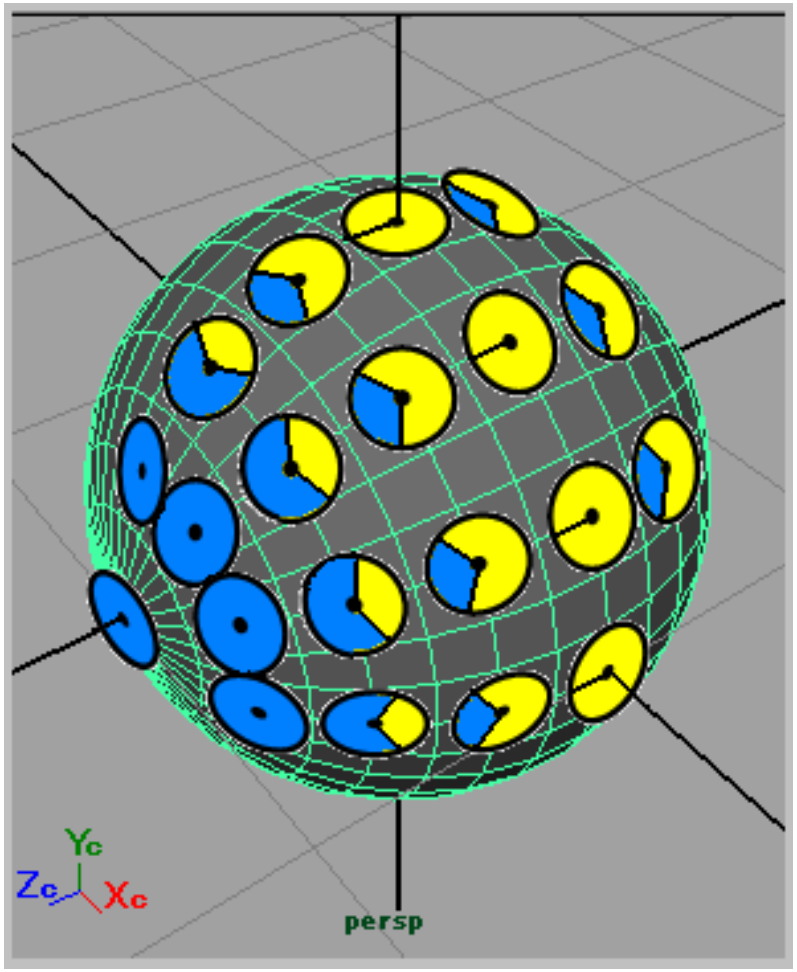
旋回範囲の制約のもとで，履歴依存型の手法を使う

特異点の解消

特異点付近では，
向き制限なし(旋回範囲が全方位)に設定

連続性の確保

球面S 上の位置に対して，
旋回範囲が連続的に変化



まとめ

- 履歴非依存型　→ 特性グリッド

  必ずフリップ → 1方向，方向指定。

- 履歴依存型

  フリップを回避できる。向きがずれる。

- ハイブリッド型

  フリップを回避できる。向きのずれを矯正。

ゲームシーンに合った選択を。

加速度センサの誤差測定

誤差測定の必要性

加速度センサには固体差がある

コントローラによって挙動が違うかも → 不安

その個体の誤差が分かれば

→ センサ値を補正

→ 誤差なし状態，誤差最大状態での動作テスト

→ 不安解消

誤差の意味

真の加速度値(重力加速度を含む): T

加速度センサによる測定値: M

誤差: E = M - T

1次式で近似 $E = e_0 + e_1 T$

$M = e_0 + (1 + e_1) T$

e_0 : 0G誤差，e_1 : 感度誤差

誤差測定(1軸)

$M = e_0 + (1 + e_1) T$

コントローラを静止させ，重力を測定

x軸を真上に向ける $T = 1G$, $M = m_1$

x軸を真下に向ける $T = -1G$, $M = m_2$

$m1 = e_0 + (1 + e_1)$

$m2 = e_0 - (1 + e_1)$

} e_0, e1 が得られる

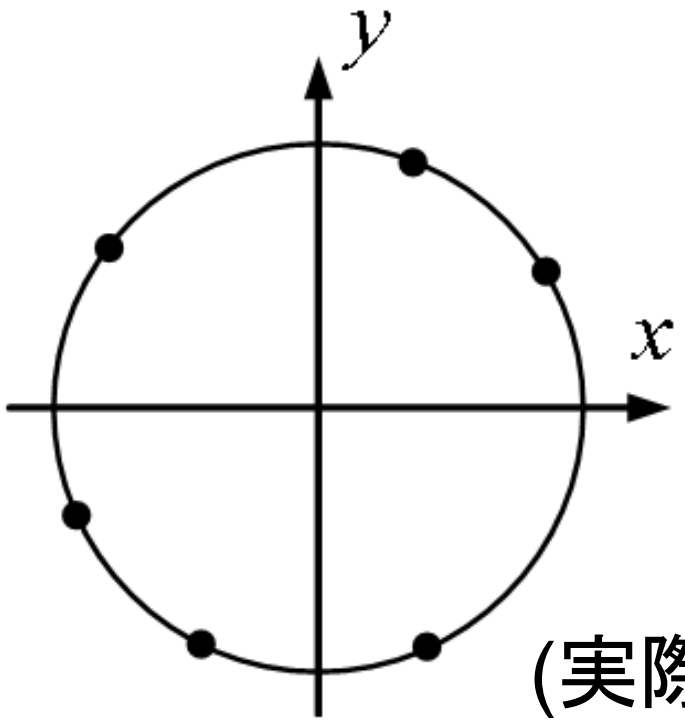
(y軸，z軸も同様)

方向合わせが面倒

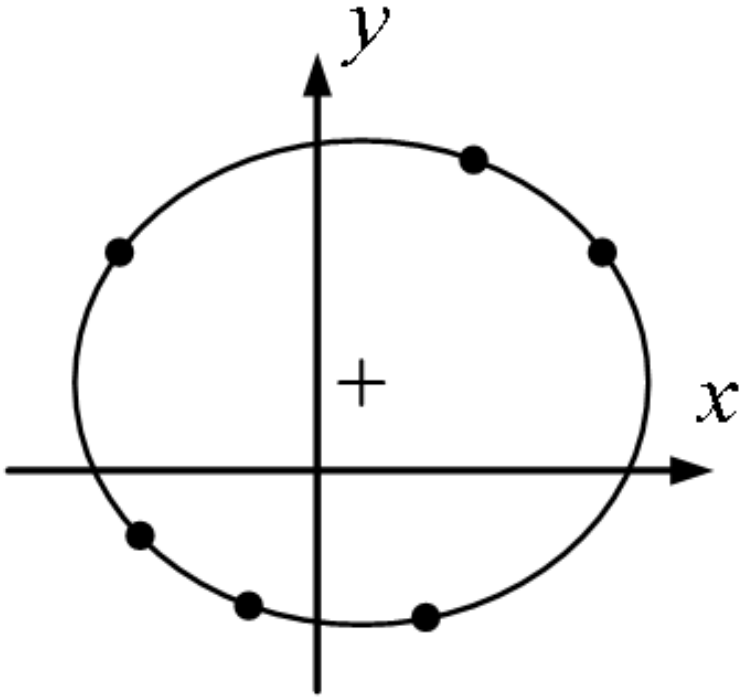
誤差測定(3軸)

コントローラを，様々な向きで静止させる
→ 加速度ベクトル(3次元)が得られる

誤差なし ⇒ 球



誤差あり ⇒ 楕円体



(実際は3次元)

楕円体の式

$$\left(\frac{x_i - p}{a}\right)^2 + \left(\frac{y_i - q}{b}\right)^2 + \left(\frac{z_i - r}{c}\right)^2 = 1$$

6個の測定値から

楕円体の式を求める(p, q, r, a, b, c を求める)

→ 0G誤差，感度誤差が得られる

方向合わせが不要

連立方程式

測定値 $(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_6, y_6, z_6)$

$$\left(\frac{x_1 - p}{a}\right)^2 + \left(\frac{y_1 - q}{b}\right)^2 + \left(\frac{z_1 - r}{c}\right)^2 = 1$$

$$\left(\frac{x_2 - p}{a}\right)^2 + \left(\frac{y_2 - q}{b}\right)^2 + \left(\frac{z_2 - r}{c}\right)^2 = 1$$

$$\cdots$$

連立2次方程式 → 解くのが困難

連立1次方程式に帰着できる

連立方程式の解法

$$\left(\frac{x_1-p}{a}\right)^2+\left(\frac{y_1-q}{b}\right)^2+\left(\frac{z_1-r}{c}\right)^2=1 \qquad \text{(a)}$$

$$\left(\frac{x_2-p}{a}\right)^2+\left(\frac{y_2-q}{b}\right)^2+\left(\frac{z_2-r}{c}\right)^2=1 \qquad \text{(b)}$$

(a) – (b)

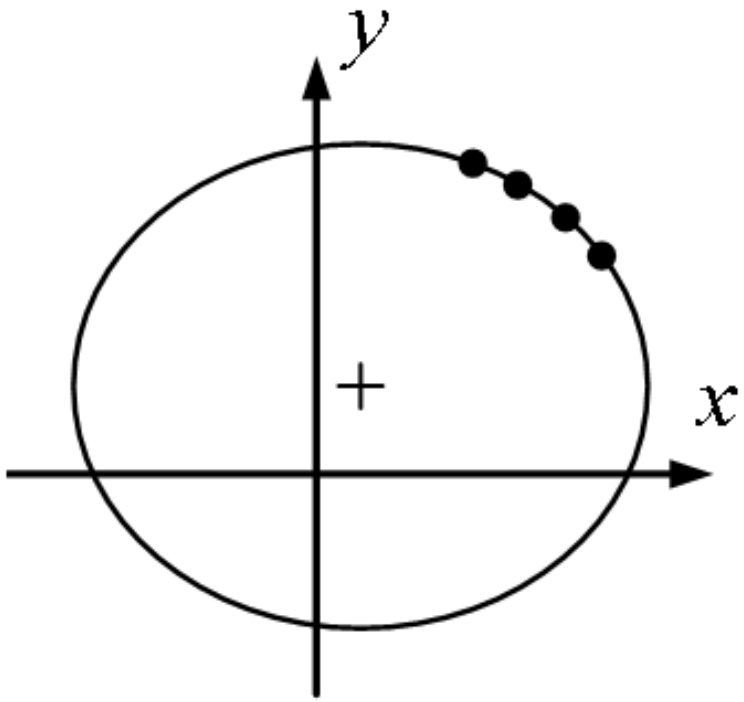
$$2(x_1-x_2)p+2(y_1-y_2)\frac{a^2}{b^2}q+2(z_1-z_2)\frac{a^2}{c^2}r$$
$$-(y_1^2-y_2^2)\frac{a^2}{b^2}-(z_1^2-z_2^2)\frac{a^2}{c^2}=x_1^2-x_2^2$$

□を別の変数に置換え → 未知数 5個の 1次式

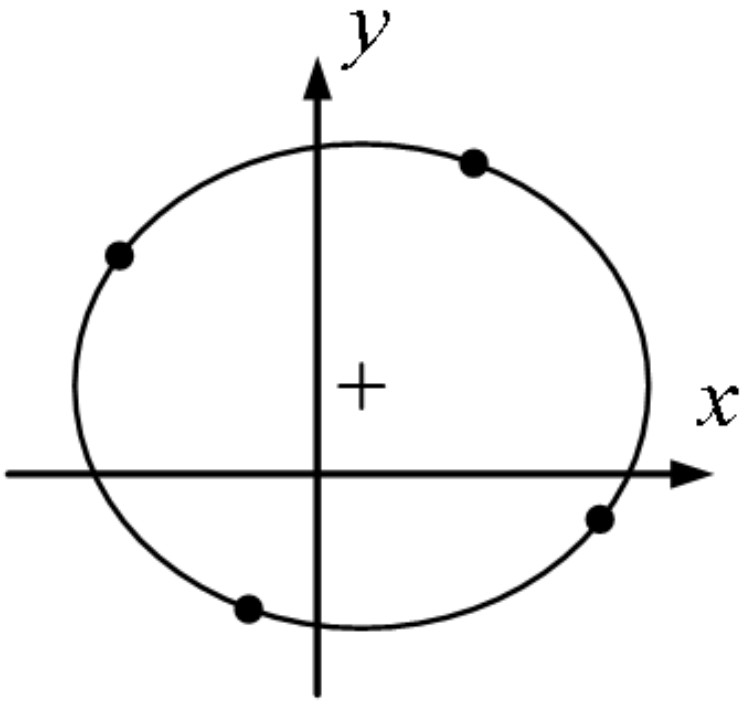
連立1次方程式に帰着 → 簡単に解ける

計算精度の向上

楕円体上で 6個の測定点を互いになるべく離す



NG



OK

(実際は3次元)

静止状態の検出

加速度の変動が 0.03G以下の状態が
1秒間持続
　→ 静止と見なす

1秒間に取得した値を平均
　→ 1個の測定値とする
　　ノイズや丸め誤差の影響を減らせる

まとめ

3軸加速度センサの誤差測定

楕円体を使う方法

　0G誤差，感度誤差が得られる

　測定時に方向合わせ不要

今後に向けて

加速度センサの活用

加速度センサは今後も使われる

多様な新コントローラの登場

- 新コントローラ → 加速度センサを更に生かす
- 新コントローラ ← 加速度センサの know-how