# 加速度センサを用いた図形入力

山口兼太郎*
(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

2/Sep '09

## 1 コントローラの移動軌跡の取得

近頃のゲーム機のコントローラや携帯電話には，入力用のデバイスとして，加速度センサが使われている。この加速度センサを使って，コントローラを空中で自由に動かしたときの軌跡を，図形として取得できれば，ゲーム等の入力として有用である。しかし，加速度センサ情報のみからコントローラの軌跡を求めることは，殆ど不可能である。その理由は主として次の 2 つである。

1. コントローラの向きが不明:
   加速度センサが検出する加速度ベクトルは，重力加速度と運動加速度 (動きによる加速度) との和である。軌跡の算出に必要な運動加速度を得るには，加速度センサ値から重力加速度を引き去る必要があるが，コントローラの向きが分からないため，正しく引き去ることができない。

2. 積分による誤差の集積:
   加速度から位置を求める際に，積分計算を 2 回行なう。そのため誤差が集積し，特に終点付近の誤差が大きくなる。

そこで，コントローラの動きに次の制約をつけて，加速度センサのみによる図形入力の実現性を調査する。

1. コントローラの移動を平行移動に限定する。
2. 始点と終点の位置関係を前もって指定する。

## 2 加速度，速度，位置の基本式

コントローラを動かして図形を入力している期間中，時間間隔 $\Delta t$ で加速度センサから値を取得するものとする。取得した時系列の値を $\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_{n-1}$ とする。初速度 (始点における速度) を $\mathbf{v}_0$ とし，時間間隔 $\Delta t$ 毎の時系列の速度値を $\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_n$ とする。同様に，初期位置 (始点の位置) を $\mathbf{p}_0$ とし，$\Delta t$ 毎の時系列の位置座標を $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{n+1}$ とする。加速度，速度，位置は，いずれも 3 次元のベクトルである。

*Kentaro_Yamaguchi@bandainamcogames.co.jp

図形を入力する際に，コントローラは平行移動する (回転しない) ものとする。そのため，加速度センサの値 $\mathbf{a}_i$ は，本来の加速度に対して，一様な重力加速度ベクトル $\mathbf{g}$ が付加されたものとなる。したがって，真の加速度は $\mathbf{a}_i - \mathbf{g}$ と表される。

速度 $\mathbf{v}_1, \ldots, \mathbf{v}_n$ は，初期値 $\mathbf{v}_0$ と次の漸化式から求められる。

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \Delta t\,(\mathbf{a}_i - \mathbf{g}) \tag{1}$$

位置 $\mathbf{p}_1, \ldots, \mathbf{p}_{n+1}$ は，初期値 $\mathbf{p}_0$ と次の漸化式から求められる。

$$\mathbf{p}_{i+1} = \mathbf{p}_i + \Delta t\,\mathbf{v}_i \tag{2}$$

速度の漸化式を解くと次式のようになる。

$$\begin{aligned}
\mathbf{v}_1 &= \mathbf{v}_0 + \Delta t\,(\mathbf{a}_0 - \mathbf{g}) \\
\mathbf{v}_2 &= \mathbf{v}_1 + \Delta t\,(\mathbf{a}_1 - \mathbf{g}) = \mathbf{v}_0 + \Delta t\,(\mathbf{a}_0 + \mathbf{a}_1) - 2\Delta t\,\mathbf{g} \\
&\vdots \\
\mathbf{v}_i &= \mathbf{v}_0 + \Delta t \sum_{k=0}^{i-1} \mathbf{a}_k - i\Delta t\,\mathbf{g} \\
&\vdots \\
\mathbf{v}_n &= \mathbf{v}_0 + \Delta t \sum_{k=0}^{n-1} \mathbf{a}_k - n\Delta t\,\mathbf{g}
\end{aligned} \tag{3}$$

位置の漸化式を解くと次式のようになる。

$$\begin{aligned}
\mathbf{p}_1 &= \mathbf{p}_0 + \Delta t\,\mathbf{v}_0 \\
\mathbf{p}_2 &= \mathbf{p}_1 + \Delta t\,\mathbf{v}_1 = \mathbf{p}_0 + \Delta t(\mathbf{v}_0 + \mathbf{v}_1) \\
&= \mathbf{p}_0 + 2\Delta t\,\mathbf{v}_0 + \Delta t^2(\mathbf{a}_0 - \mathbf{g}) \\
&\vdots \\
\mathbf{p}_i &= \mathbf{p}_0 + \Delta t \sum_{k=0}^{i-1} \mathbf{v}_k \\
&= \mathbf{p}_0 + i\Delta t\,\mathbf{v}_0 + \Delta t^2 \sum_{k=0}^{i-2} (i-k-1)\mathbf{a}_k - \frac{i(i-1)}{2}\Delta t^2 \mathbf{g} \\
&\vdots \\
\mathbf{p}_{n+1} &= \mathbf{p}_0 + \Delta t \sum_{k=0}^{n} \mathbf{v}_k \\
&= \mathbf{p}_0 + (n+1)\Delta t\,\mathbf{v}_0 + \Delta t^2 \sum_{k=0}^{n-1} (n-k)\mathbf{a}_k - \frac{n(n+1)}{2}\Delta t^2 \mathbf{g}
\end{aligned} \tag{4}$$

ここで，$\mathbf{s}_1, \mathbf{s}_2$ を次式のように定める。

$$\mathbf{s}_1 = \sum_{k=0}^{n-1} \mathbf{a}_k \tag{5}$$

$$\mathbf{s}_2 = \sum_{k=0}^{n-1} (n-k)\mathbf{a}_k \tag{6}$$

$\mathbf{s}_1, \mathbf{s}_2$ を使うと $\mathbf{v}_n, \mathbf{p}_{n+1}$ は次式のように表される。

$$\mathbf{v}_n = \mathbf{v}_0 + \Delta t\, \mathbf{s}_1 - n\Delta t\, \mathbf{g} \tag{7}$$

$$\mathbf{p}_{n+1} = \mathbf{p}_0 + (n+1)\Delta t\, \mathbf{v}_0 + \Delta t^2 \mathbf{s}_2 - \frac{n(n+1)}{2}\Delta t^2 \mathbf{g} \tag{8}$$

## 3 図形入力時の制限と位置の導出

コントローラで図形を入力する際に，次の 2 つの制限を設ける。

1. 始点と終点を一致させる。
2. 終点でコントローラを止める。

この制限により，

$$\mathbf{p}_0 = \mathbf{p}_{n+1} = \mathbf{0} \tag{9}$$

$$\mathbf{v}_n = \mathbf{0} \tag{10}$$

となる (始点を位置座標の原点とする)。

式 (7), (8), (9), (10) より $\mathbf{g}, \mathbf{v}_0$ を求めると，次式のようになる。

$$\mathbf{g} = \frac{2}{n}\mathbf{s}_1 - \frac{2}{n(n+1)}\mathbf{s}_2 \tag{11}$$

$$\mathbf{v}_0 = \Delta t \left( \mathbf{s}_1 - \frac{2}{n+1}\mathbf{s}_2 \right) \tag{12}$$

式 (9), (11), (12) を式 (4) に代入すれば，時系列の位置 $\mathbf{p}_i$ が求められる。$\mathbf{p}_i$ の値は，コントローラ (加速度センサ) 座標系における 3 次元の値として得られる。式 (4) を使う替わりに，漸化式 (1), (2) を使ってもよい。通常は，漸化式を使うほうが計算量は少なくなる。

式 (11) で求めた $\mathbf{g}$ は，コントローラ座標系における重力加速度である。$\mathbf{p}_i$ と $\mathbf{g}$ から，入力された図形が鉛直方向に対してどれだけ傾いているかを求めることができる。

## 4 加速度センサのバイアス誤差

加速度センサがバイアス誤差 (0G 誤差とも言う) を持っている場合について考える。誤差を $\mathbf{e}$ とすると，加速度センサの値は常に $\mathbf{e}$ だけずれているので，真の加速度は $\mathbf{a}_i - \mathbf{g} - \mathbf{e}$ となる。

$\mathbf{g}'$ を次のように定義すると，前節までの $\mathbf{g}$ を全て $\mathbf{g}'$ で置き換えることができる。

$$\mathbf{g}' = \mathbf{g} + \mathbf{e} \tag{13}$$

したがって，加速度センサがバイアス誤差を持っている場合でも，誤差なしの場合と同様の方法で，コントローラの軌跡を求めることができる。

## 5 入力制限のバリエーション (その 1)

図形入力時の制限は，第 3 節で設定した制限以外に，以下に挙げる方法も可能である。これらの入力制限を用いた場合でも，加速度センサのバイアス誤差を第 4 節の方法で扱うことができる。

### 5.1 終点位置と終点速度の指定

次のように入力制限を設定する。

1. 始点から所定の方向に所定の距離だけ離れた点を終点とする。
2. 終点でコントローラを止める。

始点を位置座標の原点とし，終点位置を $\mathbf{p}_e$ とすると，この制限は次式で表される。

$$\mathbf{p}_0 = \mathbf{0} \tag{14}$$

$$\mathbf{p}_{n+1} = \mathbf{p}_e \tag{15}$$

$$\mathbf{v}_n = \mathbf{0} \tag{16}$$

この制限で $\mathbf{p}_e = \mathbf{0}$ とした場合が，第 3 節で扱った入力制限に相当する。

式 (7), (8), (14), (15), (16) より $\mathbf{g}, \mathbf{v}_0$ を求めると，次式のようになる。

$$\mathbf{g} = \frac{2}{n}\mathbf{s}_1 - \frac{2}{n(n+1)}\mathbf{s}_3 \tag{17}$$

$$\mathbf{v}_0 = \Delta t\left(\mathbf{s}_1 - \frac{2}{n+1}\mathbf{s}_3\right) \tag{18}$$

ただし，

$$\mathbf{s}_3 = \mathbf{s}_2 - \frac{\mathbf{p}_e}{\Delta t^2} \tag{19}$$

である。式 (14), (17), (18) を式 (4) に代入すれば，時系列の位置 $\mathbf{p}_i$ が得られる。

なお，$\mathbf{v}_n = \mathbf{0}$ なので $\mathbf{p}_n = \mathbf{p}_{n+1}$ となる。図形描画用の頂点列としては $\mathbf{p}_{n+1}$ は不要で，$\mathbf{p}_0, \ldots, \mathbf{p}_n$ だけを使えばよい。

### 5.2 終点位置と始点速度の指定

次のように入力制限を設定する。

1. 始点から所定の方向に所定の距離だけ離れた点を終点とする。
2. 始点でコントローラを止めた状態から入力を開始する。

始点を位置座標の原点とし，終点位置を $\mathbf{p}_e$ とすると，この制限は次式で表される。

$$\mathbf{p}_0 = \mathbf{0} \tag{20}$$

$$\mathbf{p}_{n+1} = \mathbf{p}_e \tag{21}$$

$$\mathbf{v}_0 = \mathbf{0} \tag{22}$$

式 (8), (20), (21), (22) より $\mathbf{g}$ を求めると，次式のようになる。

$$\mathbf{g} = \frac{2}{n(n+1)}\mathbf{s}_3 \tag{23}$$

式 (20), (22), (23) を式 (4) に代入すれば，時系列の位置 $\mathbf{p}_i$ が得られる。

なお，$\mathbf{v}_0 = \mathbf{0}$ なので $\mathbf{p}_0 = \mathbf{p}_1$ となる。図形描画用の頂点列としては $\mathbf{p}_0$ は不要で，$\mathbf{p}_1, \ldots, \mathbf{p}_{n+1}$ だけを使えばよい。

### 5.3 始点・終点速度の指定

次のように入力制限を設定する。

1. 始点でコントローラを止めた状態から入力を開始する。
2. 終点でコントローラを止める。

始点を位置座標の原点とすると，この制限は次式で表される。

$$\mathbf{p}_0 = \mathbf{0} \tag{24}$$

$$\mathbf{v}_0 = \mathbf{v}_n = \mathbf{0} \tag{25}$$

式 (7), (25) より $\mathbf{g}$ を求めると，次式のようになる。

$$\mathbf{g} = \frac{1}{n}\mathbf{s}_1 \tag{26}$$

式 (24), (25), (26) を式 (4) に代入すれば，時系列の位置 $\mathbf{p}_i$ が得られる。

なお，$\mathbf{v}_0 = \mathbf{v}_n = \mathbf{0}$ なので $\mathbf{p}_0 = \mathbf{p}_1$, $\mathbf{p}_n = \mathbf{p}_{n+1}$ となる。図形描画用の頂点列としては $\mathbf{p}_0, \mathbf{p}_{n+1}$ は不要で，$\mathbf{p}_1, \ldots, \mathbf{p}_n$ だけを使えばよい。

## 6 入力制限のバリエーション (その 2)

### 6.1 入力制限の追加

次のように入力制限を設定する。

1. 始点から所定の方向に所定の距離だけ離れた点を終点とする。
2. 始点でコントローラを止めた状態から入力を開始する。
3. 終点でコントローラを止める。

始点を位置座標の原点とし，終点位置を $\mathbf{p}_e$ とすると，この制限は次式で表される。

$$\mathbf{p}_0 = \mathbf{0} \tag{27}$$

$$\mathbf{p}_{n+1} = \mathbf{p}_e \tag{28}$$

$$\mathbf{v}_0 = \mathbf{v}_n = \mathbf{0} \tag{29}$$

これまでに述べた入力制限方法の場合は，条件の個数と未知数の個数とが一致するので，解を一意に求めることができた。しかし，上記のような制限の下では，条件の個数が未知数の個数よりも 1 個 (3 次元ベクトルとして 1 個，スカラーとしては 3 個) 多くなるため，解が得られない。そこで，前述の 2 つの方法から得られる解を，以下のようにブレンドして解を求める。

まず，条件を $\mathbf{p}_0 = \mathbf{0}$, $\mathbf{p}_{n+1} = \mathbf{p}_e$, $\mathbf{v}_n = \mathbf{0}$ のみとし，第 5.1 節で述べた方法でコントローラの位置を求め，これを $\mathbf{p}_i^e$ $(i = 0, \ldots, n+1)$ とする。次に，条件を $\mathbf{p}_0 = \mathbf{0}$, $\mathbf{p}_{n+1} = \mathbf{p}_e$, $\mathbf{v}_0 = \mathbf{0}$ のみとし，第 5.2 節で述べた方法で位置を求め，これを $\mathbf{p}_i^s$ $(i = 0, \ldots, n+1)$ とする。$\mathbf{p}_i^e$ は，終点速度を指定して求めた位置なので，終点付近すなわち $i$ が大きい ($n+1$ に近い) 場合には精度が良いが，始点付近すなわち $i$ が小さい場合には精度が悪くなる。一方 $\mathbf{p}_i^s$ は逆に，始点付近では精度が良いが，終点付近では精度が悪い。

したがって，始点付近では $\mathbf{p}_i^s$ の割合が大きく，終点付近では $\mathbf{p}_i^e$ の割合が大きくなるように，両者をブレンドすれば，元の $\mathbf{p}_i^e$, $\mathbf{p}_i^s$ よりも精度の良い位置座標を得ることができる。$\mathbf{v}_0 = \mathbf{v}_n = \mathbf{0}$

を前提としているので，$\mathbf{p}_0 = \mathbf{p}_1$, $\mathbf{p}_n = \mathbf{p}_{n+1}$ となり，図形描画用の頂点列としては $\mathbf{p}_0$, $\mathbf{p}_{n+1}$ は不要である。そこで次式のように，$\mathbf{p}^e_1, \ldots, \mathbf{p}^e_n$ と $\mathbf{p}^s_1, \ldots, \mathbf{p}^s_n$ をブレンドし，$\mathbf{p}^b_0, \ldots, \mathbf{p}^b_{n-1}$ を作る。

$$\mathbf{p}^b_i = \frac{i\,\mathbf{p}^e_{i+1} + (n-i-1)\mathbf{p}^s_{i+1}}{n-1} \tag{30}$$

すなわち，$\mathbf{p}^e_{i+1}$ と $\mathbf{p}^s_{i+1}$ の $i : (n-i-1)$ の荷重平均を $\mathbf{p}^b_i$ とし，これを解とする。始点では $\mathbf{p}^b_0 = \mathbf{p}^s_1 = \mathbf{0}$ となり，終点では $\mathbf{p}^b_{n-1} = \mathbf{p}^e_n = \mathbf{p}_e$ となる。

重力ベクトルについても，位置の計算と同様に 2 つの方法で値を求め，その平均を用いる。第 5.1 節の方法 (終点速度 0) で，式 (17) により重力ベクトルを求め，これを $\mathbf{g}^e$ とする。第 5.2 節の方法 (始点速度 0) で，式 (23) により重力ベクトルを求め $\mathbf{g}^s$ とする。

$$\mathbf{g}^e = \frac{2}{n}\,\mathbf{s}_1 - \frac{2}{n(n+1)}\,\mathbf{s}_3 \tag{31}$$

$$\mathbf{g}^s = \frac{2}{n(n+1)}\,\mathbf{s}_3 \tag{32}$$

次式のように，両者の平均を $\mathbf{g}^b$ とし，これを重力ベクトルの解とする。

$$\begin{aligned}\mathbf{g}^b &= \left(\mathbf{g}^e + \mathbf{g}^s\right)/2 \\ &= \frac{1}{n}\mathbf{s}_1\end{aligned} \tag{33}$$

## 6.2 ブレンドの効果

実際の入力データを使ってブレンドの効果を検証した。図 1〜3 は，同一のデータから 3 とおりの方法で軌跡を求めた結果である。データ取得時には，始点速度と終点速度が共に 0 となるように意識しながら，コントローラを空中で丸を描くように動かした。

終点速度 0 の条件で求めた軌跡 (図 1) では始点付近に歪が見られ，始点速度 0 の条件で求めた軌跡 (図 2) では終点付近に歪が見られるが，両者をブレンドして得られた軌跡 (図 3) では歪が軽減されている。

同様にして，星型を描くようにコントローラを動かし，取得したデータを使って比較した結果が，図 4〜6 である。やはり，ブレンドによる歪の軽減が見られる。

## 6.3 漸化式による計算

前節で述べた手順にしたがって一旦 $\mathbf{p}^e_i, \mathbf{p}^s_i$ を求めてから，式 (30) を使って $\mathbf{p}^b_i$ を計算すると，$\mathbf{p}^e_i$ または $\mathbf{p}^s_i$ を保存するためのバッファが必要となる。そこで，メモリ使用量と計算量を抑えるために，次のような漸化式を使って $\mathbf{p}^b_i$ を直接求めることにする。

$$\mathbf{v}^b_{i+1} = \mathbf{v}^b_i + \Delta t\,\mathbf{a}^b_i \tag{34}$$

$$\mathbf{p}^b_{i+1} = \mathbf{p}^b_i + \Delta t\,\mathbf{v}^b_i \tag{35}$$

式 (34), (35) を変形すると，それぞれ次式のようになる。

$$\mathbf{a}^b_i = \left(\mathbf{v}^b_{i+1} - \mathbf{v}^b_i\right)/\Delta t \tag{36}$$

$$\mathbf{v}^b_i = \left(\mathbf{p}^b_{i+1} - \mathbf{p}^b_i\right)/\Delta t \tag{37}$$

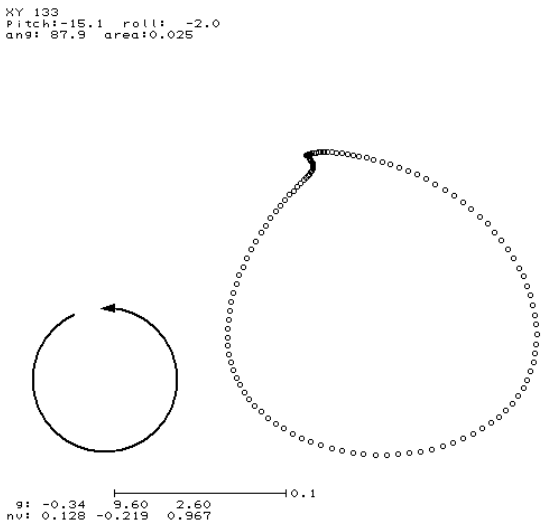


図 1: 終点速度 0 の条件で求めた軌跡

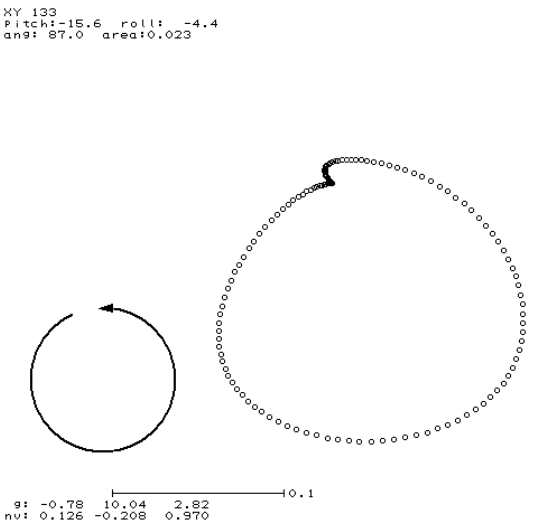


図 2: 始点速度 0 の条件で求めた軌跡

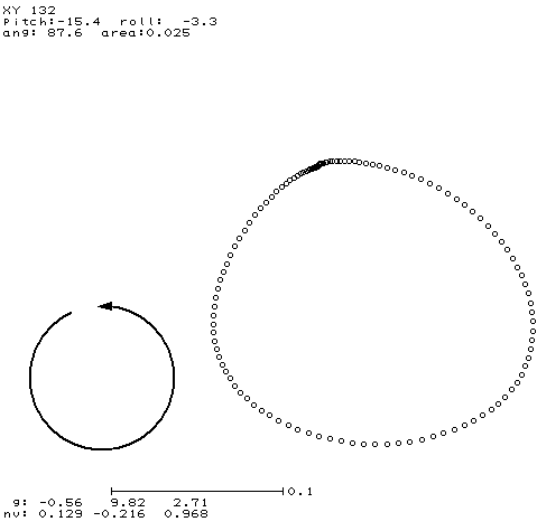


図 3: ブレンドにより求めた軌跡

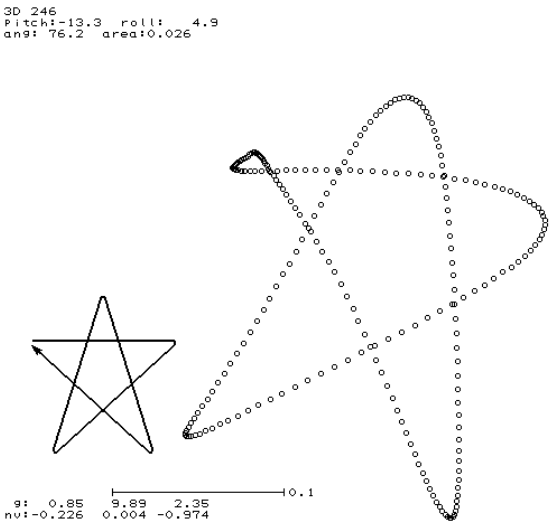


図 4: 終点速度 0 の条件で求めた軌跡

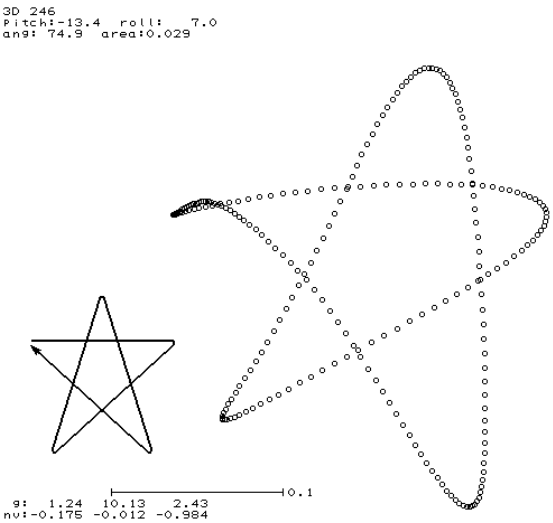


図 5: 始点速度 0 の条件で求めた軌跡

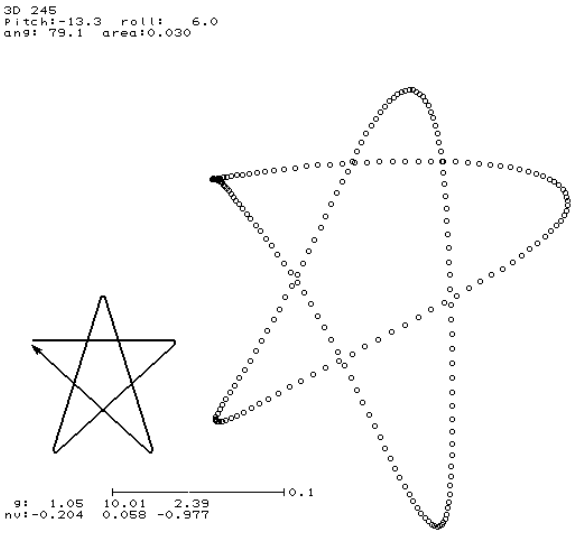


図 6: ブレンドにより求めた軌跡

式 (30), (36), (37) から $\mathbf{v}_0^b$ および $\mathbf{a}_i^b$ を求めると，次式のようになる。

$$\mathbf{v}_0^b = \Delta t \left[ \mathbf{a}_0 + \frac{2}{n}\mathbf{s}_1 - \frac{6}{n(n+1)}\mathbf{s}_3 \right] \tag{38}$$

$$\mathbf{a}_i^b = \mathbf{a}_{i+1} - \frac{2(3i-n+4)}{n(n-1)}\mathbf{s}_1 + \frac{6(2i-n+3)}{n(n-1)(n+1)}\mathbf{s}_3 \tag{39}$$

この $\mathbf{v}_0^b, \mathbf{a}_i^b$ と漸化式 (34) を使えば，$\mathbf{v}_1^b, \ldots, \mathbf{v}_{n-2}^b$ が得られ，更に $\mathbf{p}_0^b = \mathbf{0}$ と漸化式 (35) を使えば，$\mathbf{p}_1^b, \ldots, \mathbf{p}_{n-1}^b$ を求めることができる。

# 7 テスト結果

テストの結果，以下のことが分かった。

入力時間が長くなると，得られる図形形状の歪が大きくなる。実用的には 2 秒程度が限界である。歪の主原因として考えられるのは，コントローラの平行移動という制約条件に対して，実際の動きが平行移動ではないことである。平行移動を意識してコントローラを動かしても，無意識のうちに手首のスナップ挙動により向きが変化しがちである。コントローラをうまく平行移動させるには，慣れやコツが必要である。

また，本手法では入力が完了するまでは図形形状が分からないが，この点に対して，ゲーム入力として使いにくいという指摘があった。

# 8 まとめ

コントローラの動きに以下の制約をつければ，加速度センサ情報のみを使ってコントローラの軌跡を求めることができる。

- コントローラの移動を平行移動に限定。
- 始点と終点の位置関係を前もって指定。
- 始点速度 0，終点速度 0。

ただし，実際に使用する際には，以下の点に注意が必要である。

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

# 付録 A 誤差の最小化

第 6.1 節において，$\mathbf{p}_i^e$ と $\mathbf{p}_i^s$ をブレンドして誤差を減らすことを試みた。そこで，誤差を最小にするブレンド方法について検討する。ただし，ここでは $\mathbf{a}_i$ のランダムで独立な誤差のみを対象とする。実際には，$\mathbf{a}_i$ の誤差には独立でない誤差も多く含まれているので，以下の検討から得られる指針は，あくまでも参考レベルである。

$\mathbf{p}_i^e$ を $\mathbf{a}_k\,(k=0,\ldots,n-1)$ の 1 次式で表したときの，$\mathbf{a}_k$ の係数を $c_{i,k}$ とする。$c_{i,k}$ は次のようになる。

$$c_{i,k}=\begin{cases}\dfrac{-(k+1)(i-n)(i-n-1)}{n(n+1)}\Delta t^2 & (0\le k\le i-2)\\ \dfrac{-i(n^2-2kn-n+ik-k+i-1)}{n(n+1)}\Delta t^2 & (i-1\le k\le n-1)\end{cases} \tag{40}$$

$\mathbf{p}_i^s$ を $\mathbf{a}_k\,(k=0,\ldots,n-1)$ の 1 次式で表したときの，$\mathbf{a}_k$ の係数を $d_{i,k}$ とする。$d_{i,k}$ は次のようになる。

$$d_{i,k}=\begin{cases}\dfrac{-(n-i+1)(kn-in+n+ik)}{n(n+1)}\Delta t^2 & (0\le k\le i-2)\\ \dfrac{-i(i-1)(n-k)}{n(n+1)}\Delta t^2 & (i-1\le k\le n-1)\end{cases} \tag{41}$$

次式のように $\mathbf{p}_i^e$ と $\mathbf{p}_i^s$ を $\alpha_i:(1-\alpha_i)$ の割合でブレンドし，その結果を $\mathbf{p}_i^m$ とする。

$$\mathbf{p}_i^m=\alpha_i\,\mathbf{p}_i^e+(i-\alpha_i)\mathbf{p}_i^s \tag{42}$$

各 $\mathbf{a}_i$ の誤差の大きさ (標準偏差) を $\sigma_a$ とする。標準偏差を $D()$, 分散 (標準偏差の 2 乗) を $D^2()$ で表すことにする。$\mathbf{p}_i^m$ の分散は次式のようになる。

$$D^2(\mathbf{p}_i^m)=\sum_{k=0}^{n-1}\left[\left[\alpha_i\,c_{i,k}+(1-\alpha_i)d_{i,k}\right]\sigma_a\right]^2 \tag{43}$$

$D^2(\mathbf{p}_i^m)$ を最小にする $\alpha_i$ を求めるために，次式を $\alpha_i$ について解く。

$$\frac{d}{d\alpha_i}D^2(\mathbf{p}_i^m)=0 \tag{44}$$

結果として，次式が得られる。

$$\alpha_i=\frac{i-1}{n-1} \tag{45}$$

ブレンド率 $\alpha_i$ は $i$ に関してリニアに変化する。この $\alpha_i$ を使って求めた $\mathbf{p}_i^m$ を $\mathbf{p}_i^*$ とする。

$$\mathbf{p}_i^*=\left[(i-1)\mathbf{p}_i^e+(n-i)\mathbf{p}_i^s\right]/(n-1) \tag{46}$$

$\mathbf{p}_i^*$ の分散は次式のようになる。

$$D^2(\mathbf{p}_i^*)=\frac{i(i-1)(n-i)(n-i+1)(2ni-2i^2-n+2i+1)}{6n(n-1)(n+1)}\Delta t^4\,\sigma_a^2 \tag{47}$$

一方，$\mathbf{p}_i^e,\mathbf{p}_i^s$ の分散は次のようになる。

$$\begin{aligned}D^2(\mathbf{p}_i^e)&=\sum_{k=0}^{n-1}(c_{i,k}\,\sigma_a)^2\\&=\frac{i(n-i)(n-i+1)(2ni-2i^2+i+1)}{6n(n+1)}\Delta t^4\,\sigma_a^2\end{aligned} \tag{48}$$

$$\begin{aligned}D^2(\mathbf{p}_i^s)&=\sum_{k=0}^{n-1}(d_{i,k}\,\sigma_a)^2\\&=\frac{i(i-1)(n-i+1)(2ni-2i^2-n+3i)}{6n(n+1)}\Delta t^4\,\sigma_a^2\end{aligned} \tag{49}$$

$\mathbf{p}_i^*, \mathbf{p}_i^e, \mathbf{p}_i^s$ の標準偏差を比較すると，図 7 のようになる。$D(\mathbf{p}_i^*)$ が $D(\mathbf{p}_i^e), D(\mathbf{p}_i^s)$ よりも小さくなっていることが分かる。

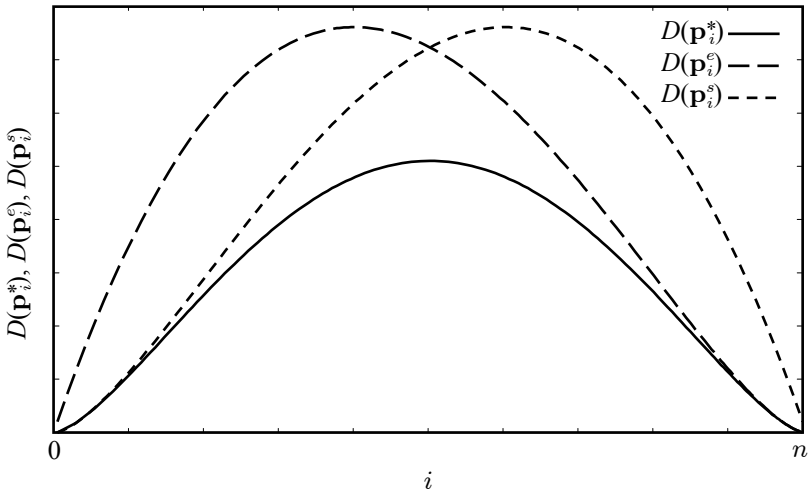


図 7: 位置誤差の比較